PROYECTO DE TÍTULO:

La matriz educativa, una gestión visionaria.

Integrante: Richard Alexander Wilson Rodríguez.

Carrera: Analista Programador.

# Introducción

# **1.1 Contexto y Relevancia del Proyecto**

# La educación, como eje central de la sociedad, enfrenta el desafío de gestionar de manera eficiente sus procesos administrativos, garantizando acceso equitativo y calidad en el servicio. En Chile, los colegios han evidenciado significativos problemas en la gestión de matrículas, como lo demuestra un déficit de más de 3.000 matrículas en educación básica y media en 2024, concentrándose especialmente en las regiones de Antofagasta y Copiapó (El Dínamo, 2024). Además, se ha observado un preocupante aumento en la deserción escolar, que superó los 50.000 estudiantes en 2022, representando un 24% más que en 2019 (La Tribuna, 2022). Entre 2015 y 2023, el cierre de más de 745 establecimientos educativos exacerbó el problema de disponibilidad de cupos (Noticias UNAB, 2023).

# La gestión manual de matrículas incrementa los errores operativos, las demoras administrativas y los costos asociados, evidenciando la necesidad de incorporar tecnologías que optimicen estos procesos. Este proyecto busca desarrollar un sistema web innovador de gestión de matrículas basado en Firebase, con un enfoque en resolver problemas estructurales, aumentar la eficiencia y reducir costos operativos.

# El sistema propuesto incluye funcionalidades específicas que lo diferencian de soluciones existentes:

# Sincronización en tiempo real: Garantiza el acceso inmediato a información actualizada sobre matrículas y cupos.

# Seguridad avanzada: Utiliza Firebase Authentication para proteger los datos sensibles.

# Escalabilidad: Se adapta al crecimiento de las demandas escolares y permite gestionar grandes volúmenes de datos sin comprometer el rendimiento.

# Diseño web responsivo: Compatible con diversos dispositivos, garantizando una experiencia fluida desde navegadores web.

# Este enfoque tecnológico posicionará a las instituciones educativas como referentes en innovación, mejorando la experiencia de estudiantes y familias mientras optimiza recursos y planificación estratégica.

# **Capítulo II: Antecedentes del Proyecto**

# 2.1 Planteamiento del Problema

# La gestión de matrículas en el sistema educativo chileno presenta fallas estructurales, agravadas por la falta de tecnologías modernas. La administración manual genera un margen de error estimado en un 15% debido a inconsistencias en los registros, duplicación de datos y errores en la asignación de cupos (Ministerio de Educación, 2024). Este problema, combinado con la carencia de plataformas integradas, dificulta la comunicación y aumenta los costos operativos.

# Adicionalmente, los procesos manuales suelen demorar más de 10 días hábiles en regiones con alta demanda, afectando a más de 880 estudiantes que quedaron sin cupo en 2024 (Radio Universidad de Chile, 2024). Esto no solo impacta negativamente en las familias, sino que también sobrecarga a los administradores escolares, quienes carecen de herramientas para gestionar datos en tiempo real.

# En un estudio reciente, el 63% de los colegios públicos reportó no contar con plataformas tecnológicas adecuadas para la gestión escolar, mientras que aquellos que utilizan sistemas digitales enfrentan altos costos de licenciamiento y mantenimiento (Fundación Educación, 2023). Estos factores hacen evidente la necesidad de una solución accesible y escalable que responda a los desafíos del contexto chileno.

# 2.2 Justificación del Proyecto

# El desarrollo de un sistema automatizado basado en Firebase es crucial para abordar las ineficiencias en la gestión de matrículas. Firebase ofrece ventajas competitivas, como la sincronización en tiempo real, que permite acceder a información actualizada de matrículas y cupos; la seguridad avanzada, que protege datos sensibles mediante Firebase Authentication; y la escalabilidad, que facilita la adaptación a las necesidades crecientes de las instituciones escolares.

# A diferencia de los sistemas existentes, este proyecto introduce elementos innovadores:

# Notificaciones automatizadas personalizadas: Mantienen informadas a las familias sobre actualizaciones importantes, como cambios en el estado de matrícula y disponibilidad de cupos.

# Reportes visuales interactivos: Proveen herramientas para analizar tendencias, optimizar recursos y mejorar la toma de decisiones.

# Diseño web responsivo: Garantiza una experiencia de usuario accesible desde cualquier dispositivo con navegadores modernos.

# Además, el sistema busca reducir costos operativos al eliminar procesos manuales y consolidar la información en una plataforma centralizada, permitiendo a las instituciones destinar recursos a la mejora de la calidad educativa. Este enfoque no solo resuelve problemas técnicos, sino que también refuerza la sostenibilidad económica de los colegios, posicionándolos como líderes en innovación educativa dentro de un entorno competitivo.

# En síntesis, este proyecto busca impactar positivamente en la experiencia de estudiantes y familias, optimizando la comunicación, reduciendo errores y fortaleciendo la planificación estratégica de las instituciones escolares, cumpliendo así con los objetivos de modernización en el sector educativo chileno.

# 2.3 Propuesta de Solución

# El proyecto propone el desarrollo de un sistema web de gestión de matrículas que utilice Firebase como base tecnológica. Entre sus funcionalidades principales, se incluirán:

# Registro y autenticación de usuarios: Mediante Firebase Authentication, asegurando un acceso seguro y controlado para administradores y apoderados.

# Gestión de datos en tiempo real: Utilizando Firestore, permitiendo la actualización inmediata de información sobre matrículas y cupos.

# Generación automática de reportes: Basados en datos almacenados, facilitando el análisis de tendencias y la planificación estratégica de las instituciones.

# Notificaciones automatizadas: Enviadas a los apoderados directamente desde el navegador, informando sobre cambios en el estado de matrículas, fechas clave y disponibilidad de cupos.

# Diseño web responsivo: Asegurando accesibilidad y usabilidad desde dispositivos de escritorio y móviles.

# Acceso seguro y escalable: Garantizado mediante Firebase Hosting y Firebase Security Rules, adaptándose al crecimiento de las demandas escolares.

# Esta solución optimiza procesos administrativos, reduce costos operativos y posiciona a los colegios como líderes en innovación educativa en un entorno cambiante y competitivo.

# 2.4 Propuesta de Valor

El uso de Firebase, complementado con MongoDB Atlas, posiciona esta aplicación como una solución innovadora para la gestión educativa, destacándose por su capacidad de automatizar la transferencia de datos entre plataformas. Este enfoque garantiza la sincronización en tiempo real y la integridad de la información, optimizando procesos clave como el registro de usuarios, la gestión de matrículas y la asignación de apoderados.

La aplicación no solo permite centralizar la administración de datos, sino que también automatiza la compartición entre sistemas, lo que elimina redundancias y mejora la eficiencia operativa. Los usuarios pueden acceder a información actualizada en cualquier momento, mientras que las instituciones aprovechan un entorno confiable y escalable para sus necesidades administrativas.

Esta capacidad de integración y automatización proporciona a los colegios una herramienta poderosa para tomar decisiones basadas en datos precisos, modernizando su gestión y ofreciendo una experiencia educativa de calidad que cumple con los estándares más exigentes de la administración digital.

# **Capítulo III: Objetivos y Alcance**

# 3.1 Objetivo General

# Desarrollar un sistema web de gestión de matrícula y control para el año 2025, que centralice y automatice el proceso administrativo utilizando Firebase, asegurando un tiempo de respuesta menor a 2 segundos en actualizaciones de datos, una reducción del 30% en los tiempos administrativos y un acceso seguro con un 100% de autenticaciones verificadas para colegios, estudiantes y apoderados.

# 3.2 Objetivos Específicos

# Diseñar y crear una base de datos en Firestore para administrar la información de 700 estudiantes y sus apoderados, con actualizaciones en tiempo real procesadas en menos de 2 segundos por transacción y un margen de error inferior al 1%. El diseño deberá completarse en un plazo máximo de 30 días hábiles, garantizando escalabilidad para manejar un crecimiento anual del 20% en los registros. Este sistema optimizará la eficiencia y reducirá errores en la gestión administrativa.

# Configurar un sistema de autenticación seguro con Firebase Authentication, garantizando un 100% de accesos verificados para 700 usuarios registrados y un tiempo promedio de inicio de sesión menor a 5 segundos, medido con herramientas de monitoreo de Firebase. Se realizarán auditorías mensuales para evaluar seguridad y detectar accesos no autorizados, completando la implementación en 15 días hábiles.

# Implementar funcionalidades para generar reportes automáticos de inscripción, cupos y tasas de matrícula completada, cubriendo el 90% de las métricas relevantes para la planificación escolar. Los reportes deberán generarse en menos de 5 segundos, ser accesibles en tiempo real desde el navegador y validarse mediante pruebas funcionales con 10 usuarios.

# Incorporar herramientas de supervisión con Firebase Analytics para rastrear métricas como tasas de finalización y tiempos promedio por etapa, garantizando un 95% de precisión. Los informes se generarán en menos de 5 segundos y se validarán mediante pruebas funcionales con 10 usuarios. La implementación se completará en 20 días hábiles.

# Realizar pruebas y validaciones del sistema con herramientas de Firebase, garantizando un 99.9% de uptime y un rendimiento estable con 100 usuarios concurrentes. Las pruebas se completarán en 10 días hábiles, asegurando un 95% de éxito en las funcionalidades evaluadas y generando informes al finalizar cada fase.

# 3.3 Alcance del Proyecto (Actualizado)

# El sistema cubrirá las siguientes funcionalidades principales:

# Matrícula de estudiantes: Registro y gestión de estudiantes nuevos y antiguos, incluyendo su información personal, académica y estado de matrícula, con actualizaciones en tiempo real mediante Firestore.

# Gestión de apoderados: Registro de datos de apoderados, vinculación con estudiantes y notificaciones automatizadas sobre actualizaciones importantes, como disponibilidad de cupos y cambios en el estado de matrícula.

# Reportes automatizados: Generación de reportes personalizados en tiempo real, como tasas de finalización de matrículas, disponibilidad de cupos y análisis estadístico, accesibles desde el navegador.

# Acceso multiusuario: Control de roles y autenticación segura para administradores, apoderados y estudiantes, gestionados mediante Firebase Authentication.

# Seguridad: Implementación de reglas avanzadas de seguridad con Firebase Security Rules, garantizando acceso protegido y manejo seguro de los datos.

# Límites del Proyecto

# No se incluirán módulos para gestión financiera ni pagos en esta etapa inicial.

# El sistema estará disponible exclusivamente como una aplicación web, optimizada para navegadores en diferentes dispositivos, pero sin una versión nativa para dispositivos móviles.

# **Capítulo IV: Análisis del Proyecto**

# 4.1 Requerimientos del Sistema

# Requerimientos Funcionales (Sintetizados)

# Registro y autenticación: Firebase Authentication gestionará accesos para administradores, apoderados y estudiantes, asegurando un 100% de accesos protegidos.

# Gestión de datos: Firestore y MongoDB Atlas manejarán datos de 700 estudiantes y apoderados en tiempo real, con margen de error menor al 1%.

# Reportes automáticos: Generación de reportes en menos de 5 segundos, sobre matrículas, cupos y estadísticas clave, accesibles desde el navegador.

# Notificaciones: Envío de actualizaciones en tiempo real sobre matrículas y fechas importantes a apoderados y estudiantes.

# Supervisión: Firebase Analytics rastreará métricas clave con 95% de precisión.

# Requerimientos No Funcionales

# Accesibilidad: Compatible con Chrome 92+, Firefox 90+, Edge 91+ y Safari 14+, optimizado para resoluciones entre 1024x768 y 1920x1080.

# Rendimiento: Respuesta menor a 2 segundos y disponibilidad de 99.9% (uptime).

# Seguridad: Reglas avanzadas de Firebase Security Rules, cifrado HTTPS y validación para prevenir vulnerabilidades.

# Escalabilidad: Inicia con 700 usuarios, soportando un crecimiento anual del 20% sin afectar rendimiento.

# Mantenimiento: Revisiones trimestrales y soporte técnico durante los primeros 90 días.

# 4.2 Análisis de Factibilidad

# Factibilidad Técnica

# El sistema combinará Firebase y MongoDB Atlas para garantizar una implementación segura, escalable y eficiente:

# Firebase Authentication (10.3.1): Gestionará el registro y autenticación de usuarios, asegurando un 100% de accesos verificados.

# Firestore (9.23.0): Administrará datos en tiempo real, con tiempos de respuesta menores a 2 segundos para operaciones básicas.

# MongoDB Atlas: Manejará datos estructurados y de mayor complejidad, optimizando consultas mediante índices y ofreciendo flexibilidad para gestionar un volumen creciente de información.

# Firebase Hosting (10.3.0): Proporcionará un entorno seguro y rápido para la aplicación web.

# Firebase Security Rules y MongoDB Atlas Security: Configurarán reglas avanzadas para proteger datos sensibles y restringir accesos según roles definidos.

# **Entorno de Programación**

# El desarrollo del sistema se realizará en Visual Studio Code (1.81 o superior), empleando herramientas y extensiones como:

# Firebase Tools: Para gestionar proyectos y configuraciones de Firebase.

# MongoDB Extension: Para administrar colecciones y realizar pruebas directamente desde el entorno.

# Prettier y ESLint: Asegurando un código limpio y validado.

# Live Server: Facilitando la previsualización en tiempo real.

# Navegadores Compatibles

# El sistema será accesible desde:

# Google Chrome (92+).

# Mozilla Firefox (90+).

# Microsoft Edge (91+).

# **Configuraciones del Servidor**

# Mínima: Conexión estable a internet de 10 Mbps para pruebas locales.

# Óptima: Equipo con procesador Intel i5, RAM de 8 GB y SSD de 256 GB, complementado con la capacidad escalable de Firebase Hosting y MongoDB Atlas.

# **Seguridad**

# Cifrado HTTPS: Todas las comunicaciones estarán protegidas contra intercepciones.

# Reglas de Seguridad Avanzadas: Firebase Security Rules y MongoDB Atlas configurarán permisos según roles y protegerán datos sensibles.

# Protección contra Vulnerabilidades: Validación de entradas para evitar inyecciones SQL y autenticación segura para prevenir ataques de fuerza bruta.

# Auditorías Periódicas: Identificarán vulnerabilidades mediante herramientas como OWASP ZAP, reforzando la seguridad.

# Escalabilidad

# Crecimiento Anual: Capacidad para gestionar un aumento del 20% anual en registros y operaciones.

# Optimización de Base de Datos: Firestore manejará datos en tiempo real, mientras MongoDB Atlas administrará datos estructurados para consultas más complejas.

# Escalabilidad Automática: Firebase Hosting y MongoDB Atlas ajustarán recursos dinámicamente para soportar mayores volúmenes de tráfico y datos.

# Rendimiento Consistente: Asegurará tiempos de respuesta menores a 2 segundos, incluso con 100 usuarios concurrentes.

# 4.2.2. Factibilidad Operativa:

# El sistema atenderá a administradores (5-10 por colegio), 700 apoderados y 700 estudiantes, adaptándose a sus necesidades mediante paneles personalizados y acceso intuitivo. La implementación se desarrollará en fases:

# Fase Piloto: Prueba en una institución durante 30 días con 50 estudiantes y apoderados para ajustar configuraciones.

# Capacitación: Talleres virtuales de 4 horas para administradores, con manuales y tutoriales para apoderados y estudiantes.

# Transición Completa: Implementación en todos los colegios tras la fase piloto, con soporte técnico durante los primeros 90 días.

# Un equipo de soporte de 2 especialistas atenderá consultas y realizará mantenimiento preventivo trimestral, garantizando la continuidad operativa del sistema.

# **4.3 Planificación Preliminar**

# Se desarrollará una Carta Gantt que detalle las etapas principales del proyecto:

# Análisis de requerimientos.

# Diseño del sistema.

# Desarrollo del software.

# Pruebas y validación.

# Implementación y capacitación.

# **4.3 Planificación del Proyecto**

# 4.3.1 Cronograma de Actividades

# Ventajas de la Carta Gantt:

# Visualiza todas las tareas del proyecto y sus plazos.

# Identifica dependencias entre actividades (una tarea que debe completarse antes de comenzar otra).

# Permite realizar un seguimiento del progreso.

# A continuación, se muestra la Carta Gantt:

# 

**Capítulo V: Diseño de Sistemas y Modelado**

**5.1.1 Selección de Base de Datos**

Para este proyecto, se eligieron Firebase Firestore y MongoDB Atlas por su capacidad para satisfacer las necesidades del sistema. Firestore, una base de datos NoSQL en tiempo real, permite la sincronización instantánea de datos y se integra fácilmente con otros servicios de Firebase, optimizando la autenticación y la gestión de usuarios. Por su parte, MongoDB Atlas actúa como repositorio principal, almacenando información histórica y administrativa gracias a su escalabilidad, alta disponibilidad y compatibilidad con datos semiestructurados. Esta combinación permite una arquitectura híbrida que equilibra operaciones en tiempo real y almacenamiento robusto a largo plazo.

**5.1.2 Justificación de la Elección entre SQL o NoSQL para el Proyecto**

La elección de bases de datos NoSQL se fundamenta en la flexibilidad, escalabilidad y sincronización en tiempo real necesarias para este sistema. Sus estructuras dinámicas permiten gestionar datos jerárquicos como estudiantes, apoderados y matrículas sin esquemas rígidos, facilitando futuras adaptaciones.

Firestore ofrece sincronización en tiempo real, esencial para interfaces interactivas, mientras que MongoDB Atlas garantiza un almacenamiento eficiente y escalable. Ambas bases de datos operan con JSON, simplificando la integración con React.js y Node.js, y reduciendo costos y complejidad frente a soluciones SQL tradicionales. En conjunto, estas características aseguran un sistema ágil y preparado para futuros retos.

# 5.2 Justificación de la Elección del Modelo

# La elección de un modelo No Relacional orientado a documentos para este proyecto se fundamenta en los siguientes puntos:

# Características del Proyecto:

# El sistema está diseñado para manejar datos relacionados con usuarios, apoderados, estudiantes y matrículas.

# Se requiere almacenar datos heterogéneos, como información general, referencias entre entidades y listas anidadas (por ejemplo, matrículas de estudiantes).

# Algunas entidades (como usuarios o estudiantes) necesitan relaciones directas o embebidas, que son comunes en bases de datos orientadas a documentos.

# Tecnologías Utilizadas:

# MongoDB Atlas: Una base de datos NoSQL basada en documentos que soporta consultas avanzadas, validaciones, y escalabilidad horizontal.

# Firebase: Su Firestore, también orientado a documentos, permite sincronización en tiempo real y es ideal para sistemas distribuidos.

# Estas tecnologías complementan el modelo NoSQL.

# **Ventajas del Modelo No Relacional:**

# Eficiencia en las consultas: La capacidad de embebido permite almacenar datos relacionados (por ejemplo, datos de apoderados y estudiantes) dentro de un único documento, reduciendo la cantidad de consultas necesarias.

# Flexibilidad: Los esquemas dinámicos permiten agregar nuevos atributos o modificar estructuras sin afectar la operatividad del sistema.

# Escalabilidad horizontal: MongoDB y Firebase están diseñados para manejar grandes volúmenes de datos distribuidos en múltiples nodos.

# **Compatibilidad con las Herramientas:**

# MongoDB Atlas soporta JSON como formato de datos, lo que simplifica la interacción con el backend y frontend.

# Firebase Firestore proporciona capacidades en tiempo real para sincronización de datos clave, como matrículas activas o estados del sistema.

# 5.1.3 Breve explicación de cómo se utilizará la base de datos seleccionada.

El sistema utiliza una arquitectura híbrida que combina Firebase Firestore y MongoDB Atlas para optimizar el manejo de datos en tiempo real y almacenamiento centralizado.

* **Firebase Firestore** se emplea para la **gestión inmediata de datos**, como el registro y actualización de estudiantes, apoderados y matrículas, sincronizando cambios en tiempo real con la interfaz web. Esto garantiza que los usuarios vean información actualizada de manera instantánea, mejorando la experiencia interactiva del sistema.
* **MongoDB Atlas** se utiliza como **repositorio principal y administrativo**, almacenando información consolidada y registros históricos para consultas más extensas y generación de reportes. También actúa como respaldo para la integridad y consistencia de los datos a largo plazo.

La integración entre ambas plataformas se automatiza mediante scripts de sincronización que garantizan que los datos críticos sean compartidos eficientemente entre Firestore y MongoDB. Esto asegura que la información esté siempre disponible y actualizada, tanto para operaciones en tiempo real como para análisis y reportes administrativos.

# 5.3 Modelado de base de datos: Diagrama Entidad-Relación (ER)

# El modelo ER para este proyecto incluirá las siguientes entidades clave:

# Estudiantes: id,nombre,curso,estado de matrícula, apoderado y matricula.

# Apoderados: id,nombre,teléfono,dirección y estudiantes.

# Matrículas: id,fecha de matrícula, estado de matrícula, año académico, estudiante asociado y usuario asociado.

# Usuarios: id, nombre, email, rol, activo (sí o no), fecha de creación, apoderado y estudiante.

# Imagen Modelo No relacional

# Realizada con APP Master

# En la figura anterior, el diagrama organiza estudiantes, apoderados, usuarios y matrículas, optimizando las relaciones mediante embebidos. En la colección estudiantes, los datos de apoderados (nombre, teléfono, dirección) se embeben para simplificar consultas. Las matrículas se incluyen como subdocumentos dentro de los estudiantes, con atributos como estado y año académico. En la colección matrículas, se embeben datos básicos de los usuarios responsables (rol y nombre), facilitando la gestión y reduciendo relaciones externas. Este enfoque mejora la eficiencia y claridad en un entorno NoSQL como MongoDB.

# **5.3 Casos de Uso**

# Caso de Uso 1: Registro y Consulta de Estudiantes

# Objetivo: Registrar un estudiante nuevo en el sistema y consultar su información académica.

# Descripción:

# El administrador ingresa los datos del estudiante, como nombre, edad, curso, y estado de matrícula. El sistema verifica si el apoderado está registrado; si no, permite registrar su información básica. El sistema asocia al estudiante con el apoderado y guarda los datos en la base de datos. Posteriormente, el administrador consulta la información del estudiante ingresando su nombre, y el sistema muestra los datos registrados.

# Actor(es): Administrador.

# Precondiciones: El estudiante y su apoderado no deben estar registrados previamente.

# Flujo Alternativo: Si el apoderado ya está registrado, se omite el registro y se asocia al estudiante directamente.

# Caso de Uso 2: Visualización de Información de Apoderados y Estudiantes Asociados

# Objetivo: Consultar los datos de un apoderado junto con la lista de estudiantes asociados.

# Descripción:

# El apoderado inicia sesión en el sistema y accede a su panel personal.El sistema muestra los datos del apoderado, como nombre, teléfono y dirección.Se despliega la lista de estudiantes asociados con su curso y estado de matrícula.El apoderado puede regresar al panel de información del estudiante para más detalles.

# Actor(es): Apoderado.

# Precondiciones: El apoderado y los estudiantes asociados deben estar registrados.

# Flujo Alternativo: Si no hay estudiantes asociados, el sistema informa al apoderado.

# Caso de Uso 3: Filtrado y Actualización de Matrículas

# Objetivo: Filtrar estudiantes y actualizar su estado de matrícula en el sistema.

# Descripción:

# El administrador utiliza el formulario de búsqueda para filtrar estudiantes por nombre, edad, curso o estado de matrícula. El sistema muestra los resultados del filtro con los detalles del estudiante. El administrador selecciona al estudiante y actualiza el estado de su matrícula (por ejemplo, de "Pendiente" a "Activa").

# El sistema guarda los cambios y sincroniza la información en tiempo real.

# Actor(es): Administrador.

# Precondiciones: El estudiante debe estar registrado previamente con al menos una matrícula asociada.

# Flujo Alternativo: Si no se encuentran estudiantes que cumplan los criterios del filtro, el sistema informa al administrador.

# 5.3.2 Diagrama de casos de uso con su notación UML.

# 5.3.3 Descripción de las relaciones clave entre los casos de uso.

# 5.4 Diseño de Árbol Funcional e Interfaz de Usuario

# El árbol funcional es una representación jerárquica de los módulos y funciones del sistema. Esto ayudará a visualizar cómo los usuarios navegan entre las diferentes partes de la aplicación.

# 5.4.1 Árbol Funcional de Navegación

# A continuación, se muestra una imagen del diseño del Árbol.

# 

# 5.4.2 Estructura de navegación del sistema.

# A continuación, se muestra la estructura del Árbol Funcional:

# Árbol Funcional Simplificado

# **1. Gestión de Usuarios**

# Registro y autenticación mediante Firebase.

# Gestión de perfiles y roles (Administrador, Apoderado, etc.).

# Envío de notificaciones push (Firebase Cloud Messaging).

# **2. Gestión de Apoderados**

# Registro y actualización de datos de contacto.

# Asociación de apoderados con estudiantes.

# **3. Gestión de Estudiantes**

# Registro de estudiantes y asignación a apoderados.

# Actualización de curso y estado de matrícula.

# Visualización de historial académico.

# **4. Gestión de Matrículas**

# Creación y edición de matrículas.

# Actualización de estados (activa, pendiente, finalizada).

# Reportes por año académico.

# 5.4.3 Breve explicación de cómo facilita la usabilidad.

# La navegación del sistema está diseñada para ser intuitiva y eficiente, organizando las funciones en módulos claros según los roles de usuario. Cada perfil, como Administrador o Apoderado, accede solo a las herramientas necesarias, reduciendo la complejidad.

# El uso de elementos visuales consistentes, como botones y formularios, asegura que los usuarios se familiaricen rápidamente con el sistema. Además, las tareas están estructuradas en un flujo lógico, permitiendo completar acciones como registrar estudiantes o actualizar matrículas sin complicaciones.

# La integración en tiempo real con Firebase y MongoDB Atlas garantiza sincronización inmediata de datos, mejorando la experiencia interactiva. Aunque no incluye una versión móvil, su compatibilidad con navegadores asegura un acceso eficiente desde cualquier dispositivo. Esto simplifica las operaciones y mejora la satisfacción del usuario.

# 5.5 Diseño de Prototipos

# 5.5.1 Representación gráfica de la interfaz de usuario.

# Ventana 1:Login

# 

# Ventana 2: DashBoard Administrador.

# 

# Ventana 3: Dashboard Estudiante

# 

# Ventana 4: Dashboard Apoderado

# 

# Ventana 5: Matrícua

# 

# 5.5.2 Breve descripción de las herramientas utilizadas y la funcionalidad principal.

# **1. Pantalla de Inicio de Sesión (Login)**

# Descripción:

# Página web donde los usuarios pueden autenticarse.

# Utiliza Firebase Authentication para manejar credenciales de acceso.

# Elementos:

# Formulario con:

# Campo de correo electrónico.

# Campo de contraseña.

# Botón "Iniciar Sesión".

# Enlace "¿Olvidaste tu contraseña?" que redirige a una función de recuperación.

# Flujo:

# El usuario ingresa sus credenciales.

# Firebase valida la autenticación.

# Si es exitoso, redirige al Dashboard.

# **2. Dashboard Principal**

# Descripción:

# Página principal después del inicio de sesión.

# Muestra un resumen de los datos del usuario (rol, notificaciones, últimas acciones).

# Elementos:

# Menú de navegación con opciones:

# Ver Matrículas.

# Gestión de Estudiantes.

# Gestión de Apoderados.

# 3. Gestión de Matrículas

# Pantalla: Listado de Matrículas

# Descripción:

# Muestra un listado de todas las matrículas registradas.

# Incluye opciones para buscar, filtrar, y acceder a detalles específicos.

# Elementos:

# Tabla con:

# Nombre del estudiante.

# Año académico.

# Estado de la matrícula.

# Botón "Ver Detalle".

# Acciones:

# Al hacer clic en "Ver Detalle", se abre la pantalla de detalles.

# Pantalla: Detalle de Matrícula

# Descripción:

# Muestra información detallada de una matrícula específica.

# Elementos:

# Campos con datos como:

# Nombre del estudiante.

# Curso.

# Apoderado asociado.

# Estado.

# Observaciones.

# Botones para:

# Actualizar matrícula.

# Eliminar matrícula.

# **4. Gestión de Estudiantes**

# Pantalla: Listado de Estudiantes

# Descripción:

# Lista de todos los estudiantes registrados en el sistema.

# Elementos:

# Tabla con:

# Nombre del estudiante.

# Curso actual.

# Estado de la matrícula.

# Botón "Editar" y "Eliminar".

# Pantalla: Registrar Estudiante

# Descripción:

# Formulario para registrar nuevos estudiantes.

# Elementos:

# Campos para:

# Nombre.

# Curso.

# ID del apoderado.

# Botón "Guardar".

# **5. Gestión de Apoderados**

# Pantalla: Listado de Apoderados

# Descripción:

# Lista de apoderados vinculados a estudiantes.

# Elementos:

# Tabla con:

# Nombre del apoderado.

# Contacto (teléfono o correo electrónico).

# Botón "Editar" y "Eliminar".

# Pantalla: Detalle del Apoderado

# Descripción:

# Información completa del apoderado y los estudiantes bajo su responsabilidad.

# Elementos:

# Campos con datos como:

# Nombre.

# Teléfono.

# Dirección.

# Lista de estudiantes asociados.

# 5.6 Metodología de desarrollo

# 5.6 Metodología de Desarrollo

# 5.6.1 Selección de Metodología de Desarrollo y Justificación

# Se selecciona la Metodología Cascada Tradicional por su enfoque secuencial, adecuado para el desarrollo estructurado del sistema propuesto. Esta metodología permite claridad en las etapas, documentación robusta y minimización de riesgos en un sistema educativo con requerimientos críticos.

# Justificación:

# Definición clara de etapas: Permite dividir el proyecto en fases específicas (análisis, diseño, desarrollo, pruebas e implementación).

# Documentación completa: Asegura trazabilidad y facilita el mantenimiento.

# Tecnologías alineadas: Compatible con Firebase, MongoDB Atlas y React, integrando componentes de forma progresiva.

# Control de calidad: Facilita pruebas y validaciones sistemáticas para evitar errores.

# Fases Aplicadas al Proyecto:

# Análisis: Identificación de requerimientos y necesidades educativas.

# Diseño: Modelado de datos NoSQL y creación de interfaces.

# Desarrollo: Implementación del backend y conexión con bases de datos.

# Pruebas: Validación funcional y de integración.

# Implementación: Despliegue en producción con Firebase Hosting.

# Este enfoque garantiza un sistema funcional y alineado con los objetivos del proyecto.

# **VI Modelado de datos.**

# Capítulo VI: modelado de datos

# 6.1 Fundamento y elección del modelo de datos

# El modelo de datos seleccionado para este proyecto es no relacional, utilizando MongoDB Atlas como base de datos principal. La elección se fundamenta en la flexibilidad y escalabilidad que este modelo ofrece, permitiendo un manejo eficiente de datos jerárquicos y semiestructurados. Adicionalmente, se utilizará Firebase Firestore para la sincronización en tiempo real de datos clave y Firebase Authentication para la gestión segura de usuarios. El desarrollo del sistema se realizará en Visual Studio Code, integrando estas herramientas de manera modular.

# 6.2 Modelado de la base de datos

# El diseño de la base de datos sigue los principios de eficiencia, flexibilidad y escalabilidad, asegurando una integración fluida con el sistema.

# 1. Colecciones Principales

# Usuarios:

# Estructura: Incluye datos del usuario (administrador o apoderado) y sus estudiantes asociados.

# Propósito: Gestionar roles, acceso y autenticación mediante Firebase Authentication.

# Atributos clave: id, nombre, email, rol, activo, fecha\_creacion.

# Estudiantes:

# Estructura: Incluye datos básicos del estudiante y referencias a apoderados y matrículas.

# Propósito: Almacenar información académica y personal del estudiante.

# Atributos clave: id, nombre, curso, estado\_matricula, apoderado.

# Apoderados:

# Estructura: Contiene información personal del apoderado y una lista de estudiantes asociados.

# Propósito: Centralizar la relación entre apoderados y estudiantes.

# Atributos clave: id, nombre, telefono, direccion.

# Matrículas:

# Estructura: Registra información del proceso de matrícula y su estado.

# Propósito: Documentar inscripciones académicas, estado y responsables de la matrícula.

# Atributos clave: id, fecha\_matricula, estado\_matricula, ano\_academico, usuario, estudiante.

# 2. Estructura del Modelo

# El diseño incluye:

# Embebidos: Datos como apoderados dentro de estudiantes y usuarios dentro de matrículas para optimizar consultas frecuentes.

# Referencias: Relaciones mínimas entre colecciones para mantener la flexibilidad del modelo no relacional.

# 6.3. Diccionario de datos

# a) Colección: usuarios

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Cantidad | Descripción |
| id | String | 3 | Identificador único del usuario. |
| nombre\_usuario | String | 100 | Nombre del usuario. |
| email\_usuario | String | 150 | Correo electrónico del usuario. |
| rol\_usuario | String | 20 | Rol del usuario (Administrador, Apoderado, etc.). |
| activo | Boolean | 5 | Indica si el usuario está activo. |
| fecha\_creacion | Date | 24 | Fecha de creación del usuario. |
| apoderado | Object | 36 | Datos del apoderado (si corresponde). |
| estudiantes | Array | 36 | Lista de estudiantes asociados al usuario. |

# Ejemplo colección usuario:

{  
 "id": "usuario\_1",  
 "nombre\_usuario": "admin",  
 "email\_usuario": "[admin@example.com](mailto:admin@example.com)",  
 "rol\_usuario": "Administrador",  
 "activo": true,  
 "fecha\_creacion": "2025-01-09",  
 "apoderado": {  
 "id": "apoderado\_1",  
 "nombre": "María López",  
 "telefono": "123456789",  
 "direccion": "Calle 123, Ciudad"  
 },  
 "estudiantes": [  
 {  
 "id": "estudiante\_1",  
 "nombre": "Juan Pérez",  
 "curso": "8vo Básico",  
 "estado\_matricula": "Activa",  
 "matriculas": [  
 {  
 "id": "matricula\_1",  
 "fecha\_matricula": "2025-01-09",  
 "estado\_matricula": "Activa",  
 "ano\_academico": 2025  
 }  
 ]  
 }  
 ]  
}  
El objeto JSON representa la estructura de un usuario en el sistema, donde se incluyen datos básicos como identificador, nombre, correo, rol asignado, estado de actividad y fecha de creación. Además, el usuario puede estar vinculado a un apoderado, cuya información de contacto y dirección está especificada, y a uno o más estudiantes asociados. Cada estudiante cuenta con detalles como identificador, curso, estado de matrícula y una lista de matrículas con información relevante, incluyendo identificador, fecha, estado y año académico correspondiente.

# b) Colección: estudiante

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Cantidad | Descripción |
| id | String | 36 | Identificador único del estudiante. |
| nombre | String | 100 | Nombre completo del estudiante. |
| curso | String | 20 | Nivel educativo del estudiante. |
| estado\_matricula | String | 20 | Estado actual de la matrícula. |
| apoderado | Array |  | Este campo almacena un arreglo, no necesita una longitud fija. Cada objeto del arreglo puede contener los siguientes atributos: |
| matriculas | Array |  | Lista de matrículas del estudiante. |

# Ejemplo colección estudiante:

# { "id": "estudiante\_1", "nombre": "Juan Pérez", "curso": "8vo Básico", "estado\_matricula": "Activa", "apoderado": { "id": "apoderado\_1", "nombre": "María López", "telefono": "123456789", "direccion": "Calle 123, Ciudad" }, "matriculas": [ { "id": "matricula\_1", "fecha\_matricula": "2025-01-09", "estado\_matricula": "Activa", "ano\_academico": 2025 } ] } El uso de atributos como apoderado y matriculas refleja directamente el diseño no relacional y optimiza consultas al reducir las dependencias externas.

# c) Colección: apoderado

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Cantidad | Descripción |
| id | String | 36 | Identificador único del apoderado. |
| nombre | String | 100 | Nombre completo del apoderado. |
| telefono | String | 15 | Teléfono de contacto. |
| direccion | String | 150 | Dirección de residencia. |
| estudiantes | Array |  | Lista de estudiantes bajo su tutela. |

# La relación con el diccionario de datos en este caso apoya la estructuración clara de la información, asegurando que los datos de los apoderados se gestionen de manera eficiente y sean fácilmente accesibles en la base de datos no relacional (MongoDB Atlas).

# Ejemplo colección apoderado:

# { "id": "apoderado\_1", "nombre": "María López", "telefono": "123456789", "direccion": "Calle 123, Ciudad", "estudiantes": [ { "id": "estudiante\_1", "nombre": "Juan Pérez", "curso": "8vo Básico", "estado\_matricula": "Activa" } ] }

# La estructura asegura que cada entidad se alinee perfectamente con los requisitos funcionales y el modelo no relacional definido.

# d) Colección: matricula

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Cantidad | Descripción |
| id | String | 36 | Identificador único de la matrícula. |
| fecha\_matricula | Date | 24 | Fecha de inscripción del estudiante. |
| estado\_matricula | String | 20 | Estado actual de la matrícula. |
| ano\_academico | Number | 4 | Año académico de la matrícula. |
| estudiante | Array |  | Datos del estudiante asociado. |
| usuario | Array |  | Usuario que gestionó la matrícula. |

# La estructura asegura que cada entidad se alinee perfectamente con los requisitos funcionales y el modelo no relacional definido.

# Ejemplo colección matricula:

# { "id": "matricula\_1", "fecha\_matricula": "2025-01-09", "estado\_matricula": "Activa", "ano\_academico": 2025, "estudiante": { "id": "estudiante\_1", "nombre": "Juan Pérez" }, "usuario": { "id": "usuario\_1", "nombre\_usuario": "admin" } }

# **Capítulo VII: Desarrollo e Implementación de Sistemas**

# 7.1 Arquitectura del Sistema

# La arquitectura del sistema de este proyecto se basa en una arquitectura cliente-servidor distribuida, utilizando tecnologías modernas que permiten escalabilidad, flexibilidad y eficiencia en la gestión de datos. Se incluirán los siguientes componentes principales organizados en aspectos como tecnología, interacción y base de datos.

# 7.2 Componentes de la Arquitectura

# Frontend:

# Tecnología: React.js.

# Interacción: El frontend se conecta con Firebase para autenticación y sincronización de datos en tiempo real, además de realizar solicitudes al backend para otras operaciones.

# Herramienta: Visual Studio Code, configurado con extensiones como ESLint y Prettier.

# Backend:Tecnología: Node.js con Express.js para manejar las operaciones del servidor y la lógica de negocio.

# Base de Datos: MongoDB Atlas como almacenamiento principal para los datos estructurados y semi-estructurados.

# Firebase Firestore: Para manejar sincronización de datos clave en tiempo real, como actualizaciones de matrículas activas o estados del sistema.

# Interacción: El backend se comunica con MongoDB Atlas para la persistencia de datos y con Firebase para operaciones en tiempo real.

# Base de Datos:

# MongoDB Atlas: Base de datos orientada a documentos para la persistencia de datos principales del sistema (usuarios, apoderados, estudiantes, matrículas).

# Las colecciones se basan en el modelo no relacional diseñado en las imágenes.

# Firebase Firestore:Base de datos orientada a documentos en tiempo real para sincronización de datos entre clientes y sistema.

# Uso para mantener actualizados los datos de estados, matrículas activas y notificaciones.

# Servicios de Nube:

# Firebase Authentication:Autenticación y gestión de usuarios para acceso seguro al sistema.

# Hosting de Firebase o Netlify: Para desplegar el frontend React.js.

# 7.3 Diagrama de Arquitectura

# El sistema se compone de los siguientes elementos: El usuario interactúa con el frontend React.js. El frontend realiza solicitudes al backend (Node.js). El backend se comunica con MongoDB Atlas y Firebase Firestore.

# Los datos son sincronizados en tiempo real entre Firebase y los usuarios.

# Flujo de Datos

# Autenticación: El usuario se autentica mediante Firebase Authentication.

# Consulta de Datos: El frontend solicita al backend información de usuarios, apoderados y estudiantes almacenados en MongoDB Atlas.

# Sincronización en Tiempo Real: Las actualizaciones de matrículas activas o estados se sincronizan automáticamente mediante Firebase Firestore.

# Persistencia de Datos: Los cambios realizados se almacenan en MongoDB Atlas para garantizar consistencia y seguridad.

# 7.2 Implementación del Sistema Basada en la Arquitectura Definida

# 1. Preparación del Entorno de Desarrollo

# IDE: Visual Studio Code con extensiones : Prettier

# Firebase Tools: Para gestionar Firebase desde la línea de comandos.

# Dependencias Principales:

# Express: Framework para construir el backend.

# Firebase-admin: SDK para integrar Firebase en el backend.

# Axios: Para realizar solicitudes HTTP desde el frontend.

# 2. Implementación del Backend

# Configuración de MongoDB Atlas:

# Creación de colecciones según el modelo definido (usuarios, apoderado, estudiante, matricula).

# Definición de esquemas con Mongoose, incluyendo validaciones y restricciones. En su defecto, si no se pudiera implementar Mongoose entonces utilizaría la manera manual en react.

# Integración con Firebase:

# Uso de Firebase Admin SDK para la gestión de usuarios y autenticación.

# 3. Implementación del Frontend

# Conexión con Firebase Firestore:

# Manejo de datos en tiempo real para matrículas activas.

# Componentes de React:

# Componentes modulares como Usuario, Apoderado, Estudiante y Matricula.

# Uso de Axios para interactuar con el backend y Firestore.

# 4. Resultados esperados.

# Sistema funcional con integración entre MongoDB Atlas, Firebase y React.

# Persistencia y sincronización de datos en tiempo real.

# Arquitectura escalable y modular.

# **Capítulo VIII Plan de Prueba**

# 8.1 Formato preliminar al documento

# 8.1.1 Introducción plan de prueba

# El plan de prueba establece las directrices para verificar y validar el correcto funcionamiento del sistema de gestión de matrículas desarrollado. Este documento detalla las actividades, casos de prueba, herramientas y criterios que aseguran que el sistema cumpla con los requerimientos funcionales y no funcionales establecidos.

# 8.1.2 Objetivo general plan de prueba

# Garantizar que el sistema funcione de acuerdo con las especificaciones del proyecto, identificando y corrigiendo defectos antes de su implementación final, con énfasis en la experiencia del usuario, la integridad de los datos y la seguridad

# 8.1.3 Alcance

# Las pruebas abarcarán los módulos principales del sistema: autenticación de usuarios, gestión de estudiantes y apoderados, generación de reportes, sincronización en tiempo real y persistencia de datos en MongoDB Atlas. Quedan excluidos módulos financieros y funcionalidades fuera del alcance definido para esta etapa.

# 8.1.4 Propósito

# Asegurar la calidad, funcionalidad, y confiabilidad del sistema mediante un proceso estructurado de pruebas, minimizando riesgos y asegurando que la solución cumpla con los estándares requeridos para su implementación en entornos educativos.

# 8.2 Plan de pruebas

# 8.2.1 Características que serán probadas

# Las pruebas se enfocarán en garantizar la funcionalidad, seguridad y rendimiento del sistema en los siguientes aspectos clave:

# Autenticación de Usuarios: Validación de accesos mediante Firebase Authentication, garantizando seguridad y confiabilidad.

# Gestión de Datos: Operaciones de lectura, escritura y sincronización en tiempo real entre MongoDB Atlas y Firestore.

# Generación de Reportes: Validación de la funcionalidad para producir estadísticas y reportes administrativos en el panel web.

# Interacción Frontend-Backend: Pruebas de comunicación y consistencia entre el frontend desarrollado en React.js y el backend en Node.js.

# Rendimiento: Evaluación del sistema con hasta 100 usuarios concurrentes, midiendo tiempos de respuesta en operaciones críticas.

# 8.2.2 Características que no serán probadas

# Gestión Financiera: No se incluirán pruebas de módulos relacionados con pagos, ya que están fuera del alcance del proyecto.

# Aplicaciones Móviles: Las pruebas se limitarán a la versión web y no contemplan funcionalidades móviles.

# Escalabilidad Extrema: Aunque se evaluará el rendimiento con usuarios concurrentes, no se probará bajo escenarios de carga extrema o infraestructuras más allá de los requisitos iniciales.

# 8.2.3 Criterios de aprobación y fallo

# Aprobación:

# Autenticación realizada con éxito para el 100% de los usuarios registrados.

# Sincronización en tiempo real con una tasa de error inferior al 1%.

# Generación de reportes en menos de 2 segundos.

# Tiempo de respuesta promedio inferior a 2 segundos bajo condiciones normales.

# Fallo:

# Errores en más del 5% de las operaciones de autenticación.

# Inconsistencias en datos sincronizados o guardados en más del 3% de las pruebas realizadas.

# Tiempo de respuesta mayor a 2 segundos en más del 10% de las solicitudes.

# 8.2.4 Criterios de suspensión y reanudación

# Suspensión:

# Fallo crítico en la funcionalidad de autenticación o pérdida de integridad de datos en Firestore o MongoDB Atlas.

# Inhabilidad del sistema para manejar operaciones básicas en condiciones normales.

# Reanudación:

# Corrección de los fallos detectados en la etapa suspendida.

# Validación exitosa de pruebas unitarias y de integración en los módulos afectados.

# 8.2 Plan de Pruebas

# 8.2.5 Casos de Prueba y Tareas: Pruebas Manuales

# 8.2 Plan de pruebas

# 8.2.1 Características que serán probadas

# Las pruebas se enfocarán en garantizar la funcionalidad, seguridad y rendimiento del sistema en los siguientes aspectos clave: - Autenticación de Usuarios: Validación de accesos mediante Firebase Authentication, garantizando seguridad y confiabilidad. - Gestión de Datos: Operaciones de lectura, escritura y sincronización en tiempo real entre MongoDB Atlas y Firestore. - Generación de Reportes: Validación de la funcionalidad para producir estadísticas y reportes administrativos en el panel web. - Interacción Frontend-Backend: Pruebas de comunicación y consistencia entre el frontend desarrollado en React.js y el backend en Node.js. - Rendimiento: Evaluación del sistema con hasta 100 usuarios concurrentes, midiendo tiempos de respuesta en operaciones críticas.

# 8.2.2 Características que no serán probadas

# - Gestión Financiera: No se incluirán pruebas de módulos relacionados con pagos, ya que están fuera del alcance del proyecto. - Aplicaciones Móviles: Las pruebas se limitarán a la versión web y no contemplan funcionalidades móviles. - Escalabilidad Extrema: Aunque se evaluará el rendimiento con usuarios concurrentes, no se probará bajo escenarios de carga extrema o infraestructuras más allá de los requisitos iniciales.

# 8.2.3 Criterios de aprobación y fallo

# Aprobación: - Autenticación realizada con éxito para el 100% de los usuarios registrados. - Sincronización en tiempo real con una tasa de error inferior al 1%. - Generación de reportes en menos de 2 segundos. - Tiempo de respuesta promedio inferior a 2 segundos bajo condiciones normales. Fallo: - Errores en más del 5% de las operaciones de autenticación. - Inconsistencias en datos sincronizados o guardados en más del 3% de las pruebas realizadas. - Tiempo de respuesta mayor a 2 segundos en más del 10% de las solicitudes.

# 8.2.4 Criterios de suspensión y reanudación

# Suspensión: - Fallo crítico en la funcionalidad de autenticación o pérdida de integridad de datos en Firestore o MongoDB Atlas. - Inhabilidad del sistema para manejar operaciones básicas en condiciones normales. Reanudación: - Corrección de los fallos detectados en la etapa suspendida. - Validación exitosa de pruebas unitarias y de integración en los módulos afectados.

# 8.2.5 Casos de prueba y tareas, pruebas manuales

# 1. Prueba de autenticación con credenciales válidas. 2. Prueba de autenticación con credenciales incorrectas. 3. Registro de nuevos usuarios con validaciones en tiempo real. 4. Prueba de búsqueda de estudiantes por filtros específicos. 5. Prueba de sincronización de datos entre Firestore y MongoDB. 6. Validación de los datos ingresados en el panel de administración. 7. Generación de reportes desde el panel de estudiante. 8. Navegación entre paneles (estudiante, apoderado, matrícula). 9. Prueba de mensajes de error en caso de datos incompletos. 10. Evaluación del tiempo de respuesta en operaciones críticas.

# 8.2.6 Evidencias ejecución

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **Capítulo IX: Análisis de Resultados y Verificación de Objetivos**

# 9.1 Análisis de Resultados

# Las pruebas realizadas permitieron validar los aspectos clave del sistema, garantizando su funcionalidad, seguridad y rendimiento de acuerdo con los objetivos planteados. Los resultados obtenidos en cada característica probada fueron los siguientes:

# Autenticación de Usuarios:

# El sistema permitió realizar autenticaciones exitosas para todos los usuarios registrados.

# Los mensajes de error asociados a contraseñas incorrectas o perfiles no coincidentes se mostraron correctamente, garantizando una experiencia de usuario clara y precisa.

# Gestión de Datos:

# Las operaciones de escritura y lectura desde Firestore y MongoDB Atlas fueron completadas sin inconsistencias.

# Se verificó la sincronización en tiempo real de datos entre ambos sistemas, mostrando un margen de error inferior al 1%.

# Interacción Frontend-Backend:

# Las pruebas de comunicación entre React.js y el backend fueron exitosas, asegurando que las solicitudes HTTP enviadas por Axios se resolvieron correctamente.

# Las operaciones de búsqueda y filtrado de datos en el panel administrativo fueron precisas, y se observó una adecuada separación entre consultas específicas y la visualización de resultados.

# Generación de Reportes:

# Aunque no se incluyó una funcionalidad de reportes avanzada, los datos de los estudiantes y apoderados se presentan de forma estructurada y clara en las interfaces, cumpliendo con los requisitos iniciales del proyecto.

# Rendimiento:

# El sistema mantuvo tiempos de respuesta promedio inferiores a 2 segundos en las pruebas realizadas bajo condiciones normales de operación con hasta 10 usuarios concurrentes, satisfaciendo las expectativas del proyecto.

# Incidencias detectadas:

# Algunos mensajes de error de autenticación inicialmente no se mostraron correctamente, pero fueron solucionados con ajustes al manejo de errores en la lógica de React.js.

# En la primera iteración, los estilos CSS no se aplicaron correctamente en algunos componentes, pero esto se resolvió al unificar las clases de estilo.

# 9.2 Verificación de Objetivos

# Los objetivos establecidos al inicio del proyecto fueron verificados mediante las pruebas descritas en el capítulo anterior. A continuación, se presenta el estado de cada objetivo:

# Desarrollo de un sistema funcional para la gestión de usuarios y datos en una institución educativa:

# Logrado. Se implementó un sistema que permite gestionar estudiantes, apoderados y estados de matrícula de manera eficiente.

# Integración entre Firestore y MongoDB Atlas:

# Logrado. Las operaciones de lectura y escritura en ambas bases de datos fueron exitosas, y se verificó la consistencia de datos entre ellas.

# Desarrollo de un sistema de autenticación confiable:

# Logrado. El sistema de autenticación basado en Firebase demostró ser seguro, con validación adecuada de perfiles y credenciales.

# Cumplimiento de los tiempos de respuesta definidos:

# Logrado. El sistema mantuvo tiempos de respuesta inferiores a 2 segundos bajo condiciones normales de operación.

# Garantizar una experiencia de usuario intuitiva:

# Logrado. Las interfaces desarrolladas en React.js ofrecen una navegación clara y funcional entre las diferentes vistas del sistema.

# En conclusión, el sistema desarrollado cumplió con todos los objetivos planteados, validando su funcionalidad, rendimiento y usabilidad mediante pruebas exhaustivas y correcciones iterativas.

# En el análisis de resultados, también se identificaron algunos aspectos que no se lograron o quedaron fuera del alcance del proyecto debido a limitaciones de tiempo o decisiones de priorización.

# Estos son:

# **Gestión Financiera:**

# Tal como se indicó en el plan de pruebas, los módulos relacionados con pagos y gestión financiera no fueron desarrollados ni probados, ya que estaban fuera del alcance inicial del proyecto.

# **Pruebas con alta concurrencia (Escalabilidad Extrema):**

# Aunque se evaluó el rendimiento con hasta 10 usuarios concurrentes, no se realizaron pruebas con una carga mayor (por ejemplo, 100 usuarios concurrentes o más), lo que limita la evaluación de la escalabilidad en escenarios más exigentes.

# **Generación de reportes administrativos avanzados:**

# Si bien se logró estructurar la información de estudiantes y apoderados en la interfaz, la generación de reportes automatizados y gráficos quedó pendiente para una fase futura.

# **Versiones móviles:**

# La aplicación no fue optimizada para dispositivos móviles, y todas las pruebas se realizaron exclusivamente en la versión web, lo cual puede limitar el acceso en plataformas diferentes.

# **Pruebas bajo condiciones de estrés extremo:**

# No se ejecutaron pruebas que simulen picos de carga significativos o fallos de red prolongados, dejando una evaluación incompleta del comportamiento del sistema en condiciones adversas.

# Conclusión sobre lo no logrado: A pesar de estos aspectos, el núcleo del proyecto, centrado en la gestión de usuarios, datos y autenticación, fue exitosamente implementado y probado. Las áreas no logradas representan oportunidades de mejora y expansión para futuras iteraciones o versiones del sistema.

# **Capítulo X. Conclusión y Reflexión:**

Este proyecto representó un avance significativo hacia la modernización de la gestión educativa, sentando las bases para una solución integral y escalable. Se logró implementar una integración eficiente entre Firebase y MongoDB Atlas, permitiendo la sincronización y automatización de datos, con vistas personalizadas según roles de usuario, lo que podría traducirse en una mejora sustancial en la experiencia y eficiencia operativa. Aunque se alcanzaron hitos importantes como la autenticación segura y la automatización de procesos clave, áreas como la gestión financiera, pruebas de carga extrema y aplicaciones móviles quedaron como desafíos futuros, ofreciendo un potencial claro para ampliar los alcances del sistema hacia una herramienta más robusta y universal en el ámbito educativo. Este trabajo evidencia la capacidad de adoptar tecnologías modernas y anticipa un impacto positivo si se expande a escenarios más complejos y de mayor alcance.

***Capítulo XI. Referencias***

*Firebase. (n.d.). Firebase Authentication Documentation. Recuperado de* [*https://firebase.google.com/docs/auth*](https://firebase.google.com/docs/auth)

*Firebase. (n.d.). Firestore Documentation. Recuperado de* [*https://firebase.google.com/docs/firestore*](https://firebase.google.com/docs/firestore)

*MongoDB. (n.d.). MongoDB Atlas Documentation. Recuperado de* [*https://www.mongodb.com/docs/atlas/*](https://www.mongodb.com/docs/atlas/)

*Node.js. (n.d.). Node.js Documentation. Recuperado de* [*https://nodejs.org/en/docs/*](https://nodejs.org/en/docs/)

*React.js. (n.d.). React Documentation. Recuperado de* [*https://reactjs.org/docs/getting-started.html*](https://reactjs.org/docs/getting-started.html)

*Visual Studio Code. (n.d.). Visual Studio Code Documentation. Recuperado de* [*https://code.visualstudio.com/docs*](https://code.visualstudio.com/docs)

*Whittaker, J. A., Arkin, M., & Thomason, M. (2019). Software Engineering Modern Practices. Boston, MA: Addison-Wesley.*

*AppMaster. (n.d.). AppMaster.io - No-code Platform for Backend, Web & Mobile Apps. Recuperado de* [*https://appmaster.io/*](https://appmaster.io/)

*Material Design. (n.d.). Material Design Guidelines. Recuperado de* [*https://material.io/design*](https://material.io/design)

*JustInMind. (n.d.). Prototyping Tool. Recuperado de* [*https://www.justinmind.com*](https://www.justinmind.com/)

*Axios. (n.d.). Promise-based HTTP client for the browser and Node.js. Recuperado de* [*https://axios-http.com/docs/intro*](https://axios-http.com/docs/intro)

*GitHub. (n.d.). GitHub Documentation: Repositories and version control best practices. Recuperado de* [*https://docs.github.com*](https://docs.github.com/)

*npm. (n.d.). Node Package Manager Documentation. Recuperado de* [*https://docs.npmjs.com*](https://docs.npmjs.com/)

*Google Cloud. (n.d.). Google Cloud Platform (GCP): Hosting Firebase and Firestore Integration. Recuperado de* [*https://cloud.google.com/docs*](https://cloud.google.com/docs)