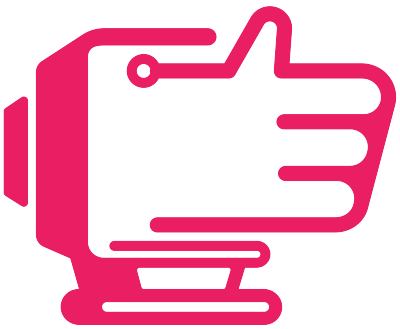
MANUAL TÉCNICO

TODO SOBRE EL DESARROLLO DE LA APLICACIÓN

Bienescript

Aplicación Educativa de Programación



DESARROLLADA POR:

CASTRO AGUILAR EDER JOEL

MARTÍNEZ PEÑAFIEL ADRIÁN FERNANDO

VALENCIA OROPEZA ANGEL YAHIR

Versión 1.0

18 de junio de 2025

Desarrollado con Flutter y Dart

Integración con Supabase y Google Gemini AI

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Bienescript es una aplicación móvil educativa desarrollada en Flutter para aprender programación de forma interactiva con asistente IA integrado. Ofrece lecciones colaborativas desde nivel básico hasta avanzado en múltiples lenguajes.

Características: Aprendizaje interactivo con videos, asistente AI (Google Gemini), múltiples lenguajes (SQL, Python, Java, HTML/CSS, Go, C++), gestión de usuarios, multiplataforma y seguimiento de progreso.

1.1 Tecnologías

Frontend: Flutter SDK ^{3},8,1+$Dart, desarrollomultiplataforma, UIexpresivaconwidgets, rendimientonativo.Gestióndees$
Backend: Google Gemini API para IA conversacional, NestJS (Node.js + TypeScript) para APIs REST escalables.

1.1.1 Stack Tecnológico Detallado

Categoría	**Tecnología**	**Versión/Descripción**
Framework Principal	Flutter	SDK ^{3},8,1
Lenguaje Frontend	Dart	^{3},8,1
Backend Framework	NestJS	Node.js + TypeScript
Base de Datos	Supabase	PostgreSQL + Auth + Storage
Inteligencia Artificial	Google Gemini	API conversacional
Gestión de Estado	Riverpod	Con generación de código
Reproductor de Video	Video Player	Componente personalizado
HTTP Client	Dio	Manejo de peticiones
Repositorio	GitHub	Control de versiones
Despliegue Backend	Render	Hosting del API

Cuadro 1.1: Stack Tecnológico Completo de Bienescript

Capítulo 2

Arquitectura del Sistema

2.1 Patrón Clean Architecture

Bienescript implementa Clean Architecture con separación clara de responsabilidades:

2.1.1 Capas de la Arquitectura

1. **Presentación**: Screens, widgets y providers de Riverpod
2. **Configuración**: Servicios externos como Gemini AI y router
3. **Datos**: Modelos de datos y contenido estático

2.1.2 Estructura de Directorios

- `lib/config/` - Configuraciones de Gemini AI y router
- `lib/data/` - Modelos de datos y contenido estático
- `lib/presentation/` - Providers, screens y widgets
- `main.dart` - Punto de entrada de la aplicación

2.2 Gestión de Estado con Riverpod

2.2.1 Providers Principales

Provider	**Responsabilidad**
ChatWithContext	Manejo de conversaciones con IA
UserProvider	Estado de autenticación y perfil
IsGeminiWriting	Estado de carga de respuestas AI

Cuadro 2.1: Providers del Sistema

2.3 Componentes Principales

2.3.1 Integración con Gemini AI

- **Modelo**: gemini-pro para conversaciones avanzadas
- **Streaming**: Respuestas en tiempo real
- **Contexto**: Integración con información de cursos
- **Multimedia**: Soporte para texto e imágenes

2.3.2 Sistema de Video

- Reproductor personalizado con controles nativos
- Modo pantalla completa para experiencia inmersiva
- Seguimiento automático de progreso de visualización
- Diseño responsivo adaptable a diferentes pantallas
- Marcado automático de lecciones completadas

2.3.3 Sistema de Autenticación

Supabase proporciona autenticación robusta con email/password, almacenamiento seguro de perfiles, gestión de sesiones y políticas de acceso granulares.

Capítulo 3

Pantallas y Funcionalidades

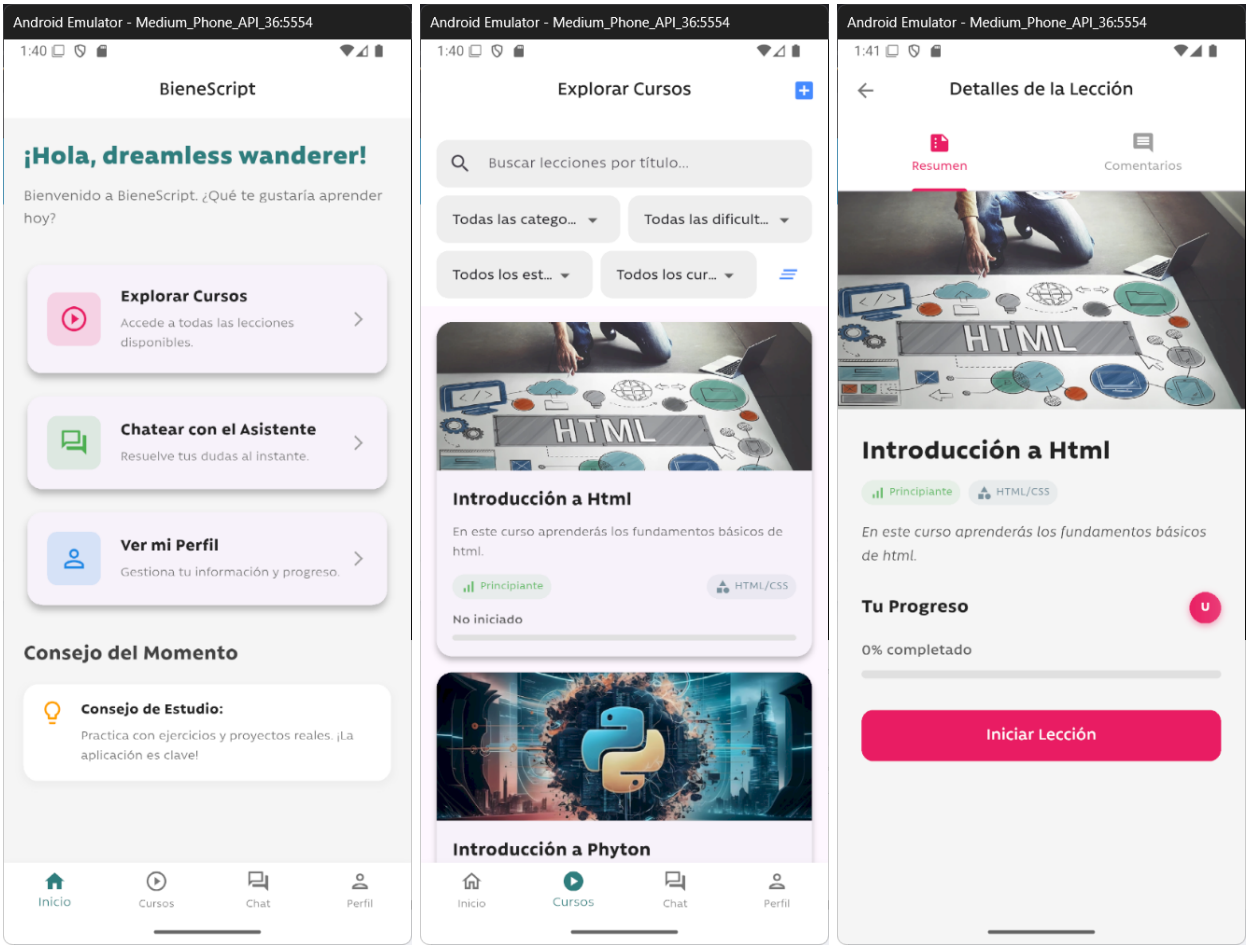


Figura 3.1: Dashboard, Explorador de Cursos y Detalle de Lección

3.1 Dashboard Principal

La pantalla de inicio proporciona acceso rápido a todas las funcionalidades principales:

- **Saludo Personalizado**: Mensaje de bienvenida dinámico con el nombre del usuario
- **Cards de Navegación**: Acceso directo a Explorar Cursos, Chat con Asistente y Perfil
- **Consejo del Momento**: Tips dinámicos de estudio y programación
- **Progreso Reciente**: Vista rápida del avance en cursos activos

3.2 Sistema de Cursos

3.2.1 Categorías Disponibles

Categoría	Lenguajes	Niveles
Web Frontend	HTML/CSS, JavaScript	Principiante, Intermedio, Avanzado
Backend	Python, Java, Go	Principiante, Intermedio, Avanzado
Base de Datos	SQL	Principiante, Intermedio, Avanzado
Sistemas	C++, C	Principiante, Intermedio, Avanzado

Cuadro 3.1: Categorías de Cursos Disponibles

3.2.2 Estructura de Lecciones

Cada lección incluye múltiples componentes educativos:

1. **Contenido Teórico**: Explicaciones detalladas con ejemplos prácticos
2. **Videos de Apoyo**: Material audiovisual complementario
3. **Sistema de Comentarios**: Interacción entre estudiantes y educadores
4. **Seguimiento de Progreso**: Porcentaje de completado con métricas detalladas
5. **Ejercicios Prácticos**: Actividades hands-on para reforzar conceptos

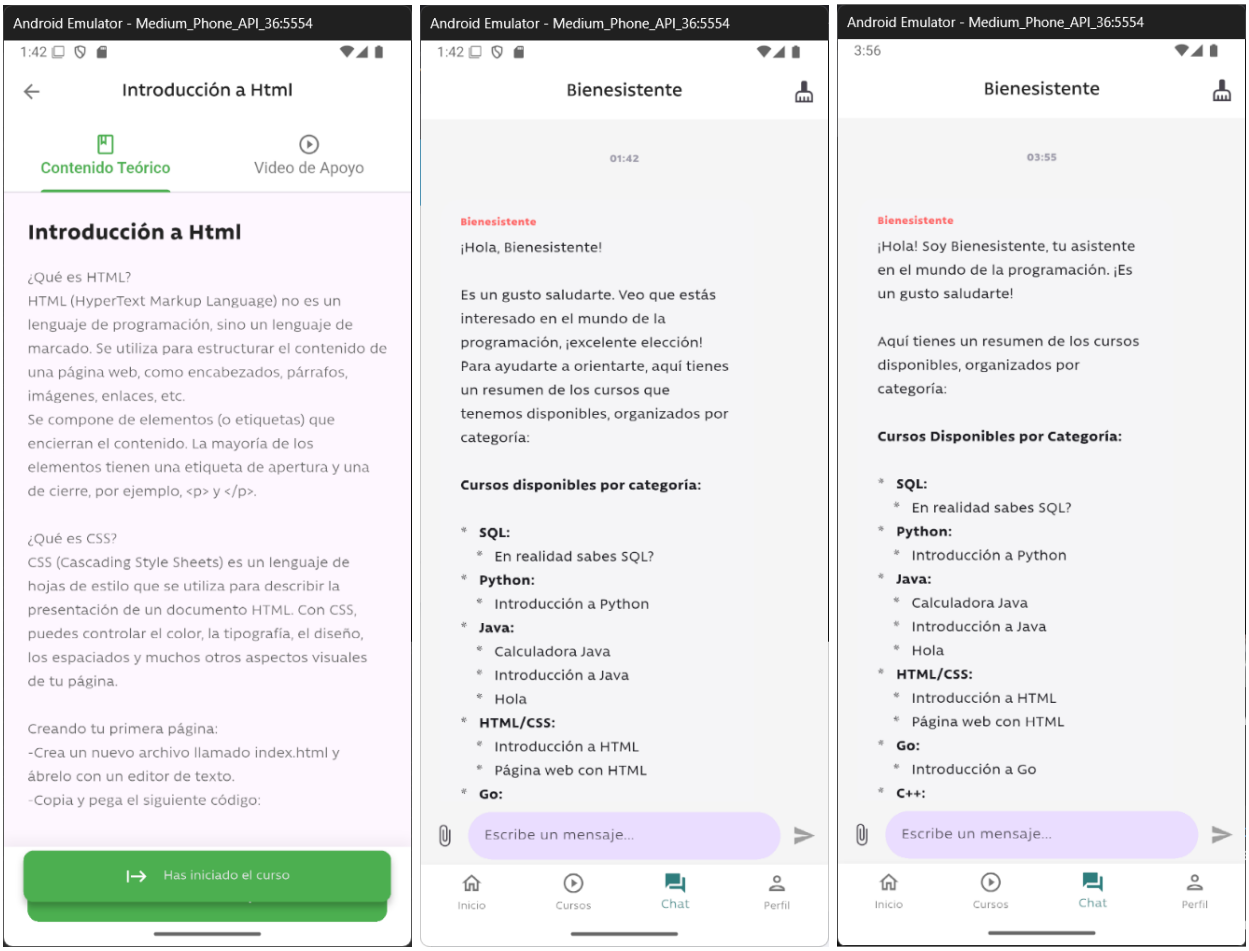


Figura 3.2: Contenido Teórico y Chat IA

3.3 Chat con Benesistente

El asistente inteligente Benesistente representa el núcleo de la experiencia de aprendizaje personalizado:

3.3.1 Funcionalidades Avanzadas

- **Contexto de Cursos**: El asistente conoce el contenido específico de cada lección
- **Respuestas Streaming**: Generación de texto en tiempo real para mejor UX
- **Historial Persistente**: Conversaciones guardadas para continuidad del aprendizaje
- **Soporte Multimedia**: Análisis de imágenes y código enviado por usuarios
- **Recomendaciones Personalizadas**: Sugerencias basadas en progreso individual

3.3.2 Mensajes Automáticos

Al acceder al chat, Benesistente proporciona automáticamente:

1. Resumen de cursos disponibles organizados por categoría
2. Estado actual del progreso del usuario
3. Sugerencias de próximos pasos en el aprendizaje
4. Respuestas a preguntas frecuentes sobre cada lenguaje

Capítulo 4

Configuración y Desarrollo

4.1 Requisitos del Sistema

4.1.1 Entorno de Desarrollo

- **Flutter SDK**: Versión ^{3},8,1$osuperior$
- **Dart SDK**: Versión ^{3},8,1$incluidaconFlutter$
- **IDE**: Android Studio, VS Code o IntelliJ IDEA
- **Control de Versiones**: Git para gestión de código fuente
- **Herramientas Adicionales**: Android SDK, Xcode (para iOS)

4.1.2 Variables de Entorno

La aplicación requiere las siguientes variables de entorno en archivo `.env`:

Variable	Descripción
SUPABASE_URL	URL del proyecto Supabase
SUPABASE_ANON_KEY	Clave anónima de Supabase
GEMINI_API_KEY	Clave de API de Google Gemini

Cuadro 4.1: Variables de Entorno Requeridas

4.2 Dependencias del Proyecto

4.2.1 Dependencias de Producción

Package	Función
flutter_riverpod	Gestión de estado reactivo
supabase_flutter	Cliente para Supabase
dio	Cliente HTTP para APIs
video_player	Reproductor de video
flutter_chat_ui	Interfaz de chat
flutter_dotenv	Manejo de variables de entorno

Cuadro 4.2: Dependencias Principales

4.2.2 Dependencias de Desarrollo

- **riverpod_generator**: Generación automática de código para providers
- **build_runner**: Herramienta de build para generación de código
- **flutter_launcher_icons**: Generación de íconos de aplicación
- **flutter_lints**: Reglas de lint para código limpio

4.3 Proceso de Build

4.3.1 Comandos de Desarrollo

1. `flutter packages pub run build_runner build` - Generar código de providers
2. `flutter packages pub run flutter_launcher_icons:main` - Generar íconos
3. `flutter run` - Ejecutar en modo desarrollo
4. `flutter test` - Ejecutar tests unitarios

4.3.2 Build de Producción

- **Android**: `flutter build apk --release`
- **iOS**: `flutter build ios --release`
- **Web**: `flutter build web --release`
- **Desktop**: `flutter build windows/macos/linux --release`

Capítulo 5

Backend y Base de Datos

5.1 Supabase como Backend

Supabase funciona como Backend-as-a-Service (BaaS) proporcionando una solución completa:

5.1.1 Componentes de Supabase

- **PostgreSQL**: Base de datos relacional robusta y escalable
- **Auth**: Sistema de autenticación integrado con múltiples proveedores
- **Storage**: Almacenamiento de objetos para archivos multimedia
- **Edge Functions**: Funciones serverless para lógica backend
- **Real-time**: Subscripciones en tiempo real a cambios de datos

5.1.2 Estructura de la Base de Datos

Tabla	Descripción y Campos Principales
users	Información de usuarios: id, email, name, role, created_at
courses	Catálogo de cursos: id, title, description, category, difficulty, instructor_id
lessons	Lecciones: id, course_id, title, content, video_url, order_index
user_progress	Progreso: user_id, lesson_id, completed_at, progress_percentage
comments	Comentarios: id, lesson_id, user_id, content, created_at, parent_id

Cuadro 5.1: Estructura de Tablas de la Base de Datos

5.1.3 Políticas de Seguridad

Supabase implementa Row Level Security (RLS) para proteger los datos:

- **Usuarios**: Solo pueden ver y modificar su propia información
- **Progreso**: Acceso restringido al progreso del usuario autenticado
- **Cursos**: Lectura pública, escritura solo para educadores
- **Comentarios**: Usuarios pueden crear y modificar sus propios comentarios

5.2 Integración con Google Gemini

5.2.1 Configuración del Modelo

Parámetro	**Valor**
Modelo	gemini-pro
Temperatura	0.7
Máximo de Tokens	1024
Streaming	Habilitado

Cuadro 5.2: Configuración del Modelo Gemini

5.2.2 Funcionalidades de IA

- **Conversación Contextual**: Mantiene el contexto de la conversación
- **Análisis de Código**: Puede revisar y explicar fragmentos de código
- **Generación de Ejercicios**: Crea ejercicios personalizados
- **Detección de Errores**: Identifica y explica errores de programación
- **Recomendaciones**: Sugiere mejores prácticas y optimizaciones

Capítulo 6

Usuarios, Diseño y Seguridad

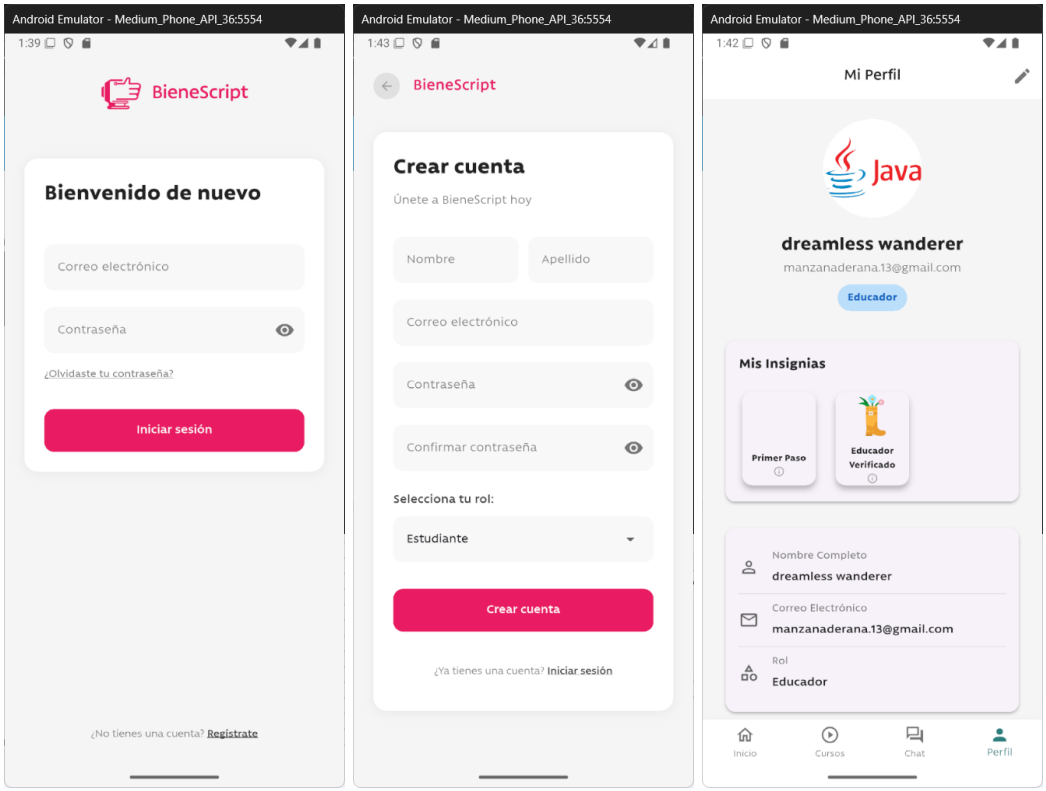


Figura 6.1: Login, Registro y Perfil

Auth: Login/registro email/contraseña, validación, recuperación, roles (Estudiante/Educador).
Perfiles: Avatar personalizable, insignias, estadísticas, edición de información.

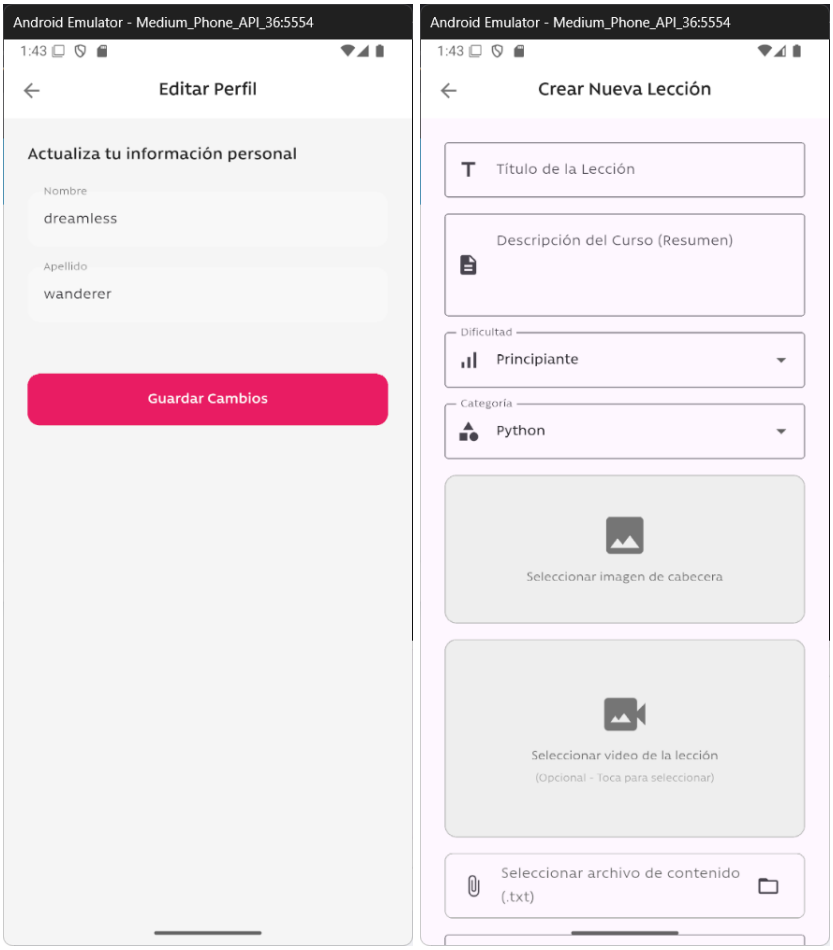


Figura 6.2: Edición de Perfil y Panel Admin

Admin: Educadores crean lecciones con título, descripción, categoría, dificultad, imagen, video, contenido.

6.1 Sistema de Autenticación

6.1.1 Flujo de Autenticación

El sistema de autenticación implementa un flujo completo y seguro:

1. **Registro**: Formulario con validación en tiempo real
2. **Verificación de Email**: Confirmación via enlace enviado por email
3. **Login**: Autenticación con email y contraseña
4. **Recuperación**: Reset de contraseña via email seguro
5. **Gestión de Sesión**: Tokens JWT con renovación automática

6.1.2 Roles de Usuario

Rol	Permisos
Estudiante	Ver cursos, completar lecciones, usar chat IA, comentar
Educador	Crear lecciones, moderar comentarios, analytics
Administrador	Gestión completa de usuarios y contenido

Cuadro 6.1: Roles y Permisos del Sistema

6.2 Diseño y Experiencia de Usuario

6.2.1 Sistema de Diseño

Elemento	Especificación	Uso
Color Primario	#EC4899 (Pink)	Botones principales, títulos
Color Secundario	#3B82F6 (Blue)	Enlaces, elementos secundarios
Fondo	#F8F9FA (Light Gray)	Fondo general de la app
Tipografía	Rawson (Múltiples pesos)	Toda la interfaz de usuario

Cuadro 6.2: Especificaciones del Sistema de Diseño

6.2.2 Navegación

La aplicación utiliza **Bottom Navigation** como patrón principal con cuatro secciones:

- **Inicio**: Dashboard con acceso rápido y progreso reciente
- **Cursos**: Exploración y gestión de contenido educativo
- **Chat**: Asistente IA conversacional contextual
- **Perfil**: Gestión de usuario y configuraciones

6.2.3 Flujo de Usuario

El flujo principal del usuario está optimizado para el aprendizaje efectivo:
Registro → Configuración de Perfil → Exploración de Cursos → Selección de Lección → Contenido Teórico → Video de Apoyo → Chat IA → Comentarios → Progreso → Siguiente Lección

6.3 Seguridad y Privacidad

6.3.1 Medidas de Seguridad Implementadas

- **Autenticación JWT**: Tokens seguros con expiración automática
- **Row Level Security**: Políticas granulares de acceso a datos
- **Validación de Entrada**: RegExp y sanitización de todos los inputs
- **Variables de Entorno**: Claves API protegidas y no expuestas
- **HTTPS**: Comunicación encriptada en todas las conexiones
- **Rate Limiting**: Protección contra ataques de fuerza bruta

6.3.2 Privacidad de Datos

- **Datos Mínimos**: Recolección solo de información necesaria
- **Consentimiento**: Opt-in explícito para features opcionales
- **Anonimización**: Datos de analytics sin identificación personal
- **Derecho al Olvido**: Capacidad de eliminar cuenta y datos

6.4 Optimización de Rendimiento

6.4.1 Frontend

- **Riverpod**: Rebuilds optimizados solo cuando el estado cambia
- **Lazy Loading**: Carga bajo demanda de imágenes y videos
- **Caché Local**: Almacenamiento temporal de datos frecuentes
- **Pagination**: Carga incremental de listas extensas
- **Image Optimization**: Múltiples resoluciones según dispositivo

6.4.2 Backend

- **Database Indexing**: Índices optimizados para consultas frecuentes
- **Connection Pooling**: Gestión eficiente de conexiones a DB
- **CDN**: Distribución de contenido estático desde edge locations
- **Compresión**: Gzip/Brotli para reducir tamaño de respuestas

Capítulo 7

Costos, Equipo y Conclusiones

7.1 Análisis Detallado de Costos

La aplicación Bienescript ha sido desarrollada con un enfoque de costos optimizados, ideal para equipos pequeños y proyectos en fase inicial.

7.1.1 Estructura de Costos

Servicio	**Plan Gratuito**	**Plan Escalado**
Flutter + GitHub	$0	$0
NestJS + Render	$0	$7/mes
Supabase	$0 (500MB, 50k requests)	$25/mes (8GB, 100k users)
Gemini API	$0 (límite de tokens)	Variable por uso
Google Play Store	$25 (único)	$25 (único)
Total Inicial	**$25**	**$25**
Total Mensual	**$0**	**$32**

Cuadro 7.1: Desglose Completo de Costos

7.1.2 Proyección de Escalabilidad

- **0-1,000 usuarios**: Completamente gratuito excepto publicación
- **1,000-10,000 usuarios**: $7-32/mes según uso de Supabase
- **10,000+ usuarios**: Costos variables según tráfico y storage

7.1.3 Ventajas del Modelo de Costos

- **Bajo Riesgo Inicial**: Inversión mínima para validar el producto
- **Escalabilidad Gradual**: Costos crecen con el éxito del proyecto
- **Sin Compromisos a Largo Plazo**: Planes mensuales flexibles
- **Infraestructura Robusta**: Tecnologías enterprise sin costos enterprise

7.2 Equipo de Desarrollo

7.2.1 Roles y Responsabilidades

Desarrollador	**Rol Principal**	**Especialización**
Castro Aguilar Eder Joel	Full-Stack Developer	Flutter + Backend Integration
Martínez Peñafiel Adrián Fernando	Frontend Developer	UI/UX + Flutter Widgets
Valencia Oropeza Angel Yahir	Backend Developer	NestJS + Supabase + APIs

Cuadro 7.2: Estructura del Equipo de Desarrollo

7.2.2 Metodología de Trabajo

El equipo implementó una metodología ágil adaptada para equipos pequeños:

- **Sprints de 2 semanas**: Iteraciones rápidas con entregas frecuentes
- **Daily Standups**: Sincronización diaria del progreso
- **Code Reviews**: Revisión cruzada de código para calidad
- **Pair Programming**: Colaboración en features complejas
- **Continuous Integration**: Automatización de tests y builds

7.3 Conclusiones

7.3.1 Logros Técnicos

Bienescript representa una implementación exitosa de tecnologías modernas para educación:

- **Arquitectura Escalable**: Clean Architecture permite crecimiento organizado
- **IA Integrada**: Gemini proporciona asistencia personalizada e inteligente
- **Multiplataforma**: Una base de código para todos los dispositivos
- **Backend Robusto**: Supabase ofrece escalabilidad enterprise
- **UX Moderna**: Diseño intuitivo optimizado para aprendizaje

7.3.2 Impacto Educativo

La aplicación contribuye significativamente al ecosistema educativo tecnológico:

- **Democratización**: Acceso gratuito a educación de programación
- **Personalización**: IA adapta el contenido al ritmo individual
- **Comunidad**: Plataforma colaborativa entre estudiantes y educadores
- **Innovación**: Integra las últimas tecnologías en educación

7.3.3 Reflexión Final

El equipo de tres desarrolladores logró crear una plataforma completa que combina desarrollo móvil moderno, inteligencia artificial y principios educativos sólidos. Bienescript demuestra que es posible construir aplicaciones de calidad enterprise con recursos limitados, utilizando tecnologías open-source y servicios escalables.

La arquitectura implementada no solo resuelve los desafíos actuales sino que está preparada para el crecimiento futuro, manteniendo la calidad técnica y la experiencia de usuario como prioridades fundamentales.