ADMINISTRACIÓN
Y NEGOCIOS

Informe: Desarrollo del Sistema de Gestión de Préstamos Bibliotecarios

NOMBRE: Matias Pavez, Lucas Moyano
CARRERA: Ingeniería en informática
ASIGNATURA: Taller de desarrollo de soluciones
PROFESOR: Gustavo
FECHA:13-12-2024

Introducción

La gestión eficiente de los recursos bibliográficos es crucial para el éxito académico y organizacional de las instituciones educativas. Este informe documenta el desarrollo de un sistema de préstamos bibliotecarios implementado para la biblioteca del Colegio Pioneros. El sistema utiliza una arquitectura serverless basada en Firebase y una interfaz frontend en React, permitiendo la automatización y modernización de los procesos.

Propósito del proyecto

Digitalizar los préstamos y devoluciones de libros, optimizar la consulta de inventario y garantizar la seguridad de los datos mediante técnicas modernas de desarrollo de software. Además, busca facilitar el acceso a los usuarios finales mediante un sistema accesible y escalable.

Objetivos específicos

1. Implementar un sistema de autenticación que garantice la seguridad de los datos de los usuarios.
2. Desarrollar un flujo eficiente para el registro, consulta y seguimiento de préstamos bibliográficos.
3. Diseñar una interfaz gráfica moderna, intuitiva y accesible para diferentes tipos de usuarios.
4. Integrar un modelo de base de datos optimizado para consultas rápidas y precisas.
5. Implementar un sistema escalable que permita futuras integraciones y expansiones.

Actividades realizadas

Actividad 1: Construir las Interfaces Gráficas

Paso 1: Construcción de vistas según las necesidades del negocio

Se desarrollaron las siguientes vistas clave, adaptadas a las necesidades identificadas durante el análisis del negocio:

1. Inicio de Sesión:

- Autenticación utilizando Firebase Authentication con opciones para correo electrónico y contraseña.
- Implementación de un sistema de verificación de roles almacenados en Firestore.
- Redirección automática a las vistas correspondientes según el rol del usuario (administrador, bibliotecario o usuario).
- Validación de errores como contraseñas incorrectas o usuarios no registrados.

2. Panel del Bibliotecario:

- Registro de nuevos préstamos de libros mediante un formulario dinámico.

- Consulta de préstamos activos con opciones para filtrar por nombre de usuario, título del libro o fecha de préstamo.

- Funcionalidad para registrar devoluciones y marcar préstamos como completados.

3. Panel del Administrador:

- Gestión completa de usuarios con funciones para crear, editar, desactivar y eliminar usuarios.

- Gestión de inventario bibliográfico, permitiendo registrar nuevos libros, actualizar información existente y marcar libros como fuera de circulación.

Paso 2: Diseño moderno de las interfaces

El diseño de las interfaces sigue estándares de usabilidad y accesibilidad:

- Responsividad: Interfaces optimizadas para dispositivos móviles y de escritorio mediante CSS Grid y Flexbox.

- Interactividad: Incorporación de retroalimentación visual para formularios, botones y acciones críticas.

- Estética: Paleta de colores homogénea con tonos agradables para largas sesiones de uso, tipografía legible y separación clara entre secciones funcionales.

- Prototipado: Se realizaron prototipos en Figma para validar la experiencia del usuario antes de la implementación.

Actividad 2: Codificar la Solución

Paso 1: Implementación de requerimientos funcionales y no funcionales

La solución se desarrolló para cumplir con los siguientes requerimientos:

- Funcionales:

  - Gestión de préstamos y devoluciones.

  - Registro y consulta de usuarios y libros.

  - Control de accesos según roles definidos (administrador, bibliotecario, usuario).

- No funcionales:

  - Respuesta rápida del sistema incluso con datos en tiempo real.

  - Garantía de seguridad y privacidad de la información almacenada.

Paso 2: Codificación siguiendo el patrón de arquitectura

La solución se diseñó utilizando una arquitectura serverless, con los siguientes componentes clave:

- Frontend: React y herramientas modernas como Vite para un desarrollo rápido y eficiente.

- Backend:

  - Firebase Authentication: Autenticación segura.

  - Firestore: Base de datos no relacional para almacenar información estructurada.

  - Realtime Database: Sincronización en tiempo real de eventos críticos como actualizaciones de préstamos.

  - Cloud Storage: Almacenamiento seguro de documentos como copias digitales de manuales o información adicional.

Paso 3: Programación segura

Para garantizar la seguridad de la solución se implementaron:

- Validaciones: En el cliente y en el servidor para entradas de datos.

- Reglas de seguridad en Firebase: Configuradas para limitar el acceso a datos según los roles de usuario.

- Cifrado: Uso de conexiones HTTPS en todas las comunicaciones.

Actividad 3: Implementar la Base de Datos

Paso 1: Implementación de Firebase como base de datos no relacional

- Firestore: Diseñado para manejar colecciones como "usuarios", "libros" y "préstamos" con relaciones bien definidas entre ellas.

- Realtime Database: Utilizado para sincronización inmediata en operaciones críticas como actualizaciones de estado de préstamos.

Paso 2: Normalización de datos

Se evitó la redundancia y se optimizó el rendimiento con una estructura de datos organizada, garantizando consultas rápidas.

Paso 3: Procedimientos de extracción y despliegue

- Uso de índices compuestos en Firestore para consultas complejas.

- Reglas de seguridad detalladas para garantizar la integridad y privacidad de los datos.

Actividad 4: Configurar el Entorno que Soporta la Aplicación

Paso 1: Instalación y configuración del hardware necesario

El sistema no requiere infraestructura física adicional, funcionando directamente desde navegadores modernos. Se realizaron pruebas en diferentes dispositivos para validar la compatibilidad.

Paso 2: Instalación y configuración del software necesario

- Configuración de Firebase: Autenticación, Firestore, Realtime Database y Storage.

- Entorno de desarrollo con Node.js, React y herramientas adicionales como ESLint y Prettier para asegurar calidad de código.

Pruebas Realizadas

1. Pruebas Funcionales

Se validaron todas las funcionalidades principales:

- Inicio de sesión, registro de usuarios y gestión de préstamos.

- Flujo de trabajo para roles específicos como administrador y bibliotecario.

2. Pruebas de Integración

- Sincronización de datos entre el frontend y Firebase.

- Validación de integridad y consistencia de datos en operaciones simultáneas.

3. Pruebas de Usuario

- Sesiones de prueba con usuarios finales para validar la usabilidad y accesibilidad del sistema.

- Incorporación de retroalimentación para ajustar elementos críticos.

Documentación Generada

1. Manual de Usuario: Documentación paso a paso para utilizar el sistema, acompañada de capturas de pantalla.

2. Manual de Instalación: Instrucciones detalladas para configurar el entorno y desplegar la aplicación.

3. Código Fuente: Repositorio organizado con documentación interna en el código.

Plan de Mantenimiento

1. Correctivo: Resolución de errores reportados por usuarios o detectados en monitoreo.

2. Preventivo: Actualización regular de dependencias y revisión de reglas de seguridad.

3. Adaptativo: Incorporación de nuevas funcionalidades basadas en las necesidades de la institución.

4. Perfectivo: Mejoras en el diseño y rendimiento del sistema.

Entrega Final:

- Código fuente documentado. (En la carpeta de entrega)

-Manuales de Usuario y Manual de Instalacion

Manual de Usuario

Propósito: Guía paso a paso para que los usuarios finales puedan navegar y utilizar el sistema de manera eficiente.

1. **Inicio de Sesión**
    - Acceda al sistema desde la URL proporcionada.
    - Introduzca su correo electrónico y contraseña.
    - Según su rol, será redirigido al panel correspondiente:
        - *Admin*: Gestión de usuarios y libros.
        - *Bibliotecario*: Registro y consulta de préstamos.
        - *Usuario*: Verificación de préstamos activos.
        (Incluir captura de pantalla del formulario de inicio de sesión con instrucciones claras).
2. **Gestión de Libros** *(Rol: Admin)*
    - Navegue al menú "Gestión de Libros".
    - Use el botón "Añadir Nuevo Libro" para registrar títulos en el sistema.
    - Editar o eliminar registros utilizando los botones asociados a cada fila de la tabla.
    (Incluir captura de pantalla de la tabla de libros con las opciones).
3. **Registro de Préstamos** *(Rol: Bibliotecario)*
    - Seleccione un usuario en el menú desplegable de préstamos.
    - Escanee o introduzca manualmente el código del libro.
    - Confirme el préstamo presionando el botón "Registrar".
    (Incluir un diagrama simple mostrando el flujo de préstamo).

1. **Manual de Instalación**

Propósito: Instrucciones para configurar el sistema desde cero en un entorno nuevo.

2. **Requisitos Previos**
    - **Software:**
        - Node.js (v16 o superior).
        - Acceso a una cuenta de Firebase con permisos de administrador.
    - **Hardware:**
        - Navegador web moderno (Chrome, Firefox).
        - Conexión a Internet estable.
3. **Configuración Inicial**
    - Clone el repositorio del sistema desde GitHub:
    - git clone https://github.com/tu-repositorio.git
    - cd tu-repositorio
    - Instale las dependencias del proyecto:
    - npm install
4. **Conexión a Firebase**
    - Cree un nuevo proyecto en la consola de Firebase.
    - Configure Authentication, Firestore y Realtime Database.
    - Descargue el archivo google-services.json y colóquelo en la carpeta raíz del proyecto.
5. **Despliegue Local**
    - Ejecute el proyecto en un entorno de desarrollo:
    - npm run dev
    - Acceda al sistema en el navegador mediante http://localhost:3000.
6. **Despliegue en Producción** *(opcional)*
    - Configure Firebase Hosting para subir el frontend.
    - Use el comando:
    - firebase deploy

Script Base de Datos

1 Sección: Scripts para la Configuración Inicial

1. Reglas de Firebase Firestore

json

Copiar código

```
rules_version = '2';
service cloud.firestore {
 match /databases/{database}/documents {
  match /users/{userId} {
   allow read, write: if request.auth.uid == userId;
  }
  match /books/{bookId} {
   allow read: if true;
   allow write: if request.auth.token.role == 'admin';
  }
  match /loans/{loanId} {
   allow read, write: if request.auth.token.role in ['bibliotecario', 'admin'];
  }
 }
}
```

2. Script de Inicialización de Usuarios (Firestore)
   Este script crea usuarios iniciales con roles específicos:

javascript

Copiar código

```
const admin = require('firebase-admin');

admin.initializeApp();

const db = admin.firestore();


const users = [

 { email: 'admin@biblioteca.com', role: 'admin' },

 { email: 'biblio@biblioteca.com', role: 'bibliotecario' },

 { email: 'user@biblioteca.com', role: 'usuario' },

];


users.forEach(async (user) => {

 const userRef = db.collection('users').doc();

 await userRef.set(user);

 console.log(`Usuario ${user.email} creado con éxito`);

});
```

3. Script de Carga Masiva de Libros (Firestore)
   Para añadir libros iniciales:

javascript

Copiar código

```
const books = [

 { title: 'Cien años de soledad', author: 'Gabriel García Márquez', available: true },

 { title: 'Don Quijote de la Mancha', author: 'Miguel de Cervantes', available: true },

];


books.forEach(async (book) => {

 const bookRef = db.collection('books').doc();

 await bookRef.set(book);
```

```
  console.log(`Libro ${book.title} registrado con éxito`);
});
```