- **Portada**: debe aparecer el nombre del proyecto, el nombre del alumno, el ciclo, el curso y periodo (octubre/marzo, marzo/Junio), la denominación oficial del centro, el logo y el nombre del tutor individual.

- **Agradecimientos**: Opcional

- **Resumen**: El resumen debe de contar al menos un mínimo de 50 palabras y debe reflejar la idea principal desarrollada en el proyecto. Es obligatorio que el resumen esté en dos idiomas como mínimo, aunque es altamente recomendable que esté en tres: español, valenciano e inglés. Organizarlo en un par de párrafos. Debe estar diseñado para ser leído por alguien que no sabe nada del proyecto.

SustiHorario

Índice

1. Introducción

Introducción

SustiHorario es una aplicación móvil diseñada para la gestión de guardias de profesores en centros educativos. El proyecto surge para la educación secundaria y superior, donde las ausencias de profesores generan problemas en el horario lectivo. El objetivo principal es proporcionar una herramienta digital que automatice el reparto de guardias, garantice equidad en las asignaciones y facilite el acceso a información en tiempo real.

El problema

En muchos centros educativos, la gestión de sustituciones se realiza de manera manual o semimanual, utilizando hojas de cálculo, correos electrónicos o sistemas de papel. Esto conlleva varios problemas: falta de transparencia en el reparto de guardias, sobrecarga en ciertos profesores debido a criterios subjetivos, errores en la asignación por falta de visibilidad de horarios y disponibilidades, y retrasos en la comunicación de bajas o ausencias. Además, en entornos con múltiples roles, la coordinación se complica, lo que puede afectar la calidad educativa y el bienestar del personal.

Solución Actual

Actualmente, muchos centros dependen de herramientas como Microsoft Excel o Google Sheets para registrar horarios y ausencias, combinadas con comunicaciones vía email o WhatsApp para asignar sustituciones. En algunos casos, se utilizan plataformas educativas genéricas como Moodle o Google Classroom, pero estas no están especializadas en la gestión de guardias. Estas soluciones son ineficientes, propensas a errores humanos y no garantizan una distribución justa basada en criterios objetivos como la carga horaria o la disponibilidad declarada por los profesores.

Propuesta del Proyecto

La propuesta de SustiHorario es crear una app móvil para facilitar la gestión de guardias y sustituciones en los centros. Los profesores podrán consultar sus horarios, solicitar bajas y ver las guardias que se les han asignado. Los coordinadores, por su parte, podrán revisar las bajas registradas, crear modelos de horario y reasignar sustituciones (de forma manual o con apoyo automático) teniendo en cuenta la carga y la disponibilidad del profesorado. Además recibirán alertas ante casos de sobrecarga o falta de personal, podrán gestionar usuarios (verlos y eliminarlos) y establecer el límite de guardias. Los administradores únicamente tendrán la función de crear y configurar los centros. En conjunto, la aplicación usa algoritmos sencillos para distribuir las guardias de manera más justa, mejorando la eficiencia y la transparencia frente a los métodos tradicionales.

Reflexiones y algunas Dudas

Durante el proyecto se han planteado preguntas, como por ejemplo: ¿Cómo asegurar la igualdad en la asignación de sustituciones? ¿Cuáles son los mejores criterios para la distribución automática? ¿Cómo crear una interfaz accesible para usuarios con diferentes niveles tecnológicos? Estas se resolverán en fases futuras, pero el proyecto actual demuestra viabilidad básica.

2. Estado del arte

Para avanzar en el proyecto, analizaremos las soluciones en el mercado, las tecnologías y estrategias que usan. Con esta información podremos mejorar y definir un mejor proyecto.

Análisis de Trabajos Relacionados

En el ámbito de la gestión de sustituciones docentes, existen varias aplicaciones y sistemas diseñados para optimizar la asignación de profesores sustitutos en entornos educativos.

Entre las más destacadas están WebUntis y GHC.

- **WebUntis** es una plataforma online que permite a los centros educativos gestionar la planificación de sus horarios, la asignación de sustituciones y la comunicación en tiempo real entre profesores, alumnos y responsables. Su uso de aplicaciones web y móviles facilita el acceso desde diferentes dispositivos y mejora la interacción a través de avisos y notificaciones automáticas.
- **GHC (Generador de Horarios de Peñalara Software)** emplea un motor de cálculo avanzado que optimiza la creación de horarios complejos considerando múltiples variables como la disponibilidad de aulas, profesorado y preferencias del centro. Además, ofrece herramientas para la gestión diaria y el análisis del reparto de sustituciones.

SustiHorario, Se define por presentar información clara de la sustitución (hora, clase, materia y el sustituto) dándole importancia a la transparencia y la igualdad en la asignación de sustituciones. Busca disminuir el trabajo administrativo, delegando la responsabilidad automáticamente para evitar que recaiga únicamente en los coordinadores o responsables académicos o un documento con acceso a cualquier persona, y así facilita la colaboración del equipo completo.

Tecnologías y Lenguajes

La plataforma está desarrollada utilizando tecnologías modernas y eficientes, principalmente con Flutter y Dart para la creación de una aplicación móvil multiplataforma, que garantiza una experiencia fluida y responsiva tanto en Android como en iOS. Para el backend y almacenamiento de datos se utiliza Firebase, que ofrece una base de datos en tiempo real, autenticación segura. Esta combinación permite un desarrollo ágil, con sincronización en tiempo real y una gestión eficiente de usuarios y datos.

Estrategias y Métodos

Los métodos fundamentales en estas soluciones consisten en algoritmos automatizados de asignación que consideran parámetros personalizados, normas internas y criterios de igualdad. También se utilizan interfaces intuitivas para facilitar la consulta.

Novedades del Proyecto

SustiHorario se diferencia al priorizar una interfaz clara y accesible, enfocada en la experiencia del profesor y la información, con la posibilidad de consultar, añadir sustituciones de forma sencilla desde cualquier dispositivo.

Los coordinadores podrán gestionar las sustituciones de forma completa: re asignar guardias, consultar todas las bajas registradas y crear modelos de horario para asignarlos al profesorado. El sistema les propondrá asignaciones automáticas basadas en disponibilidad y equilibrio de cargas, pero también podrán reasignar sustituciones manualmente cuando haga falta. Además contarán un panel de notificaciones que alertará sobre casos de sobrecarga o falta de profesorado y con herramientas para gestionar usuarios (verlos y eliminarlos) y configurar el límite de guardias. Por su parte, los administradores tendrán la función de crear y configurar centros.

Además, introduce mejoras en la transparencia del proceso, evitando que la responsabilidad recaiga en una sola persona y fomentando un reparto justo y controlado a través de criterios personalizados. Esta plataforma cubre un nicho aún sin explotar por sistemas más completos, en especial, para las instituciones pequeñas y medianas.

La integración exclusiva con Firebase para actualizaciones en tiempo real y autenticación segura permite una arquitectura serverless de bajo costo, lo que facilita su adopción en escuelas con presupuestos limitados. Finalmente, aunque no implementado, el proyecto podría tener una integración con APIs de calendarios educativos españoles (como los del Ministerio de Educación), justificando una mayor personalización cultural y regulatoria que no existe en herramientas globales, promoviendo una gestión más inclusiva y sostenible.

3. Estudio de viabilidad. Método DAFO

Análisis DAFO

El análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) evalúa el proyecto SustiHorario desde una perspectiva interna y externa, identificando aspectos clave para su éxito en el contexto de la gestión de sustituciones docentes.

- **Debilidades**
    - Falta de experiencia en desarrollo integral de software.
    - Recursos humanos limitados.
    - Posible resistencia a cambiar modelos manuales en algunos centros.
- **Amenazas**
    - La fuerte competencia como WebUntis y GHC.
    - Cambios legales o normativos que puedan requerir adaptaciones.
    - Limitaciones presupuestarias en centros educativos para nuevas herramientas.
- **Fortalezas**
    - Enfoque claro hacia la simplicidad y accesibilidad para todos los usuarios.
    - Adaptación a necesidades reales detectadas en centros educativos pequeños o medianos.
    - Herramienta colaborativa que facilita la gestión y la equidad en sustituciones.
- **Oportunidades**
    - Creciente digitalización que favorece soluciones tecnológicas.
    - La necesidad de herramientas sencillas, para la organización y administración en educación.
    - Tecnologías accesibles y económicas para desarrollo y despliegue.

a. Estudio del mercado

El mercado actual necesita soluciones para administrar con eficacia y de forma justa las sustituciones docentes, dado el gran volumen de bajas y la frecuencia de reemplazos necesarios en las escuelas. Las plataformas disponibles, aunque robustas, no siempre son idóneas para escuelas pequeñas o con presupuestos limitados. Esto crea hueco para aplicaciones simples, accesibles y efectivas, como SustiHorario.

El público meta incluye escuelas públicas y privadas de primaria y secundaria, coordinadores y responsables académicos y también, por supuesto, los profesores sustitutos, necesitados de poder consultar sus horarios, sencillamente y desde cualquier dispositivo.

i. Factibilidad técnica y económica

El proyecto es viable técnicamente gracias a tecnologías maduras y accesibles. Económicamente, su costo es bajo.

1. **Recursos HW:** Un ordenador personal (PC o laptop con al menos 8GB RAM, procesador i5 o equivalente) para desarrollo y testing. Smartphones Android/iOS para pruebas móviles. No requiere servidores dedicados, ya que Firebase maneja el backend en la nube.

2. **Recursos SW:** Flutter (framework open-source para apps cross-platform), Dart (lenguaje principal), Firebase (Auth, Firestore, Cloud Messaging para real-time). Entorno de desarrollo: Visual Studio Code (gratuito). Librerías: intl para fechas, go_router para navegación, cloud_firestore para BBDD.

3. **Recursos humanos:** Un desarrollador, con conocimientos en programación móvil y cloud. No sería necesario expertos externos, aunque se podría consultar para validación.

ii. Viabilidad temporal

El proyecto es viable temporalmente, con un plazo de 4 meses (marzo-junio 2026). Esta duración va con un curso, así el proyecto se hace en el tiempo del curso y así se instala poco a poco en el lugar y asegurando entrega en plazo.

b. Planificación Temporal o Agenda de Trabajo

La planificación se basa en un diagrama de Gantt, que divide el proyecto en fases, desde investigación hasta documentación.

Planificación y Diseño (1 mes**)**

- La definición de requisitos y objetivos aplicados a la gestión automática de sustituciones en centros educativos.
- Realizar preguntas y encuestas con profesores de centro educativos.
- Investigación de mercado y análisis de competencia (WebUntis, GHC).
- El diseño de la arquitectura de la aplicación, centrada en facilidad de uso para profesores y responsables.
- La creación de wireframes y prototipos de la interfaz de usuario clara y accesible.

Desarrollo (2 - 3 meses**)**

- Implementación de la funcionalidad principal,con generación automática y gestión de horarios de sustitución.
- Desarrollo de filtros personalizados y criterios para una distribución justa.
- Creación de interfaces para dispositivos móviles y ordenadores.

Pruebas (1 - 2 meses**)**

- Pruebas funcionales y de usabilidad con grupos piloto.
- Recopilación y análisis de feedback para mejoras.
- Corrección de errores y optimización de la experiencia.

Lanzamiento (2 semanas**)**

- Configuración final y despliegue en servidores.
- Documentación y manuales para usuarios y responsables.
- Presentación y formación para usuarios finales.

Mantenimiento y Actualizaciones (continua**)**

- Seguimiento continuo, correcciones y mejoras basadas en feedback.
- Implementación de nuevas funcionalidades.
- Soporte técnico a usuarios.

Diagrama de Gantt

Fase	Duración Estimada	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
Planificación y Diseño	1 mes						
Desarrollo	3 meses						
Prueba	2 meses						
Lanzamiento	1 mes						
Mantenimiento	Continuo						

Estudio Económico

Forma Jurídica

La forma jurídica recomendada para el desarrollo y comercialización del proyecto SustiHorario es la Sociedad Limitada (SL). Esta estructura permite distribuir las responsabilidades adecuadamente, facilita la obtención de financiación y proporciona un marco legal que protege a los socios. Además, es adecuada para startups y proyectos tecnológicos.

Gastos Estimados

Los principales gastos asociados al proyecto se dividen en las siguientes categorías:

- **Recursos Humanos**: salario para el desarrollador/diseñador con posibles consultores externos en UX/UI o bases de datos.
- **Recursos Hardware y Software:** inversión en ordenadores, servidores o servicios de alojamiento en la nube, licencias de software necesarias (IDE, base de datos, frameworks), y herramientas de colaboración.
- **Marketing y Publicidad:** campañas iniciales para dar a conocer la aplicación, estrategias de comunicación y mantenimiento de redes sociales o canales digitales.
- **Gastos Operativos:** costos relacionados con oficina, suministros, seguros y mantenimiento técnico.

Se estima que el coste inicial para el desarrollo completo puede tener un costo entre los 10.000 y 20.000 euros, considerando recursos básicos y uso prioritario de software libre o de bajo coste. Los gastos anuales posteriores para mantenimiento, actualizaciones y soporte serían menores, en torno a un 25-30% del coste inicial.

Modelo de Negocio

SustiHorario se plantea como una aplicación dirigida mayoritariamente a centros educativos públicos y privados que requieren una solución accesible y sencilla para la gestión de sustituciones. Se propone un modelo basado en:

- **Suscripciones**: planes mensuales o anuales con soporte y diseño adaptado con actualizaciones.
- **Freemium**: acceso gratuito con funcionalidades y versiones generales globales sin personalización.

Estrategias de Financiación y Crecimiento

El proyecto buscará financiación inicial a través de subvenciones, programas de apoyo a emprendedores tecnológicos o crowdfunding dirigido a comunidades educativas y docentes. Además, se fomentará el crecimiento orgánico mediante recomendaciones y buena experiencia del usuario.

Inversión del proyecto

Concepto	Descripción		Coste Inicial (€)	Coste Anual (€)
Oficia				
Oficina	Espacio físico para desarrollo y gestión del proyecto		1.500,00 €	3.600,00 €
Hardware				
- Servidores	Servidores necesarios para alojar la aplicación y los datos de los usuarios.		1.200,00 €	600,00 €
- Dispositivos Móviles para Pruebas	Dispositivos móviles para pruebas y desarrollo de la aplicación.		800,00 €	600,00 €
Software				
- Licencias de Desarrollo	Licencias para herramientas de desarrollo como Unity y diseño gráfico.		500,00 €	300,00 €
- Herramientas de Gestión	Herramientas para la colaboración y gestión del proyecto.		300,00 €	150,00 €
Recursos Humanos		Numero Empleados		
- Desarrolladores de Software	Salarios para desarrolladores encargados de programar y desarrollar la aplicación.	1	12.000,00 €	24.000,00 €
- Diseñadores Gráficos	Salarios para diseñadores encargados de crear una interfaz atractiva y funcional.	1	4.000,00 €	8.000,00 €
- Expertos en Marketing Digital	Salarios para expertos encargados de estrategias de lanzamiento y promoción.	1	3.000,00 €	3.000,00 €
- Personal de Soporte y Administración	Salarios para personal encargado de soporte técnico y administración.	1	2.000,00 €	4.000,00 €
Servicio Post-Venta	Implementación de un sistema de soporte técnico y atención al cliente.		2.000,00 €	2.000,00 €
Seguro de Responsabilidad	Contratar un seguro para cubrir posibles riesgos legales.		1.000,00 €	500,00 €
Marketing y Publicidad	Campañas de marketing y publicidad para promocionar la aplicación.		5.000,00 €	8.000,00 €
		Total Inversión Inicial	31.300,0 €0	
		Total Coste Anual		51.250,0€0

Comparación con la competencia

- webuntis : la tarifa dependerá del centro educativo que proporcione el acceso
- GH: tiene una tarifa de GHC Optimum 95€ I.V.A. o una versión gratuita pero limitada

Ventaja competitiva

SustiHorario se diferencia al ser una aplicación accesible, económica y adaptada a las necesidades reales de centros educativos pequeños y medianos, priorizando la simplicidad en la gestión y una interfaz intuitiva, ideal para usuarios con diferentes niveles tecnológicos.

Viabilidad temporal

El proyecto es viable desde el punto de vista temporal, con un plan organizado que contempla fases de análisis, diseño, desarrollo, pruebas y despliegue. Se estima una duración total de 4 a 6 meses, suficiente para implementar y estabilizar la aplicación antes de su uso generalizado.

4. Análisis de Requisitos

El objetivo del análisis de requisitos es identificar y describir las necesidades y expectativas de los usuarios finales para que el sistema SustiHorario cumpla con sus funciones esenciales. Se abordan tanto requerimientos funcionales relacionados con las operaciones que el sistema debe realizar, como no funcionales enfocados en la usabilidad, rendimiento, seguridad y accesibilidad.

Los requisitos se derivan de análisis de necesidades: los profesores necesitan declarar disponibilidades y solicitar bajas fácilmente; coordinadores requieren herramientas para reasignar guardias de forma justa entre otras ya nombradas; administradores gestionar la creación de centros.

Requisitos funcionales

- **Autenticación y Registro:** El sistema permite el login con email/contraseña (página de acceso) y registro de nuevos usuarios seleccionando rol y centro (formulario que permitirá ingresar nombre, email, contraseña y asociar a un centro educativo).
- **Gestión de Horarios y Disponibilidades:** Los Coordinador pueden añadir clases fijas y marcar horas disponibles para guardias, con visualización en los perfiles de los profesores.

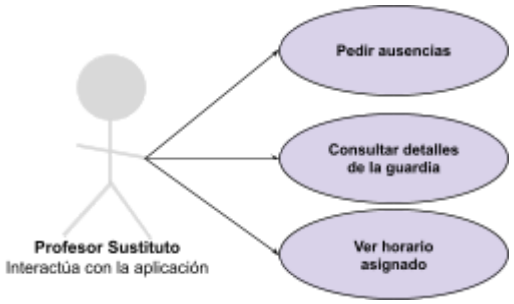
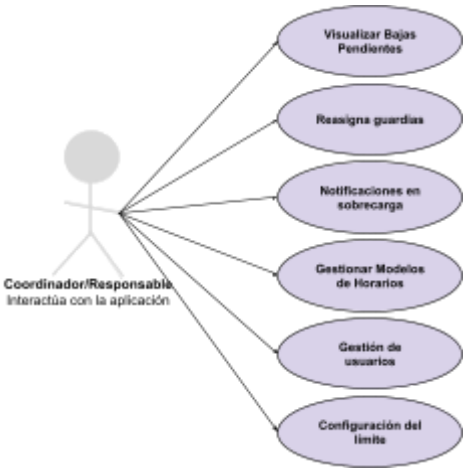
- **Solicitud de Bajas y Guardias:** Formularios para solicitar ausencias, con asignación automática o manual por coordinadores, incluyendo reasignaciones.
- **Paneles por Rol:** Vista específica para profesores (horarios, guardias, bajas), coordinadores (asignación de sustituciones, vistas globales, etc..) y administradores (gestión de centros).
- **Consultas y Actualizaciones:** Interacciones con BBDD para cargar/guardar datos en Firestore, como horarios y disponibilidades, con timestamps para auditoría.

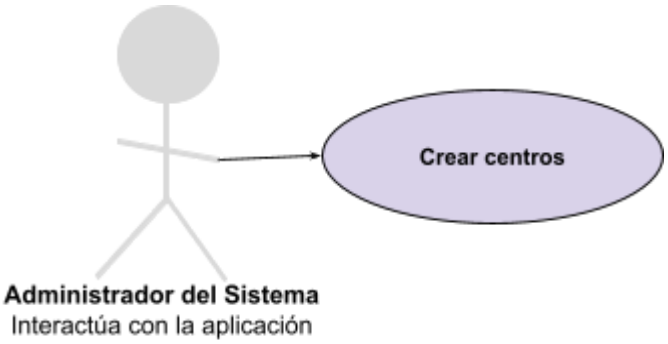
Requisitos no funcionales

- **Usabilidad:** Interfaz intuitiva con diseños responsive.
- **Seguridad:** Autenticación segura vía Firebase Auth.
- **Rendimiento:** Carga rápida de datos en tiempo real, manejo de errores y optimización para dispositivos con conexiones variables.
- **Escalabilidad:** Soporte para múltiples centros educativos, con límites en consultas Firestore para costos bajos.
- **Mantenibilidad:** Código modular con providers y rutas definidas.

Diagramas de Casos de Uso

Se presentan diagramas de casos de uso en UML para ilustrar interacciones. Se incluye un caso de uso general y específicos por perfil.

Caso de uso de un profesor sustituto	Caso de uso de un Coordinador / Responsable
	

Caso de uso de un Administrador del Sistema


5. Diseño

Los elementos de diseño de SustiHorario se definieron en las fases iniciales del proyecto y han evolucionado durante su implementación, adaptándose a las necesidades detectadas y a las limitaciones técnicas de Flutter + Firebase. El diseño prioriza la simplicidad, la usabilidad móvil y la escalabilidad en un entorno serverless.

a. Diseño Conceptual Entidad-Relación (ERD)

Aunque SustiHorario utiliza una base de datos NoSQL (Firestore), que no requiere un modelo relacional estricto, se elaboró un diagrama conceptual Entidad-Relación para representar la lógica de datos de forma clara y facilitar la comprensión del modelo.

Entidades principales y relaciones

- **Centro**
    - Atributos: id (PK), nombre, dirección
    - Relación: 1:N → Usuarios (un centro tiene muchos profesores/coordinadores/administradores)
- **Usuario**
    - Atributos: uid (PK), nombre, email, rol (profesor/coordinador/admin)
    - centroId (FK) Relación: 1:1 → Horario (cada usuario tiene su propio horario)
    - Relación: 1:N → GuardiasAsignadas (un profesor puede tener varias guardias)

- **Horario**
    - Atributos: uid (PK), nombreProfesor, centroId, horarioFijo (mapa), disponibilidad (mapa) updatedAt
    - Relación: N:1 → Usuario
- **Baja**
    - Atributos: id, profesorId, fechaInicio, fechaFin, horasAfectadas (array), tipo, estado
    - Relación: N:1 → Usuario (profesor solicitante)
- **Guardia**
    - Atributos: id, dia, hora, profesorAusenteId, sustitutoId, asignatura, curso, aula, tipo
    - Relación: N:1 → Usuario (ausente y sustituto)

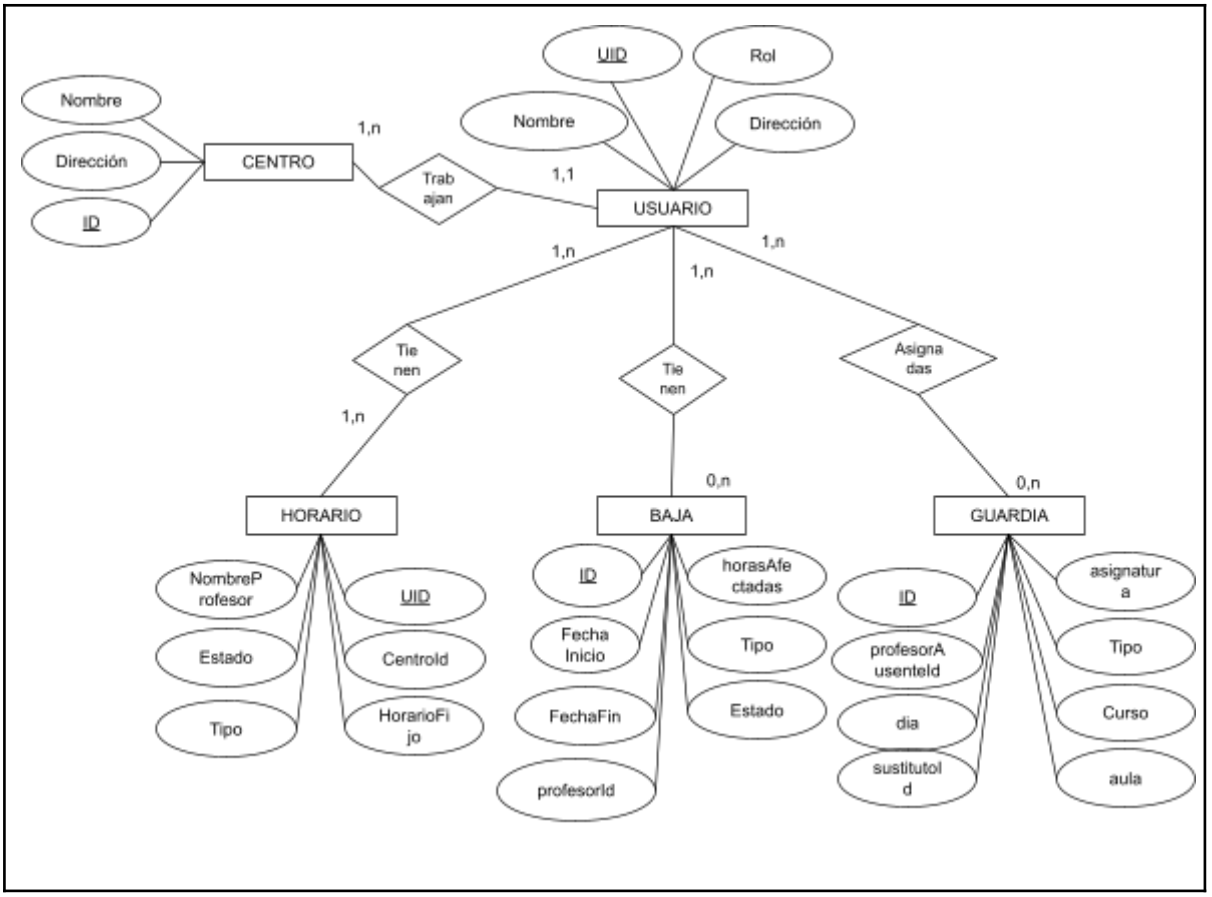


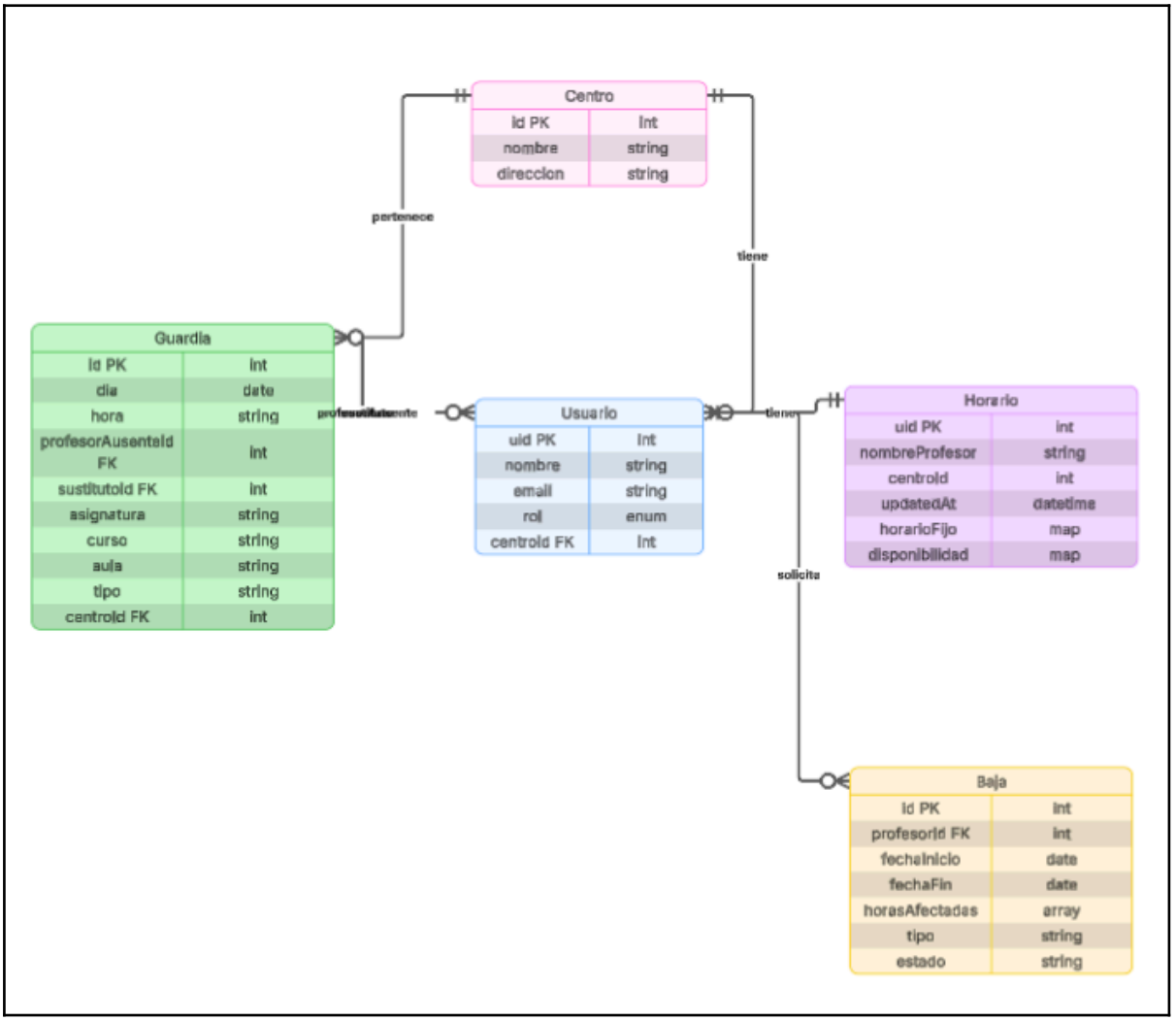
Diagrama ER conceptual

Entidades principales:

- **Centro Atributos:** id (clave primaria), nombre, dirección
- **Usuario Atributos:** uid (clave primaria), nombre, email, rol (profesor / coordinador / admin), centroId (clave foránea)
- **Horario Atributos:** uid (clave primaria, mismo que usuario), nombreProfesor, centroId, updatedAt Nota: Contiene dos estructuras embebidas: horarioFijo (mapa) y disponibilidad (mapa)
- **Baja Atributos:** id (clave primaria), profesorId (FK), fechaInicio, fechaFin, horasAfectadas (array), tipo, estado
- **Guardia Atributos:** id (clave primaria), dia, hora, profesorAusenteId (FK), sustitutoId (FK), asignatura, curso, aula, tipo

Relaciones:

- Centro (1) → Usuario (N) [Un centro tiene muchos usuarios]
- Usuario (1) → Horario (1) [Cada usuario tiene un único horario]
- Usuario (1) → Baja (N) [Un profesor puede solicitar varias bajas]
- Usuario (N) ← Guardia → Usuario (N) [Muchos-a-muchos: profesor ausente y sustituto]
- Guardia (N) → Centro (1) [Todas las guardias pertenecen a un centro]



En Firestore, la estructura real es jerárquica y denormalizada para optimizar lecturas:

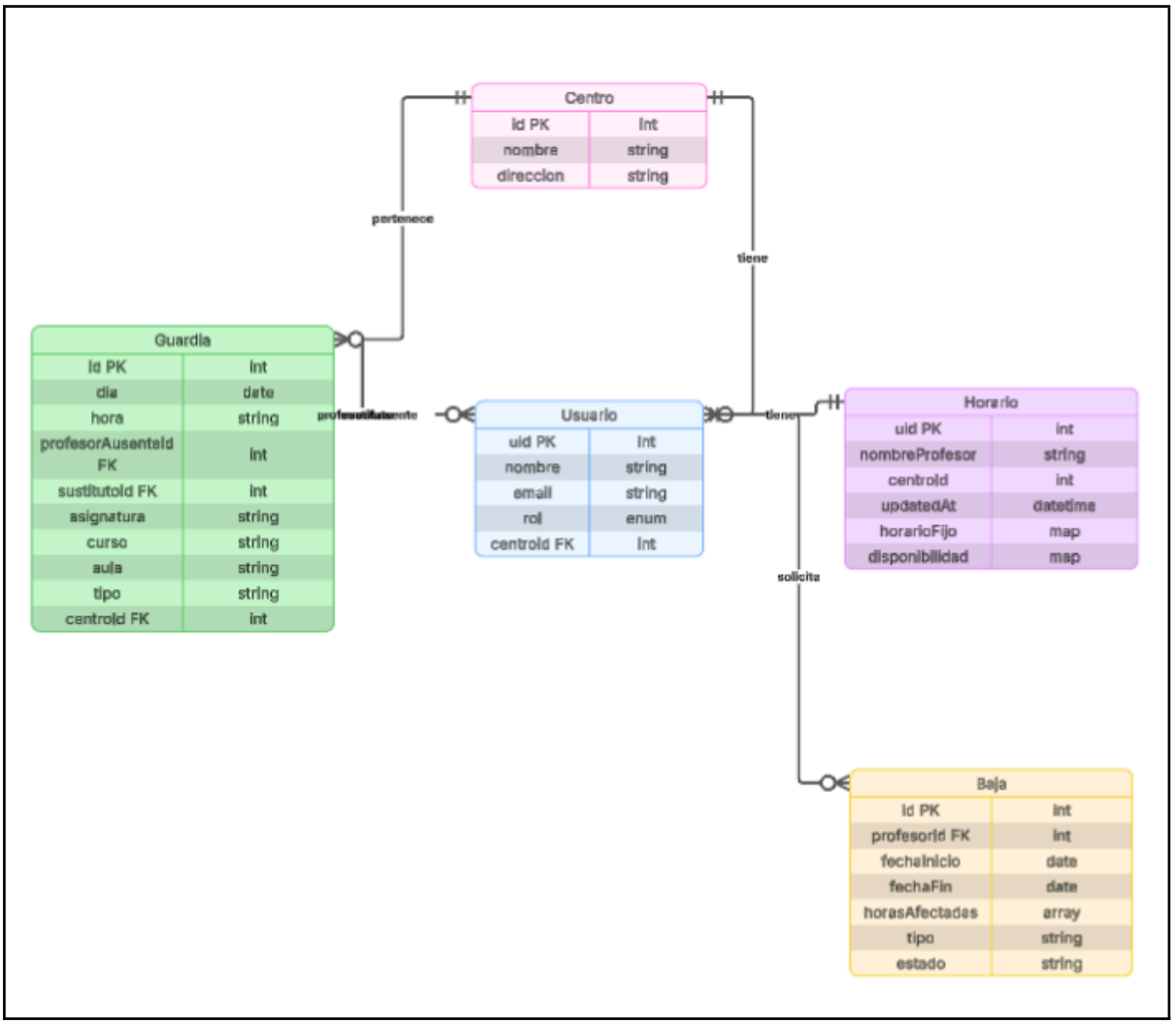
- Colección centros → documentos con nombre y dirección
- Colección users → uid como id, con campo centroId
- Colección horarios → uid como id del documento, con mapas embebidos para horarioFijo y disponibilidad
- Colección bajas y guardias para registros temporales y asignaciones

b. Diseño Lógico Relacional o Paso a tablas

Dado que Firestore es NoSQL, no se aplicó un diseño relacional estricto con tablas normalizadas. En su lugar, se utilizó un enfoque denormalizado típico de Firestore:

- Los datos de usuario y centro se duplican parcialmente para evitar joins costosos (lecturas rápidas).
- Mapas embebidos para horarioFijo {hora → {día → clase}} y disponibilidad {hora → {día → boolean}}.
- Arrays para horas afectadas en bajas.
- Referencias (centroId, uid) para mantener integridad referencial lógica.

Esta estructura prioriza la velocidad de lectura (frecuente en consultas de horarios) sobre la escritura, alineándose con las mejores prácticas de Firestore.



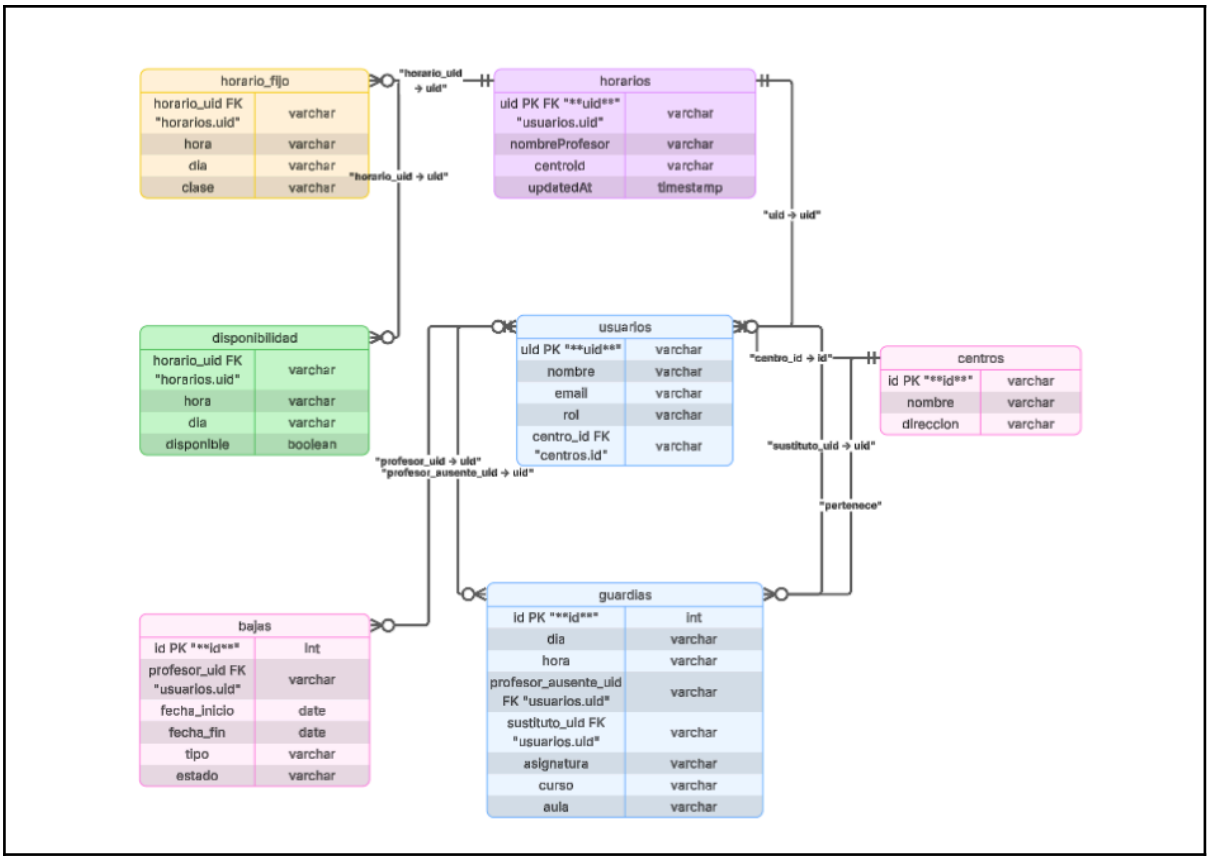
c. Diseño Físico o Diagrama MySQL

Aunque el proyecto no usa MySQL (sino Firestore), a continuación se describe una posible traducción a modelo relacional tradicional para referencia o futura migración:

Un diagrama de modelo relacional. Mostrar las siguientes tablas con columnas y claves:

1. **centros** (id, nombre, direccion)
2. **usuarios** (uid , nombre, email, rol , centro_id)
3. **horarios** (uid, nombreProfesor, centroId, updatedAt)
4. **horario_fijo** (horario_uid, hora, dia, clase)
5. **disponibilidad** (horario_uid, hora, dia, disponible)
6. **bajas** (id, profesor_uid, fecha_inicio, fecha_fin, tipo, estado)
7. **guardias** (id, dia, hora, profesor_ausente_uid, sustituto_uid, asignatura, curso, aula)

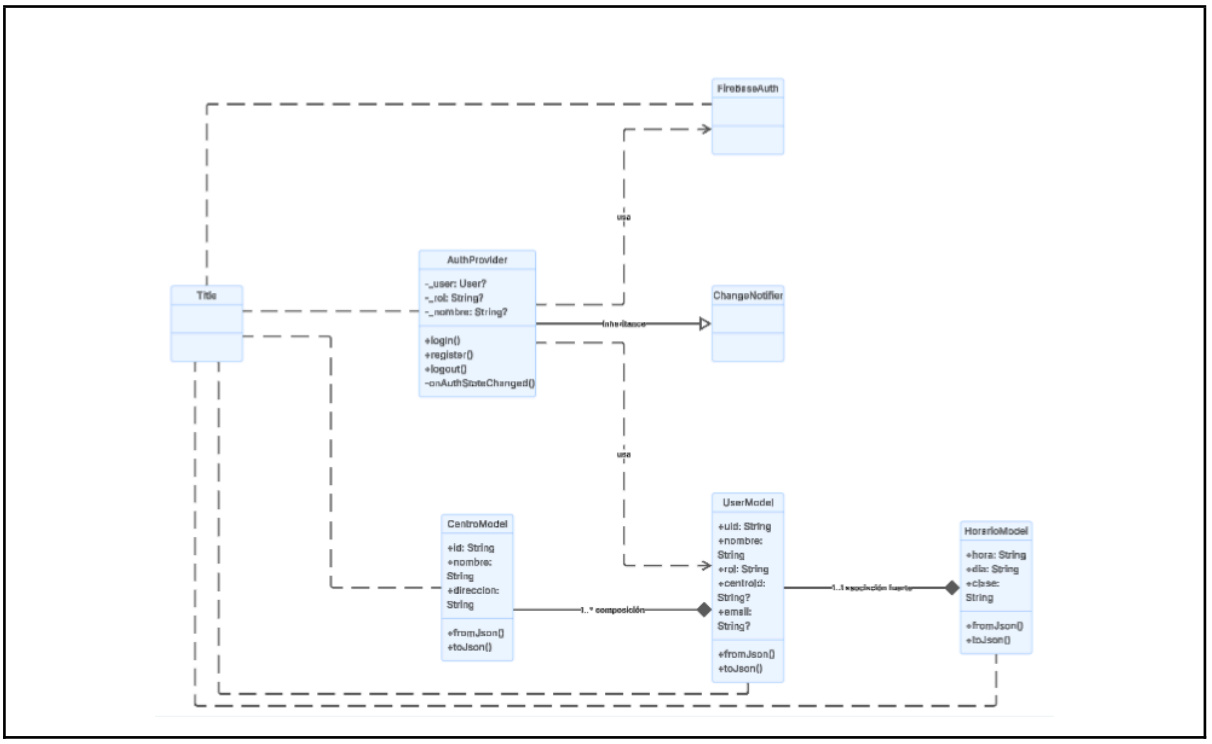
Este modelo relacional sería equivalente, pero implicaría más joins y menor rendimiento en lecturas frecuentes.



d. Orientación a Objetos

1. Diagramas de clases. Descripción de clases y atributos

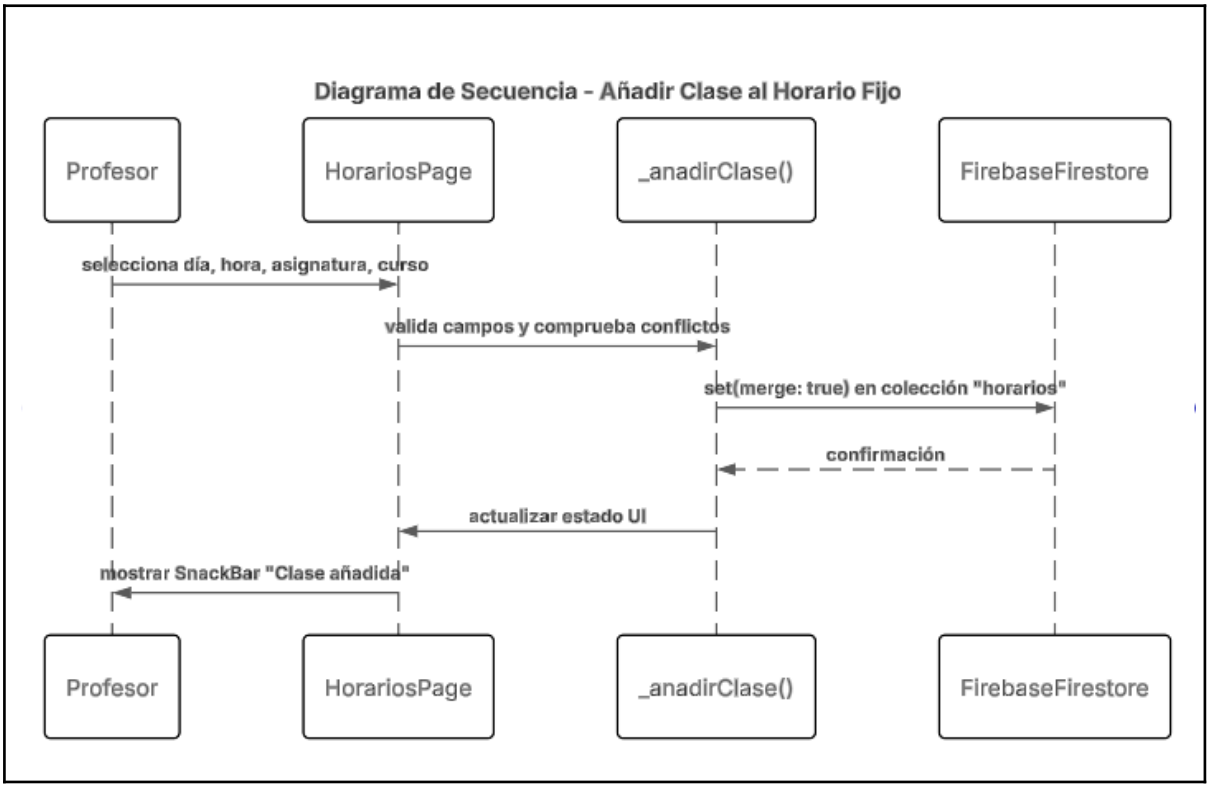
- **CentroModel** Atributos: String id, nombre, direccion Métodos: fromJson, toJson
- **UserModel** Atributos: String uid, nombre, rol, centroId?, email? Métodos: fromJson, toJson
- **HorarioModel** (auxiliar para parsing) Atributos: String hora, dia, clase Métodos: fromJson, toJson



2. Diagrama de secuencias

Secuencia: Añadir clase al horario fijo (Profesor)

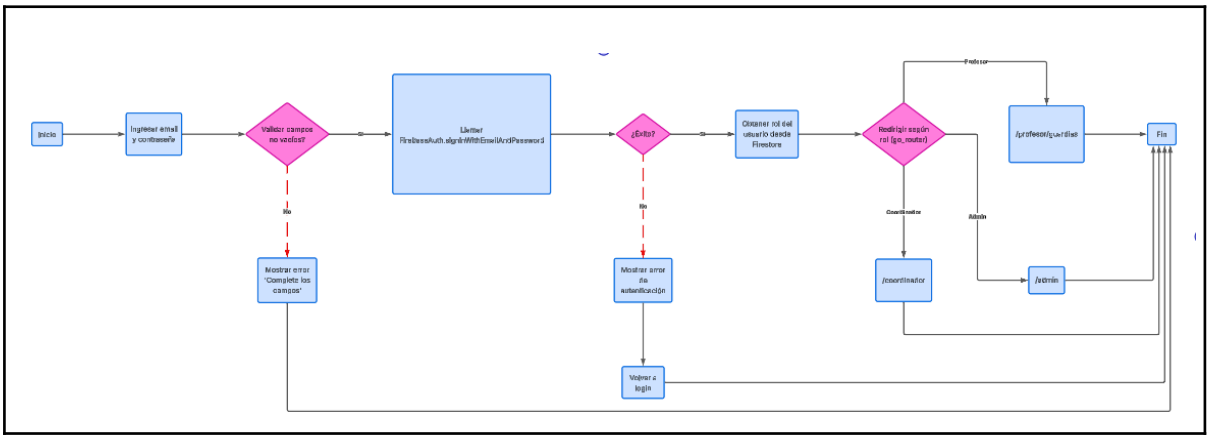
1. Usuario → HorariosPage: selecciona día/hora/asignatura
2. HorariosPage → _anadirClase() → comprueba conflictos
3. HorariosPage → _guardarHorarioEnFirestore()
4. _guardarHorarioEnFirestore → FirebaseFirestore: set(merge: true)
5. Firestore → confirma escritura
6. HorariosPage → muestra SnackBar éxito



3. Diagrama de actividad

Actividad: Login

Inicio → Ingresar email/contraseña → Validar campos → FirebaseAuth.signIn → Éxito? → Sí: redirigir según rol (go_router) → No: mostrar error → Fin

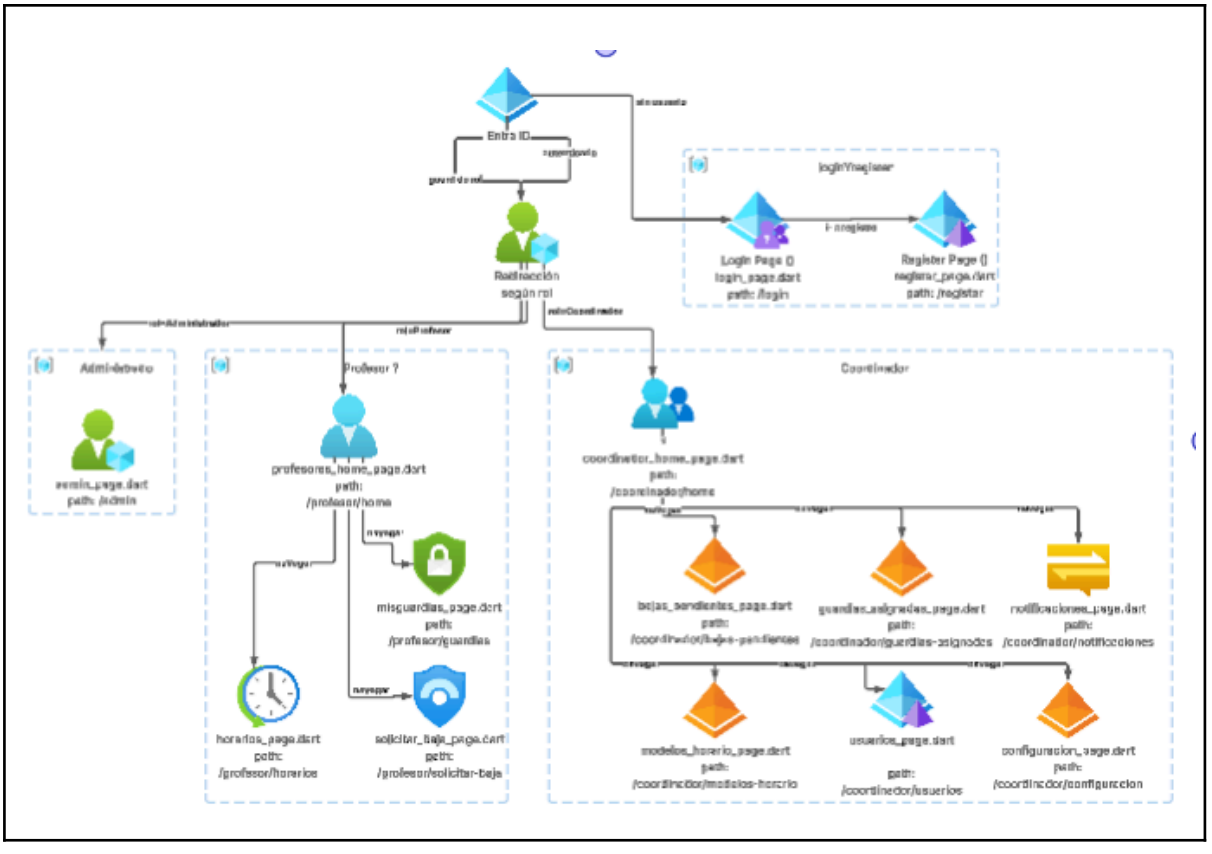


e. Mapa Web

SustiHorario utiliza go_router para navegación declarativa. El mapa de rutas principal es el siguiente:

- Login (/login) ——> Register (/register)
    - Home / Redirección según rol:

Profesor- ProfesoresHomePage	Coordinador - CoordinadorHomePage	Admin
Horarios - (HorariosPage)	**Bajas pendientes** - (BajasPendientesPage)	**Admin** - (AdminPage)
Guardias - (MisGuardiasPage)	**Configuración** - (ConfiguracionPage)	
Solicitud - (SolicitarBajaPage)	**Guardias Asignadas** - (GuardiasAsignadasPage)	
	Modelos Horario- (ModelosHorarioPage)	
	Notificaciones - (NotificacionesPage)	
	Usuarios - (UsuariosPage)	

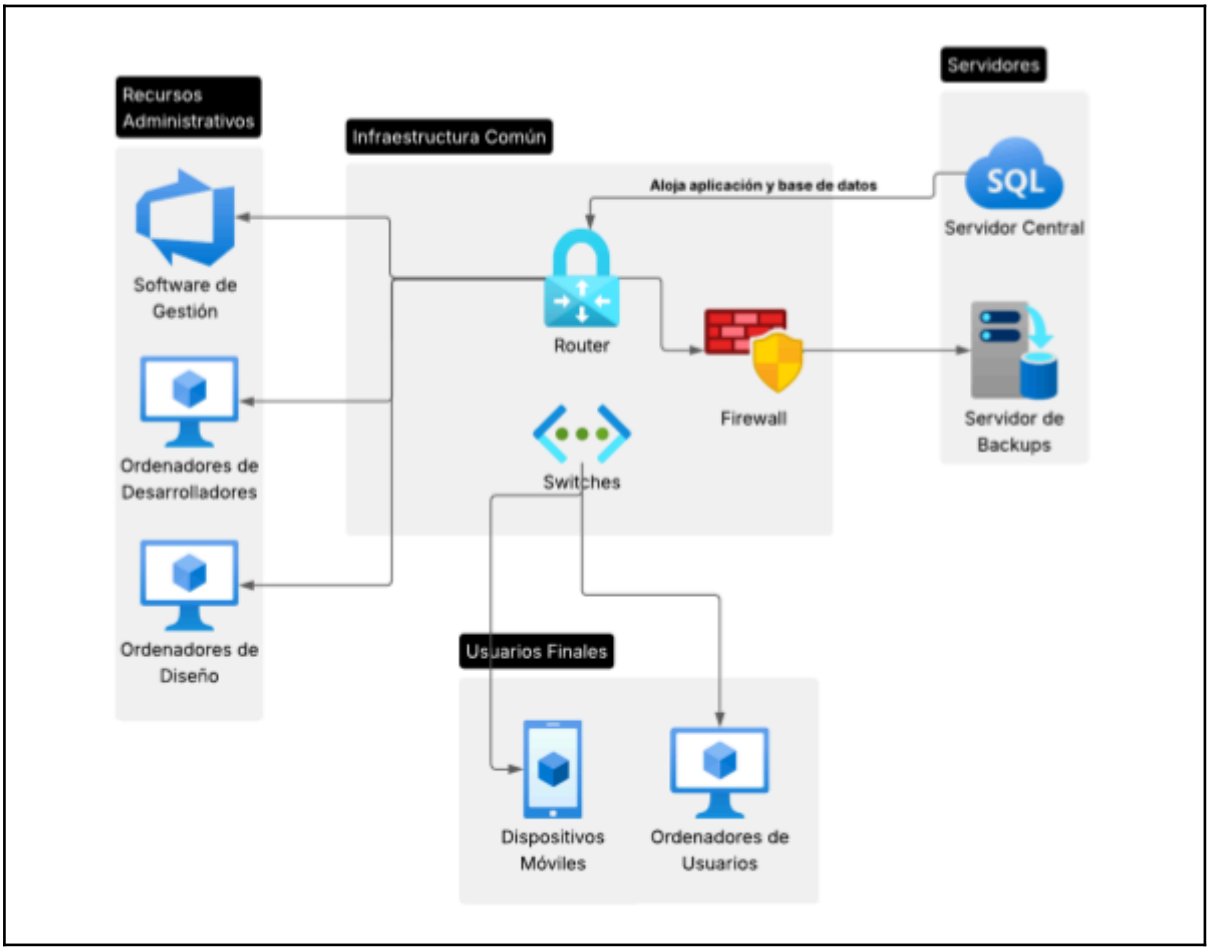


Red interna

Diagrama de red interna.

El esquema muestra:

- Un servidor central que aloja la aplicación y la base de datos.
- Un switch o router que conecta la red interna.
- Ordenadores de los desarrolladores y personal de diseño.
- Dispositivos móviles y ordenadores de usuarios (profesores y coordinadores) en la intranet.
- Elementos de infraestructura común: firewall, servidor de backups, dispositivos de red (switches, routers).
- Software de gestión colaborativa para seguimiento de proyecto conectado a la red.

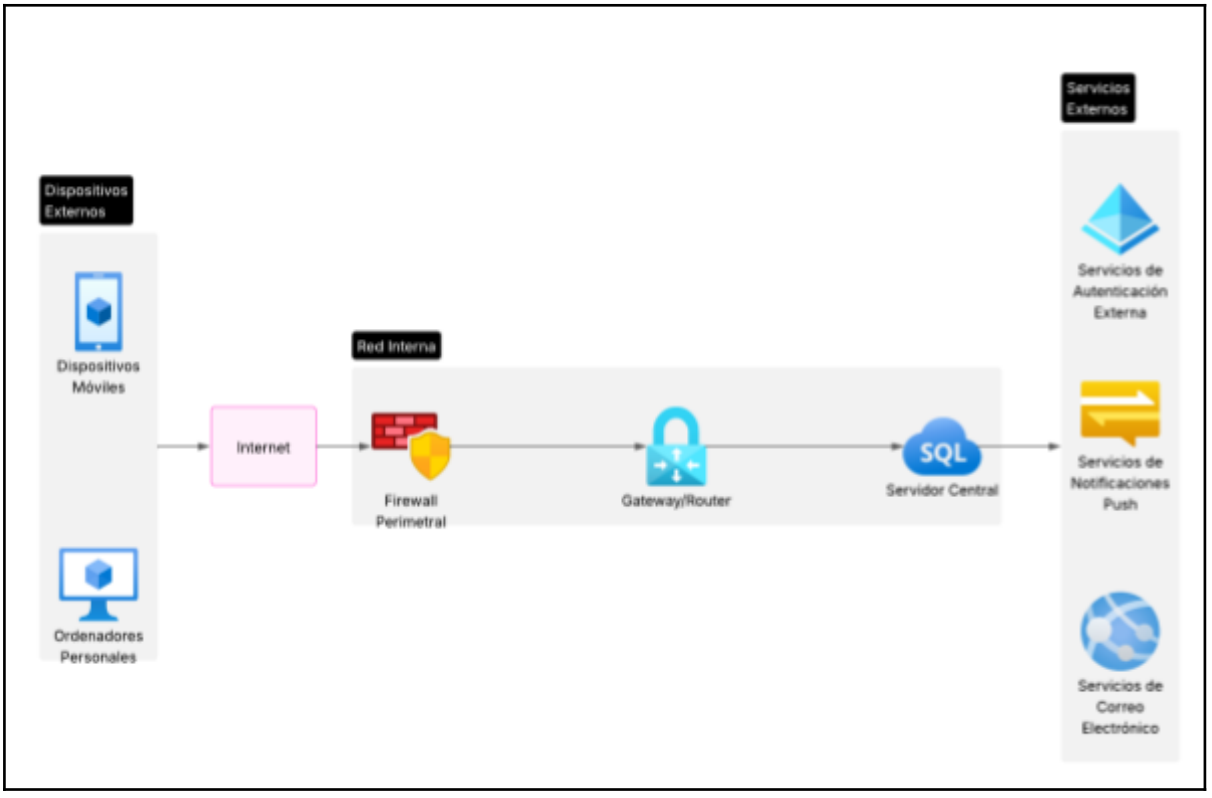


Red externa

Diagrama de red externa, muestre la infraestructura que conecta la aplicación con usuarios y servicios externos.

El esquema muestra:

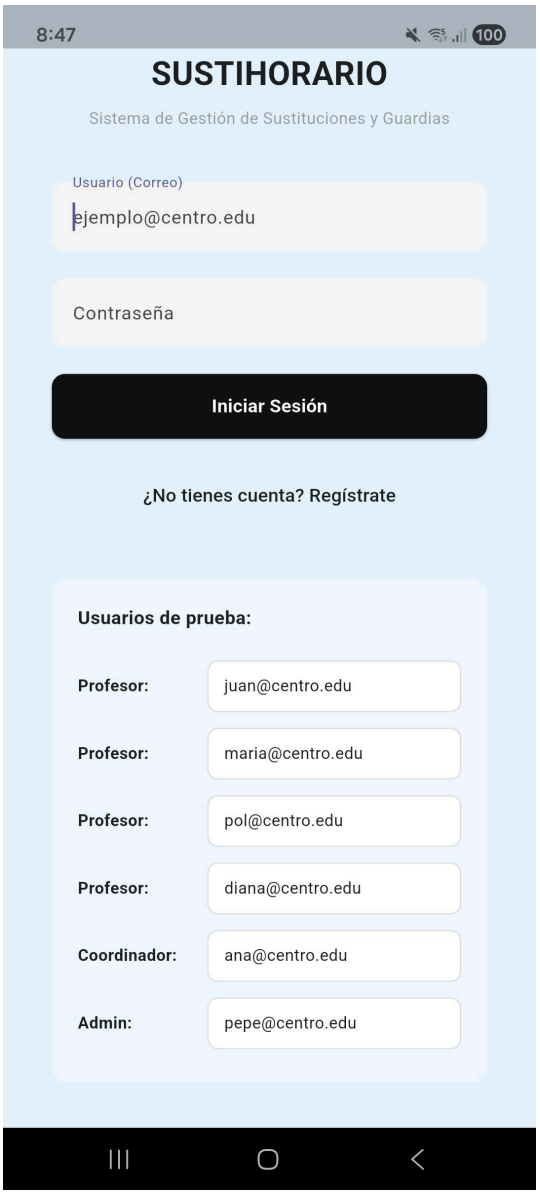
- Internet como entorno externo.
- Dispositivos externos de usuarios: móviles, ordenadores personales (profesores y coordinadores).
- Firewall perimetral que protege la red interna.
- Gateway o router que conecta la red interna con internet.
- Servidores externos o en la nube que alojan la aplicación y base de datos accesibles vía internet.
- Servicios de notificaciones push o correo electrónico.
- Servicios externos de autenticación (si aplica, como OAuth o Google Sign-In).



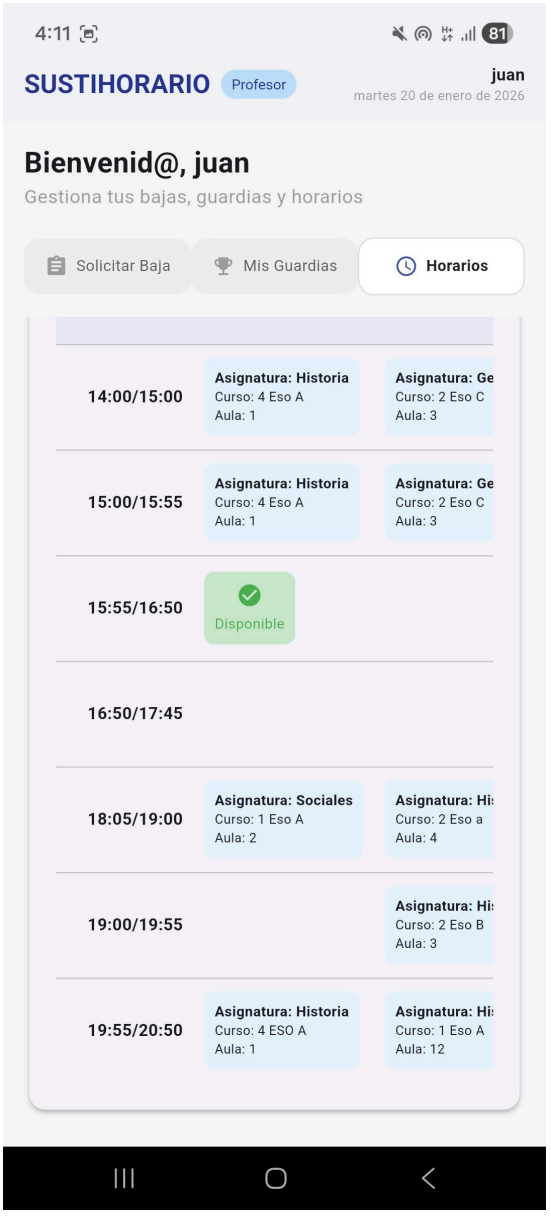
f. Mockups

Aquí vemos veremos un esquema de los paneles programados para la parte visual y estética

- **Panel de Login**



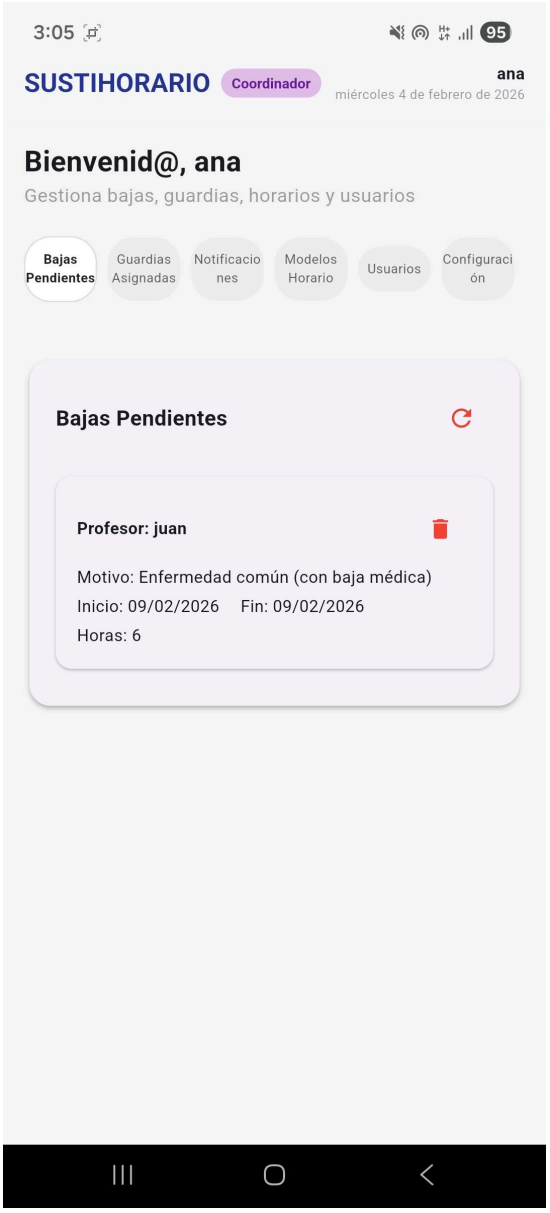
- **Panel Profesor - Horarios**



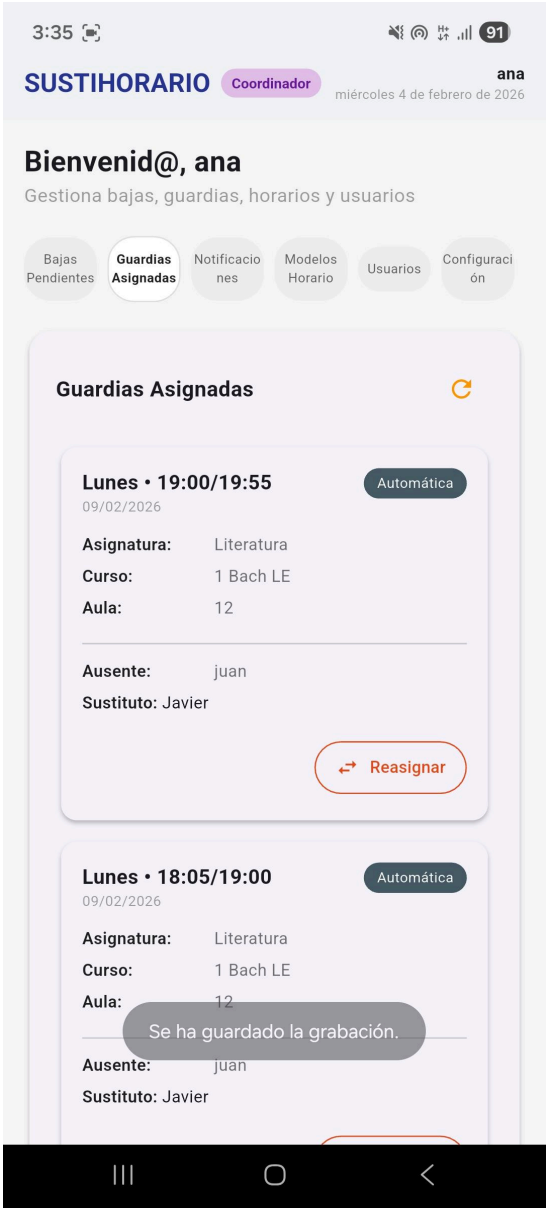
- **Panel Profesor - MisGuardias**

- **Panel Profesor - SolicitarBajas**

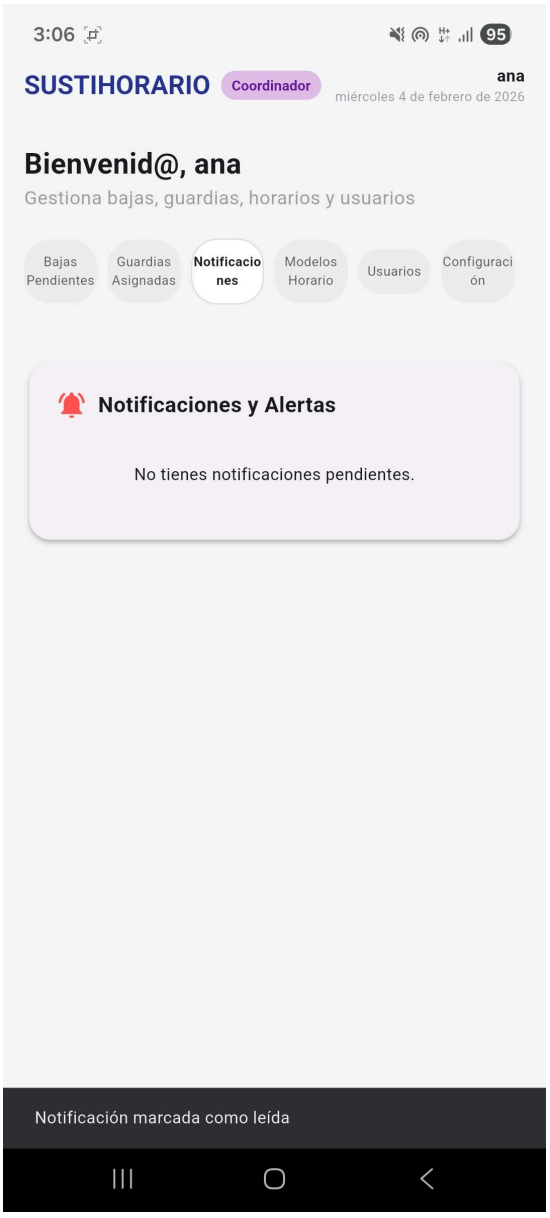
- **Panel Coordinadores - BajasPendientes**



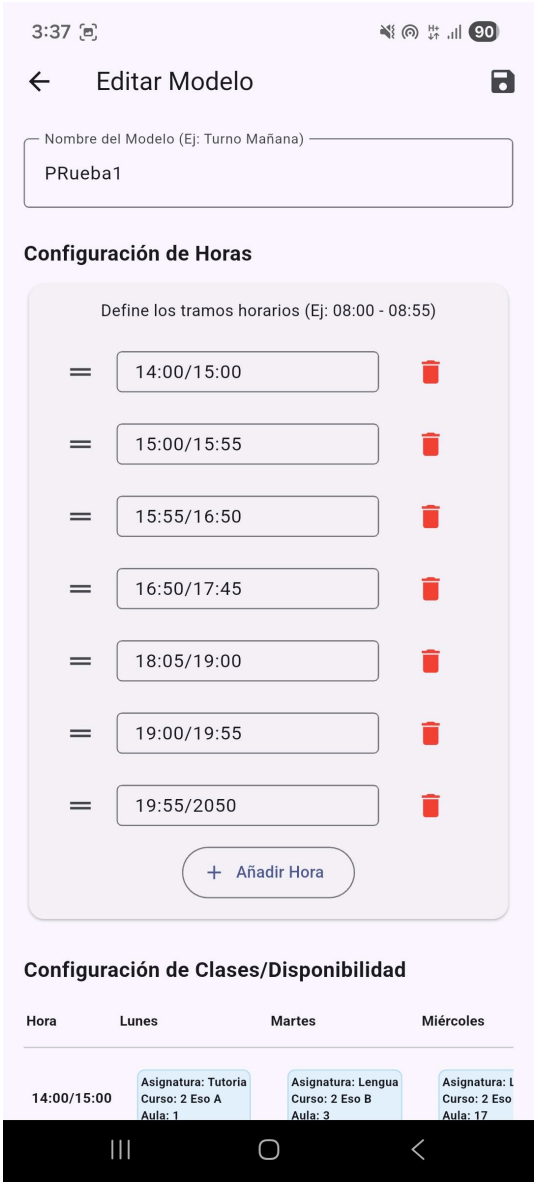
- **Panel Coordinadores - Guardias Asignadas**



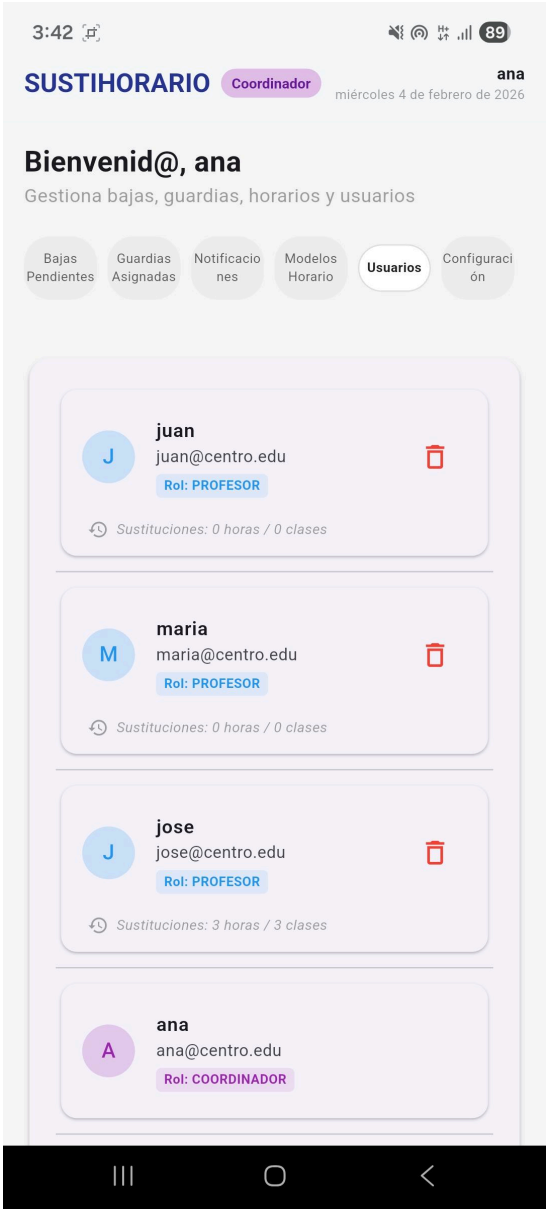
- **Panel Coordinadores - Notificaciones**



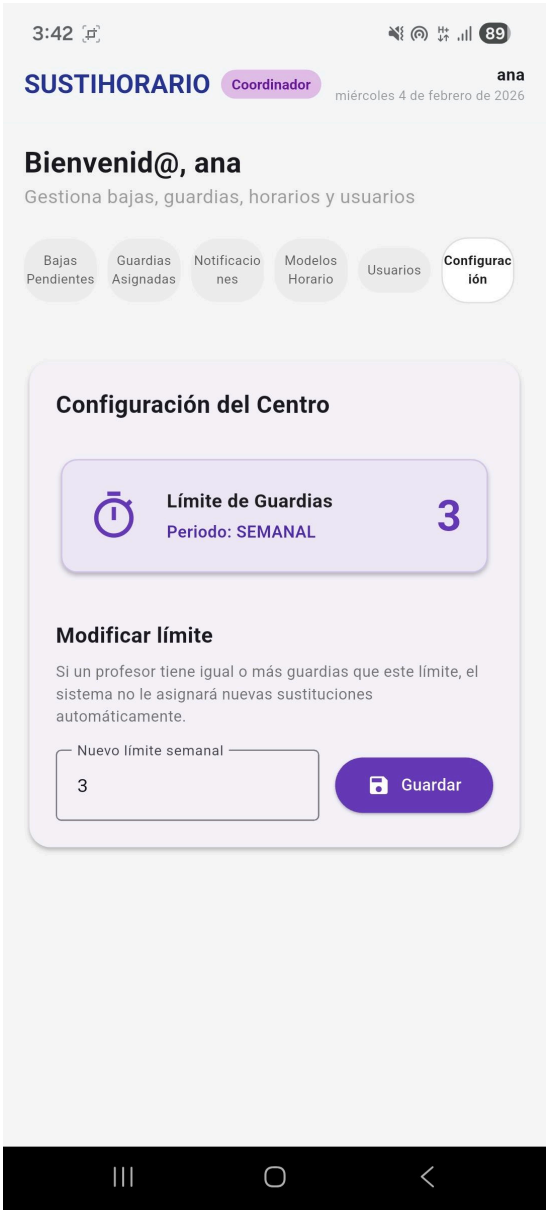
- **Panel Coordinadores - Modelos Horarios**



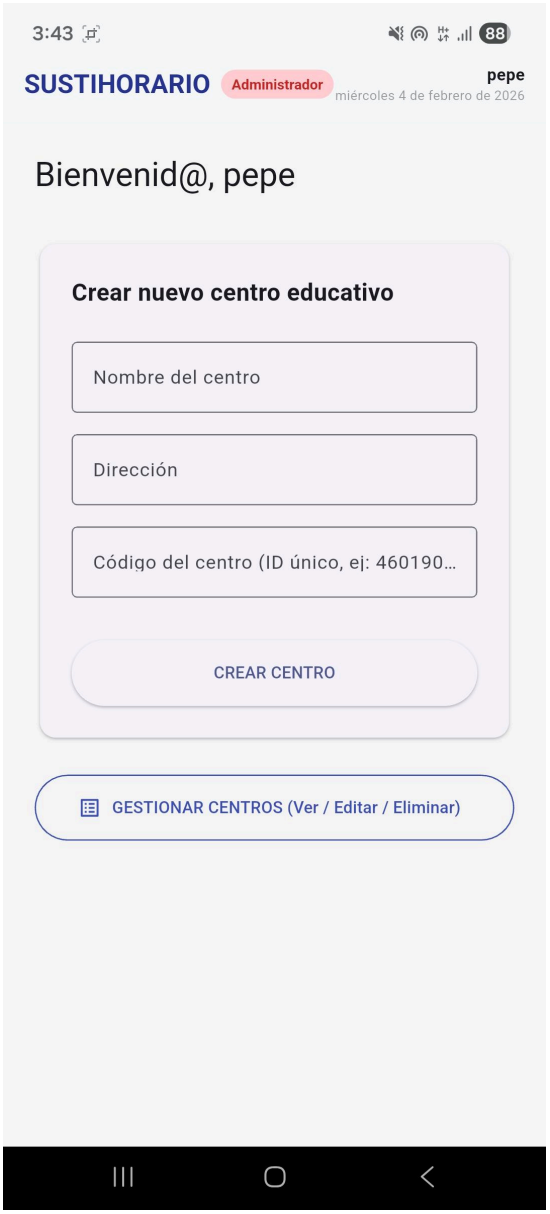
- **Panel Coordinadores - Ususarios**



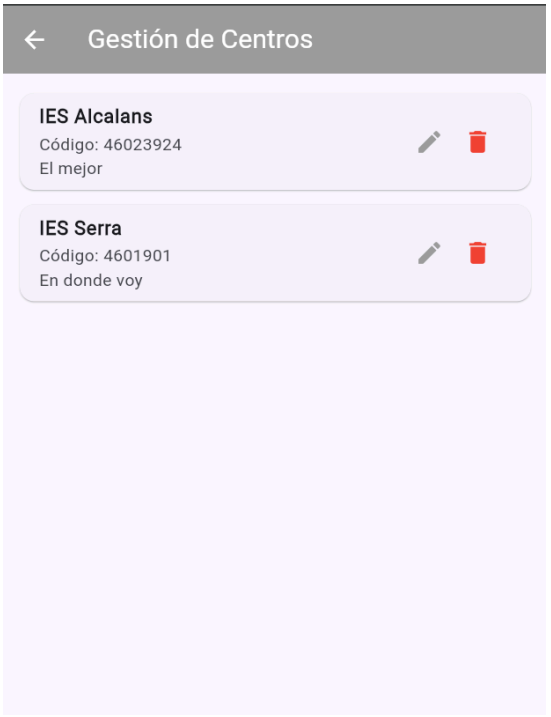
- **Panel Coordinadores - Configuracion**



- **Panel Administrador - Centros**



- **Panel Administrador - Gestion de centros**



6. Codificación

a. Tecnologías elegidas y su justificación (lenguajes, frameworks, librerías, etc.)

Tecnologías principales seleccionadas:

- **Lenguaje principal:** Dart (lenguaje oficial de Flutter)
- **Framework móvil cross-platform:** Flutter
- **Backend y servicios cloud:** Firebase (Firebase Authentication, Firestore)
- **Gestión de estado y providers:** Provider (con ChangeNotifier) + go_router para navegación declarativa
- **Librerías adicionales clave:**
    - cloud_firestore
    - firebase_auth
    - firebase_core
    - intl (internacionalización y formateo de fechas en español)
    - flutter_localizations
    - flutter_dotenv

Comparación y justificación:

Creado en Flutter ya que es el contenido más avanzado multiplataforma que he visto.

- Flutter destaca por su rendimiento cercano al nativo gracias a la compilación AOT (Ahead-of-Time) y su motor de renderizado propio (Skia), ofreciendo animaciones fluidas y consistencia visual perfecta entre iOS y Android sin depender de componentes nativos del sistema.

Flutter cuenta con mayor popularidad en búsquedas y estrellas en GitHub y ofrece mejor soporte para desktop/web/embedded desde una única base de código, aunque en este proyecto solo se usa móvil.

Justificación de la elección de Flutter:

- Desarrollo rápido para proyectos individuales o educativos con plazos ajustados
- Widgets personalizable consistente sin esfuerzo extra
- Aprendizaje con Dart (lenguaje moderno, tipado fuerte, null-safety) dado en clase
- Excelente rendimiento en tablas y vistas complejas.

Justificación de Firebase como backend:

Firebase sigue siendo la opción más sencilla para aplicaciones móviles.

- Firebase ofrece integración nativa perfecta con Flutter
- Facilidad extrema para autenticación, base de datos en tiempo real
- Coste muy bajo (o gratuito) para el proyecto.

b. Entorno servidor

i. Descripción general

El backend es completamente serverless gracias a Firebase. No se gestiona ningún servidor físico o virtual. Los servicios utilizados son:

- **Firebase Authentication** → gestión de usuarios (email/contraseña)
- **Cloud Firestore** → base de datos NoSQL en tiempo real para usuarios, centros, horarios, bajas y guardias
- **Firebase Hosting** → posible despliegue web si se expande la app (no usado en versión actual)

ii. Seguridad. Evitar inyección en bases de datos

Firestore utiliza reglas de seguridad declarativas (Firestore Security Rules) que se configuran en la consola Firebase. En el proyecto se aplican reglas básicas como:

- Solo usuarios autenticados pueden leer/escribir
- Un profesor solo puede modificar su propio horario
- Campos sensibles (email, rol) protegidos contra modificación no autorizada

iii. Evitar o capturar errores y warnings

Se implementa manejo exhaustivo de errores:

- Uso sistemático de try-catch en todas las operaciones asíncronas con Firestore y Auth
- Feedback al usuario mediante ScaffoldMessenger (SnackBar) con mensajes claros y colores (rojo para error, verde para éxito)
- Logging básico mediante print() en desarrollo y posible integración con Firebase Crashlytics en producción
- Validaciones previas en UI (campos obligatorios, conflictos horario vs disponibilidad)
- Estados de carga (_isLoading) para evitar interacciones durante operaciones

c. Entorno cliente

i. Descripción general

Aplicación móvil desarrollada con Flutter, compilada para Android e iOS desde un único código base.

Interfaz mobile con soporte para diferentes tamaños de pantalla (LayoutBuilder para responsive).

ii. Asegurar la funcionalidad en los navegadores más usados

SustiHorario es una aplicación móvil.

La app se compila a binarios nativos (APK/AAB para Android, IPA para iOS) y se instala directamente en dispositivos móviles.

Por tanto, la compatibilidad con navegadores no aplica. Si en el futuro se desea versión web (Flutter Web), se probaría principalmente en Chrome, Edge, Firefox y Safari actuales, aprovechando que Flutter usa Web para renderizado consistente.

d. Documentación interna de código

La documentación interna está integrada directamente en el código mediante comentarios detallados.

i. Descripción de cada fichero principal

- **lib/models/centro_model.dart** → Modelo de datos para centros educativos
- **lib/models/horario_model.dart** → Modelo de datos para horarios
- **lib/models/user_model.dart** → Modelo de datos para usuarios
- **lib/pages/admin/admin_page.dart** → Página principal para administradores
- **lib/pages/coordinador/bajas_pendientes_page.dart** → Página para gestionar bajas pendientes
- **lib/pages/coordinador/configuracion_page.dart** → Página de configuración para coordinadores
- **lib/pages/coordinador/editar_modelo_page.dart** → Página para crear los horarios
- **lib/pages/coordinador/coordinador_home_page.dart** → Página principal del coordinador
- **lib/pages/coordinador/guardias_asignadas_page.dart** → Página para ver guardias asignadas
- **lib/pages/coordinador/modelos_horario_page.dart** → Página para gestionar modelos de horario
- **lib/pages/coordinador/notificaciones_page.dart** → Página de notificaciones para coordinadores

- **lib/pages/coordinador/usuarios_page.dart** → Página para gestionar usuarios
- **lib/pages/loginYregister/login_page.dart** → Página de inicio de sesión
- **lib/pages/loginYregister/register_page.dart** → Página de registro de usuarios
- **lib/pages/profesor/horarios_page.dart** → Página para que profesores consulten sus horarios
- **lib/pages/profesor/misguardias_page.dart** → Página para que profesores vean sus guardias asignadas
- **lib/pages/profesor/profesores_home_page.dart** → Página principal para profesores
- **lib/pages/profesor/solicitar_baja_page.dart** → Página para que profesores soliciten bajas
- **lib/providers/horario_provider.dart** → Proveedor de datos y lógica para horarios
- **lib/providers/usuarios_provider.dart** → Proveedor de datos y lógica para usuarios
- **lib/routes/routes.dart** → Definición de rutas de la aplicación
- **lib/main.dart** → Punto de entrada principal de la aplicación

ii. Descripción de funciones clave

- _guardarHorarioEnFirestore() en horarios_page.dart → Guarda el estado completo del horario y disponibilidad en Firestore con merge para no sobrescribir datos
- _anadirClase() → Válida, comprueba conflictos y añade entrada al mapa horarioFijo
- _login() en login_page.dart → Autenticación con Firebase Auth + redirección según rol
- _registrar() en register_page.dart → Crea usuario en Auth + guarda datos adicionales en colección users

e. Documentación externa

i. Manual del usuario

¿Cómo usar SustiHorario?

1. **Registro** → Accede a /register, introduce nombre, email, contraseña, selecciona rol y centro
2. **Inicio de sesión** → Usa email y contraseña (usuarios de prueba disponibles en login)

3. **Profesor**:
    - Gestiona tu horario fijo y disponibilidad para guardias
    - Solicita bajas
    - Consulta tus guardias asignadas

4. **Coordinador/Admin** → Paneles específicos para asignaciones y gestión

Accesibilidad desde web: Actualmente no existe versión web pública. La aplicación se distribuye como APK/iOS build o mediante TestFlight/Google Play Internal Testing.

Posible futura versión web usando Flutter Web para consulta de horarios.

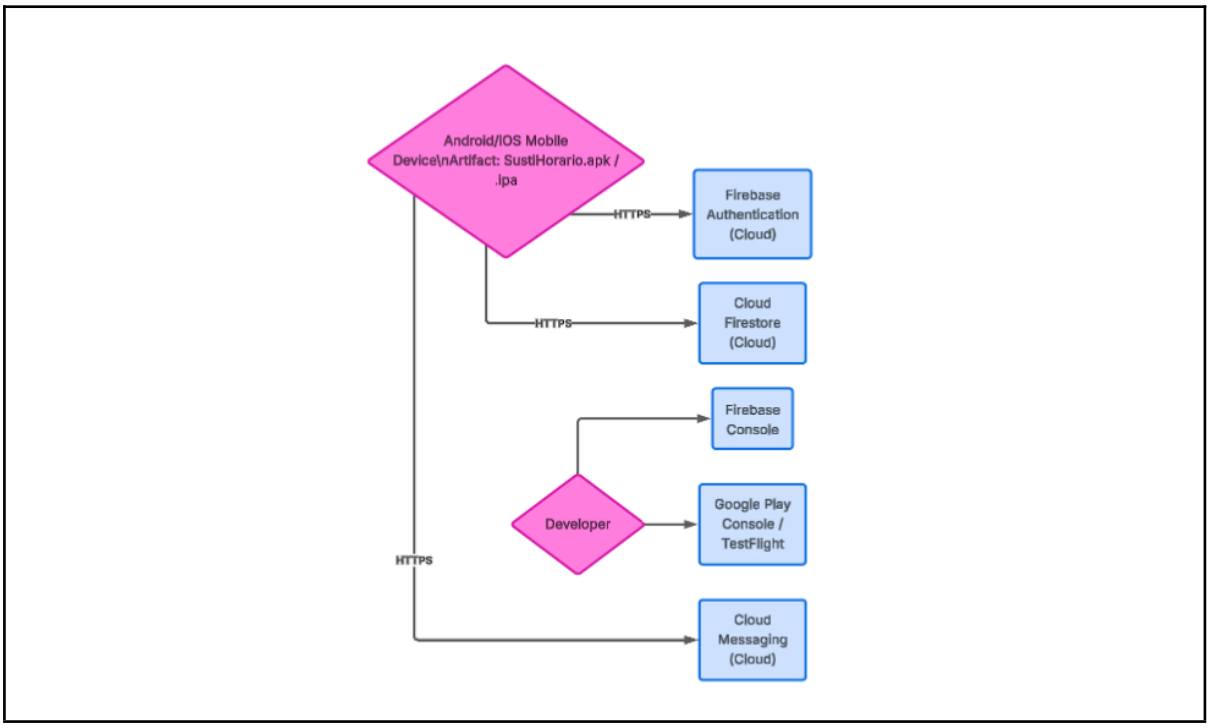
7. Despliegue

SustiHorario es una aplicación móvil desarrollada con Flutter y compilada para las plataformas Android e iOS. Que utiliza una arquitectura serverless completa basada en Firebase. Esto simplifica enormemente el despliegue, elimina la necesidad de gestionar servidores físicos/virtuales y reduce los costes y la complejidad.

a. Diagramas de despliegue

El diagrama de despliegue refleja la arquitectura real del proyecto:

Esquema de despliegue:



b. Descripción de la instalación o despliegue

1. Fichero de configuración

Archivos de configuración utilizados en el proyecto:

- .env (en la raíz del proyecto) Contiene las claves de Firebase necesarias para inicializar la
- Reglas de Firestore Security Rules

2. Descripción del servidor hosting utilizado

El proyecto utiliza exclusivamente servicios **serverless** de Google Firebase, que eliminan la necesidad de contratar o configurar servidores propios. Todos los componentes (autenticación, base de datos, notificaciones) se ejecutan en la infraestructura de Google Cloud, con alta disponibilidad, escalado automático y cero mantenimiento.

- **Plan utilizado:** Firebase Spark Plan (gratuito) durante todo el desarrollo y pruebas (suficiente para cientos/miles de usuarios en fase educativa/prototipo).
- **Coste estimado:** 0 € durante el ciclo formativo. Posible paso a Blaze Plan (pago por uso) si se lanza a producción con muchos usuarios activos.
- **Ventajas:** Despliegue instantáneo de cambios en reglas y funciones, monitoreo integrado, backups automáticos, CDN global para datos.

Distribución de la aplicación para pruebas:

Dado que se trata de una aplicación móvil nativa, las formas habituales de compartirla para pruebas son:

1. **Android**
    - Generar APK o AAB con: flutter build apk --release o flutter build appbundle
    - Subir a **Google Play Internal Testing** (requiere cuenta de desarrollador Google Play, coste 25 $ una vez)
    - URL de invitación: (se genera al crear pista interna en Play Console)
2. **iOS**
    - Generar IPA con: flutter build ipa
    - Subir a **TestFlight** (requiere cuenta Apple Developer, coste 99 $/año)
    - Invitaciones por email a testers

Alternativa gratuita y rápida para pruebas :

Entregar directamente los archivos:

- **Android:**build/app/outputs/flutter-apk/app-release.apk
- **iOS:** Archivo .ipa (solo instalable en dispositivos reales vía TestFlight o con Mac + Xcode)
- **WEB:** accesible mediante URL.

Si se desea en el futuro una versión web, se podría compilar con flutter build web y alojar en **Firebase Hosting** de forma gratuita, pero esto no está implementado en la versión actual del proyecto.

8. Herramientas de apoyo

Durante el desarrollo de SustiHorario, se han utilizado diversas herramientas y buenas prácticas para garantizar la calidad del código, facilitar la colaboración, detectar errores tempranamente y permitir una evolución controlada del proyecto. A continuación se describen las principales categorías empleadas.

a. Control de versiones

Herramienta principal: Git + GitHub

- **Sistema de control de versiones:** Git (versión 2.43+)
- **Plataforma de alojamiento y colaboración:** GitHub (repositorio privado durante el desarrollo)
- **Estrategia de ramificación:** GitHub Flow simplificado (rama principal main para versiones estables + ramas de características feature/nombre-funcionalidad + rama develop para integración continua)
- **Buenas prácticas aplicadas:**
    - Commits atómicos y con mensajes claros en español (convención: "tipo: descripción corta" → ej. feat: añadir formulario de disponibilidad, fix: corregir conflicto horario vs disponibilidad, docs: actualizar README)
    - Uso de Pull Requests (aunque en proyecto individual, se crearon PRs auto-revisados para marcar hitos)
    - Etiquetas (tags) para versiones importantes: v0.1 (login + registro), v0.5 (gestión horarios), v1.0 (versión final evaluable)

Justificación: Git es el estándar para desarrollo de software. GitHub facilita el backup automático, historial completo y posible colaboración futura si el proyecto se abre a más desarrolladores.

b. Sistemas de integración continua

Herramienta utilizada: GitHub Actions (CI/CD nativo de GitHub)

- **Workflow principal:**.github/workflows/flutter_ci.yml
    - Ejecutado automáticamente en cada push y pull request a main y develop
    - Pasos implementados:
        1. Checkout del código
        2. Instalación de Flutter (versión estable canal)
        3. flutter pub get
        4. Análisis estático con flutter analyze
        5. Formateo de código con dart format --set-exit-if-changed .
        6. Ejecución de pruebas unitarias: flutter test
        7. (Opcional) Build APK de debug para validación rápida

Justificación: GitHub Actions es gratuito para repositorios privados en planes educativos y personales (2026), tiene excelente integración con Flutter y permite detectar errores de compilación, estilo y tests de forma inmediata. Aunque no se implementó despliegue automático a Firebase App Distribution o Play Store (por ser proyecto académico), el pipeline está preparado para ello.

c. Gestión de pruebas

Se han realizado pruebas de varios tipos durante todo el ciclo de desarrollo, aunque por tratarse de un proyecto individual, el enfoque ha sido práctico y centrado en calidad.

Tipos de pruebas realizadas y ejemplos concretos:

1. **Pruebas unitarias** (nivel más bajo – código Dart puro)
    - Herramienta: paquete test de Flutter/Dart
    - Cobertura: pruebas para modelos

2. **Pruebas de integración**
    - Realizadas manualmente + semi-automatizadas
    - Ejemplos:
        - Login/registro con cuentas de prueba → verificación de creación en Auth y Firestore
        - Guardado y carga de horario → comprobar que los mapas horarioFijo y disponibilidad se persisten y recuperan correctamente
        - Simulación de conflictos (añadir clase en hora marcada como disponible) → validación de SnackBar de advertencia
3. **Pruebas de usuario / funcionales**
    - Realizadas en dispositivo físico Android (Samsung A52) y emulador iOS (Xcode)
    - Escenarios probados exhaustivamente:
        - Flujo completo profesor: login → ver/editar horario → marcar disponibilidad → guardar → cerrar y reabrir app (persistencia)
        - Flujo de conflicto: intentar añadir clase en hora ya disponible → mensaje y opción de eliminar disponibilidad
        - Pruebas de responsividad: diferentes tamaños de pantalla (teléfono pequeño, tablet, foldable)
        - Pruebas multilingüe: cambio de idioma del dispositivo → comprobación de fechas y textos
4. **Pruebas de rendimiento**
    - Herramienta: Flutter DevTools (Performance tab)
    - Resultados:
        - Carga inicial de pantalla de horarios < 800 ms en dispositivo medio
        - Actualizaciones en tiempo real (Firestore snapshots) sin lag perceptible
        - Memoria estable < 120 MB en uso normal

5. **Pruebas de escalabilidad**
    - No se realizaron pruebas con miles de usuarios
    - Simulación manual: creación de 20 usuarios de prueba + 50 entradas de horario cada uno → Firestore responde correctamente (latencia < 200 ms)
    - Límite conocido: Firestore gratis permite 1 escritura/segundo por documento → suficiente para uso educativo real

9. Conclusiones

a. Conclusiones sobre el trabajo realizado

El proyecto SustiHorario cumple con sus objetivos iniciales, demostrando la viabilidad de una aplicación móvil para la gestión inteligente de sustituciones docentes en centros educativos.

Desarrollada íntegramente en Flutter con Dart y Firebase como backend serverless, la app permite a profesores gestionar horarios fijos y disponibilidades, solicitar bajas, y visualizar guardias asignadas, mientras coordinadores y administradores acceden a paneles específicos para asignaciones equitativas y gestión de usuarios/centros.

b. Conclusiones personales

Como alumno de Desarrollo de Aplicaciones Multiplataforma (DAM) en el ciclo 2025-2026, este proyecto ha sido transformador. Aprendí a dominar Flutter/Dart en profundidad, integrando Firebase.

Este trabajo no solo valida mi capacidad para proyectos reales, sino cumple con los requisitos para aprobar el curso.

c. Posibles ampliaciones y mejoras

- **IA para predicciones:** Integrar IA para prever ausencias por patrones estacionales/históricos, optimizando asignaciones
- **Notificaciones push completas:** Para alertas urgentes.
- **Accesibilidad/UX:** Soporte voz (TalkBack/VoiceOver), temas oscuro/claro nativo.

10. Bibliografía (comentada)

a. Libros, artículos y apuntes

- **Apuntes DAM - IES Serra Perenixsa (Torrent, Valencia, 2025-2026).** Clases de Desarrollo de interfaces - SONIA CARRILERO.

b. Direcciones web

- **Documentación oficial Flutter:** https://docs.flutter.dev/
- **Firebase for Flutter:** https://firebase.google.com/docs/flutter
- **Udemy - Flutter tu Guía Completa de Desarrollo para IOS y Android:** https://elhacker.info/Cursos/Udemy%20-%20Flutter%20tu%20Guía%20Completa%20de%20Desarrollo%20para%20IOS%20y%20Android/
- **GHC es el Generador de Horarios de tu Centro de enseñanza** https://www.penalara.com/es/?gad_source=1&gad_campaignid=85053201&gclid=Cj0KCQiApfjKBhC0ARIsAMiR_IvHBpwMTjAJNf5EE9AXsAHXwSQVQnfK8xrPsoHbO0wJ0RqY3GVcj74aAo8LEALw_wcB
- **WebUntis** https://webuntis.com
- **Borrador-instrucciones-secundaria-castellano:** https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjluMLd7peSAxWtMvsDHSyHLgwQFnoECCAQAw&url=https%3A%2F%2Fanpecomunidadvalenciana.es%2FopenFile.php%3Flink%3Dnotices%2Fatt%2F4%2Fborrador-instrucciones-secundaria-castellano_t1717774521_4_1.docx%23%3A~%3Atext%3DLa%2520parte%2520lectiva%2520de%2520la%2Cprevistas%2520en%2520la%2520normativa%2520vigente.&usg=AOvVaw15NsOQ6S_h6WesS7jxjeRY&opi=89978449
- **GUÍA DE HORARIOS 2023-2024:** https://documentos.anpecomunidadvalenciana.es/GUIAS/GuiaHorarios25-26/#page/16