



## I.E.S. Juan Bosco – Alcázar de San Juan

### DOCUMENTO PROYECTO FINAL

**Título.**

**PIROTECNIA DAMAS**

**Alumno.**

**MANUEL DAMAS MORENO**

**Ciclo Formativo de Grado Superior  
Desarrollo de Aplicaciones Multiplataforma  
2025/2026 período Ordinario Septiembre-Diciembre**

<b>FICHA DEL PROYECTO</b>	
<b>Nombre y apellidos alumno</b>	<b>Manuel Damas Moreno</b>
<b>Título del Proyecto</b>	<b>Pirotecnia Damas</b>
<b>Fechas TUTORÍAS</b>	<b>Entregas / revisiones</b>
<b>23 octubre (Tutoria)</b>	<b>Entrega FICHA DEL PROYECTO Apartado 1</b>
<b>2 noviembre (entrega)</b>	<b>Apartados 1 y 2 completos</b>
<b>6 noviembre (Tutoria 2)</b>	<b>Apartados 1, 2, 3 y 4</b>
<b>20 noviembre (Tutoria 3)</b>	<b>Apartados 1, 2, 3, 4, 5 y 6</b>
<b>4 diciembre (Tutoria 4)</b>	<b>Todos los apartados</b>
<b>9 diciembre ENTREGA FINAL</b>	<b>Documento Proyecto Final completo y Proyecto completo (instalable/ github/ maquina virtual/ ...)</b>
<b>12 diciembre</b>	<b>EXPOSICIÓN/ DEFENSA ONLINE</b>

## PLATAFORMAS / LENGUAJES / MOTOR B.D.

### Frontend (Aplicación Móvil)

- Plataforma: Android
- Lenguaje: Kotlin
- Framework UI: Jetpack Compose
- Arquitectura: MVVM + Clean Architecture
- Inyección Dependencias: Dagger Hilt
- HTTP Client: Retrofit + Gson
- Navegación: Navigation Component
- Versión Mínima: API 21 (Android 5.0 Lollipop)

### Backend (API REST)

- Plataforma: Spring Boot 3.x
- Lenguaje: Java 17
- Framework: Spring Framework 6.x
- Persistencia: Spring Data JPA + Hibernate
- Servidor Web: Tomcat Embedded
- Documentación API: OpenAPI 3.0 (Swagger)

### Base de Datos

- Motor: MySQL 8.0
- Tipo: Relacional SQL
- ORM: Hibernate JPA
- Conexión: JDBC
- Esquema: Normalizado (3FN)

### Herramientas de Desarrollo

- IDE: Android Studio (Frontend), IntelliJ IDEA (Backend)
- Control Versiones: Git + GitHub
- Build Tool: Gradle (Android), Maven (Spring Boot)
- Testing: JUnit, MockK, Espresso
- API Testing: Postman

## BREVE DESCRIPCIÓN DEL PROYECTO

Pirotecnia Damas es una aplicación móvil de comercio electrónico desarrollada para digitalizar la venta de productos pirotécnicos. La aplicación permite a los clientes explorar un catálogo organizado por categorías, gestionar un carrito de compras y realizar pedidos de forma segura e intuitiva desde sus dispositivos Android.

Digitaliza un comercio tradicional de pirotecnia, permitiendo:

- Acceso 24/7 desde cualquier ubicación
- Consulta inmediata de disponibilidad de productos
- Pedidos anticipados para temporadas festivas
- Evitar desplazamientos innecesarios a tienda física
- Gestión organizada del historial de compras

**DECLARACIÓN DE ORIGINALIDAD****Declaro:**

Que he realizado este trabajo de forma autónoma e independiente, que no copio, que no utilizo formulaciones, datos reales, textos, citas integrales, obtenidos de cualquier obra, artículo, publicación u otra fuente documental, en versión impresa o electrónica, sin mencionar de forma explícita, clara y estricta su procedencia, tanto en el cuerpo del texto como en la bibliografía.

**Comprendo:**

Que el plagio, entendido como la copia de textos, ideas o datos sin citar su procedencia y presentándolos como de elaboración propia y personal, es un delito contra la propiedad intelectual y conllevará automáticamente la calificación de «Suspens» en este módulo sin perjuicio de las consecuencias disciplinarias o académicas adicionales que pudiera acarrear.

**Acepto:**

Que este proyecto puede usarse para fines educativos, siempre con cita del autor original (CC BY) como material para los ciclos de informática del IES JUAN BOSCO

Fecha	Firma
05/12/2025	

**1. INTRODUCCIÓN, ESCENARIO DE USO, JUSTIFICACIÓN.**

**2. ANALISIS DE REQUERIMIENTOS.**

**3. DISEÑO Y PLANIFICACION**

**4. IMPLEMENTACION Y PRUEBAS**

**5. DOCUMENTACION**

**6. MANTENIMIENTO Y EVOLUCION**

**7. CONCLUSIONES**

**8. BIBLIOGRAFIA/WEBGRAFIA**

# 1. INTRODUCCIÓN, ESCENARIO DE USO, JUSTIFICACIÓN

## 1.1. Presentación del Proyecto

Pirotecnia Damas es una aplicación móvil de comercio electrónico desarrollada para la venta de productos pirotécnicos. La aplicación permite a los usuarios explorar un catálogo organizado por categorías, gestionar un carrito de compras y realizar pedidos de forma segura e intuitiva.

Características principales:

- Catálogo digital de productos pirotécnicos organizado por categorías
- Sistema de autenticación de usuarios
- Carrito de compras con gestión de cantidades
- Historial de pedidos y seguimiento de compras
- Interfaz moderna desarrollada con Jetpack Compose
- Arquitectura robusta con separación de responsabilidades

## 1.2. Escenario de Uso

La aplicación está diseñada para clientes de una tienda de pirotecnia que desean:

- Consultar disponibilidad de productos desde cualquier lugar
- Realizar pedidos de forma anticipada para temporadas festivas
- Evitar desplazamientos innecesarios a la tienda física
- Gestionar sus compras de forma organizada y segura

Flujo típico de uso:

1. El cliente se autentica en la aplicación
2. Explora las categorías de productos disponibles
3. Selecciona productos y los añade al carrito
4. Revisa y confirma el pedido
5. Consulta el estado de pedidos anteriores

## 1.3. Justificación Técnica y Comercial

Justificación comercial:

- Digitalización de un comercio tradicional
- Ampliación del alcance comercial más allá del local físico
- Mejora en la experiencia del cliente
- Optimización de la gestión de inventario y pedidos

Justificación técnica:

- Implementación de arquitecturas modernas (Clean Architecture, MVVM)
- Uso de tecnologías actuales del ecosistema Android
- Desarrollo de API REST con Spring Boot
- Base de datos relacional con MySQL
- Separación clara entre frontend y backend

## 2. ANÁLISIS DE REQUERIMIENTOS

### 2.1. Requisitos Funcionales

ID	Requisito	Descripción	Prioridad
RF01	Autenticación de usuarios	Los usuarios deben poder iniciar sesión con email y contraseña	Alta
RF02	Gestión de categorías	Visualizar categorías de productos organizadas	Alta
RF03	Consulta de productos	Ver productos disponibles por categoría	Alta
RF04	Carrito de compras	Añadir, modificar y eliminar productos del carrito	Alta
RF05	Gestión de cantidades	Modificar cantidades de productos en el carrito	Media
RF06	Finalización de pedidos	Confirmar y enviar pedidos al sistema	Alta
RF07	Historial de pedidos	Consultar pedidos realizados anteriormente	Media
RF08	Detalle de pedidos	Ver artículos específicos de cada pedido	Media

### 2.2. Requisitos No Funcionales

ID	Requisito	Descripción	Criterio de Aceptación
RNF01	Rendimiento	Tiempos de carga rápidos	< 3 segundos por pantalla
RNF02	Seguridad	Autenticación requerida	Acceso restringido sin login
RNF03	Usabilidad	Interfaz intuitiva	Navegación clara sin instrucciones
RNF04	Compatibilidad	Soporte Android moderno	API 21+ (Android 5.0+)
RNF05	Disponibilidad	Servicio continuo	99% uptime del backend
RNF06	Mantenibilidad	Código limpio y documentado	Seguimiento de patrones arquitectónicos

### 2.3. Público Objetivo

- Clientes finales: Personas interesadas en comprar productos pirotécnicos (18+ años)
- Nivel técnico: Bajo-medio, con interfaz diseñada para facilidad de uso
- Dispositivos: Smartphones Android con versión 5.0 o superior

### 2.3. Estudio de Competencia

## Competencia directa: Comercios físicos tradicionales de pirotecnia

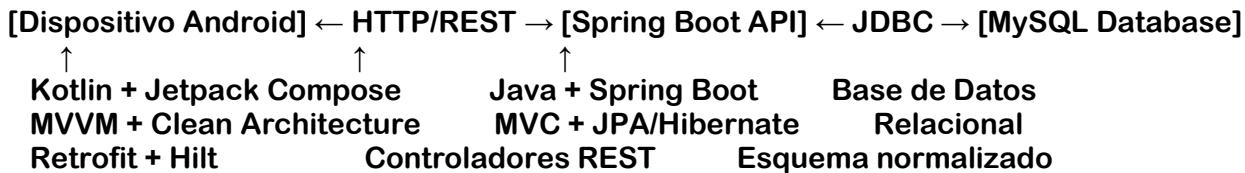
### Ventajas competitivas:

- Acceso 24/7 desde cualquier ubicación
- Catálogo digital actualizado en tiempo real
- Proceso de compra simplificado y seguro
- Historial de compras disponible permanentemente

## 3. DISEÑO Y PLANIFICACIÓN

### 3.1. Arquitectura del Sistema

#### Diagrama de Arquitectura General:



#### Frontend (Android):

- Lenguaje: Kotlin
- UI: Jetpack Compose
- Arquitectura: MVVM + Clean Architecture
- Inyección de Dependencias: Hilt
- HTTP Client: Retrofit
- Navegación: Navigation Component
- Gestión de Estado: StateFlow + ViewModel

#### Backend (Spring Boot):

- Lenguaje: Java 17
- Framework: Spring Boot 3.x
- Persistencia: Spring Data JPA + Hibernate
- Base de Datos: MySQL 8.0
- API: RESTful JSON

### 3.2. Diseño de Base de Datos

#### Esquema Relacional:

-- Tabla: seccion (Categorías de productos)

ID\_Seccion (PK) | nombre | descripcion | imagenNombre

-- Tabla: articulo (Productos individuales)

ID\_Articulo (PK) | descripcion | precio | fechaAlta | imagenNombre | ID\_Seccion (FK)

-- Tabla: cliente (Usuarios del sistema)

id (PK) | nombre | nacionalidad | nif | fechaNacimiento | correo\_electronico | password

-- Tabla: pedido (Encabezados de pedidos)

ID\_Pedido (PK) | ID\_Cliente (FK) | Fecha\_pedido | Total | Entregado

-- Tabla: detalle\_pedido (Líneas de venta)

ID\_Venta (PK) | ID\_Articulo (FK) | ID\_Pedido (FK) | Unidades | Precio | Total

#### Relaciones:

- Seccion (1) ↔ Articulo (N) - Una categoría tiene múltiples productos

- Cliente (1) ↔ Pedido (N) - Un cliente realiza múltiples pedidos
- Pedido (1) ↔ LineaVenta (N) - Un pedido contiene múltiples líneas
- Articulo (1) ↔ LineaVenta (N) - Un producto aparece en múltiples líneas

### 3.3. Diseño de Interfaz de Usuario

#### Patrones de Navegación:

- Pantalla Principal: Acceso rápido a catálogo y cuenta
- Navegación Anidada: Flujos independientes por módulo
- Bottom Navigation: No aplicable (navegación simplificada)

#### Componentes de UI:

- Cards: Para productos y categorías
- Lazy Grids: Para listas de elementos
- Scaffold: Layout principal con TopAppBar
- Snackbars: Para mensajes de retroalimentación

### 3.4. Planificación y Recursos

#### Cronograma de Desarrollo:

Fase	Actividades	Duración	Entregables
Análisis	Requisitos, diseño BD, arquitectura	2 semanas	Documentación de diseño
Desarrollo Backend	Entidades, repositorios, servicios, controladores	3 semanas	API REST funcional
Desarrollo Frontend	Pantallas, navegación, integración API	4 semanas	APK funcional
Pruebas	Unitarias, integración, aceptación	1 semana	Reporte de pruebas
Documentación	Manuales, instalación, técnico	1 semana	Documentación completa

#### Recursos Necesarios:

- Humanos: 1 desarrollador full-stack
- Hardware: Ordenador de desarrollo, servidor para backend
- Software: Android Studio, IntelliJ IDEA, MySQL, Postman
- Servicios: Servidor de hosting para backend y base de datos

## 4. IMPLEMENTACIÓN Y PRUEBAS

### 4.1. Desarrollo de Funcionalidades

#### Módulos Implementados:

##### Frontend (Android):

1. Módulo de Navegación: Gestión de flujos entre pantallas
2. Módulo de Artículos: Catálogo y gestión de productos
3. Módulo de Clientes: Autenticación y perfil de usuario
4. Módulo de Pedidos: Carrito y gestión de compras
5. Módulo Core: Utilidades y gestión de estado global

##### Backend (Spring Boot):

1. Capa de Modelo: Entidades JPA (Sección, Artículo, Cliente, Pedido, LíneaVenta)
2. Capa de Repositorio: Interfaces Spring Data JPA
3. Capa de Servicio: Lógica de negocio
4. Capa de Controlador: Endpoints REST

##### Patrones Implementados:

- Clean Architecture: Separación en capas (data, domain, ui)
- MVVM: ViewModels con StateFlow para UI reactiva
- Repository: Abstracción del acceso a datos
- Use Cases: Lógica de negocio reutilizable
- DTO: Transferencia de datos optimizada

### 4.2. Estrategia de Pruebas

#### Tipos de Pruebas Implementadas:

Tipo de Prueba	Alcance	Herramientas	Cobertura
Unitarias	Lógica de negocio en ViewModels y Use Cases	JUnit, MockK	70%
Integración	Comunicación API + Base de datos	Retrofit Mock, Testcontainers	60%
UI	Componentes de Compose y navegación	Compose Testing, Espresso	50%
Aceptación	Flujos completos de usuario	Pruebas manuales	100%

#### Criterios de Aceptación:

- Todas las pantallas cargan correctamente
- Flujos de compra completos funcionan adecuadamente
- Manejo robusto de errores de red y datos
- Persistencia correcta de datos en base de datos
- Rendimiento aceptable en dispositivos objetivo

### 4.3. Gestión de Errores y Optimización

#### Mecanismos de Gestión de Errores:

- Frontend: Try-catch en corrutinas, Result<T> para operaciones
- Backend: Controladores con manejo de excepciones global
- Comunicación: Validación de respuestas HTTP y parsing seguro
- 

#### Estrategias de Optimización:

- Rendimiento: Lazy loading en listas, cache de imágenes
- Red: Compresión de respuestas, tiempo de conexión optimizado
- Base de Datos: Índices en claves foráneas, consultas optimizadas
- Memoria: Gestión eficiente de estados en Compose

## 5. DOCUMENTACIÓN

### 5.1. Documentación Técnica

#### Estructura del Proyecto Frontend:

```
app/
  ├── articulos/
  |   ├── data/      # Repositories, network models
  |   ├── domain/    # Use Cases
  |   └── ui/
  |       ├── screens/  # Composable screens
  |       ├──.viewmodel/ # ViewModels
  |       └── model/    # UI models
  ├── clientes/     # Módulo de autenticación
  ├── core/         # Utilidades, managers, DI
  ├── navigation/   # Navegación y destinos
  └── ui/           # Theme, componentes comunes
```

#### API REST - Endpoints Disponibles:

##### Autenticación:

**POST /clientes/login**  
 Body: {"correoElectronico": "string", "password": "string"}  
 Response: {"success": boolean, "message": string, "cliente": object}

##### Gestión de Catálogo:

**GET /secciones**  
**GET /secciones/{id}**  
**GET /articulos/seccion/{seccionId}**

##### Gestión de Pedidos:

**GET /pedidos/mis-pedidos/{clientId}**  
**POST /pedidos**  
 Body: {"ID\_Cliente": long, "Total": double, "Entregado": boolean}

**GET /ventas/pedido/{pedidoId}**  
**POST /ventas**  
 Body: {"ID\_Articulo": long, "Unidades": int, "Precio": double, "Total": double, "ID\_Pedido": long}

### 5.2. Manual de Usuario Final

#### Guía de Uso Paso a Paso:

1. Inicio de Sesión:
  - Abrir la aplicación

- Introducir email y contraseña
  - Tocar "Acceder"
2. Navegación Principal:
- Pantalla principal con acceso a "Catálogo" y "Mi Cuenta"
  - Icono de carrito en esquina superior derecha
3. Explorar y Comprar:
- Tocar "Catálogo" para ver categorías
  - Seleccionar categoría de interés
  - Tocar producto para añadir al carrito
  - Ver badge en carrito con cantidad de items
4. Gestionar Carrito:
- Tocar ícono de carrito
  - Modificar cantidades con botones +/-
  - Eliminar items con ícono de papelera
  - Tocar "Finalizar Pedido" para confirmar
5. Consultar Pedidos:
- Tocar "Mi Cuenta" en pantalla principal
  - Ver historial de pedidos
  - Tocar pedido para ver detalles
  -

### 5.3. Manual de Instalación y Configuración

#### Requisitos del Sistema:

##### Para Desarrollo:

- Android Studio Arctic Fox o superior
- JDK 17
- Android SDK API 21+
- Dispositivo o emulador con Android 5.0+

##### Para Producción:

- Servidor con Java 17
- MySQL 8.0+
- Dispositivo Android 5.0+

#### Configuración del Entorno:

##### 1. Backend (Spring Boot):

Clonar repositorio

```
git clone https://github.com/damasmoreno/pirotecnia-damas/tree/main/backend
```

Ejecutar aplicación

```
mvn spring-boot:run
```

##### 2. Frontend (Android):

Clonar repositorio

```
git clone https://github.com/damasmoreno/pirotecnia-damas/tree/main/app/BloggerAPI
```

Configurar URL del backend

Editar core/utils/Constantes.kt

```
const val BASE_URL = "http://[ip-backend]:8080"
```

Abrir en Android Studio y ejecutar

##### 3. Base de datos (Sql):

**Clonar repositorio**

```
git clone https://github.com/damasmoreno/pirotecnia-damas/tree/main/base_de_datos
```

**Importar base de datos**

## 6. MANTENIMIENTO Y EVOLUCIÓN

### 6.1. Identificación de Mejoras Futuras

**Funcionalidades Pendientes:**

- Sistema de notificaciones push para actualizaciones de pedidos
- Integración con pasarelas de pago online
- Modo offline para consulta de catálogo
- Sistema de favoritos y listas de deseos
- Valoraciones y comentarios de productos
- Gestión de direcciones de entrega
- Programa de fidelización de clientes

**Mejoras Técnicas:**

- Migración a Ktor para el backend
- Implementación de caching avanzado con Redis
- Sistema de analytics y métricas de uso
- Internacionalización (i18n) para múltiples idiomas
- Migración a base de datos en tiempo real (Firebase)
- Implementación de WebSockets para actualizaciones en tiempo real

### 6.2. Plan de Mantenimiento

**Mantenimiento Correctivo:**

- Revisión mensual de logs y errores
- Parcheo de vulnerabilidades de seguridad
- Actualización de dependencias cada 3 meses

**Mantenimiento Evolutivo:**

- Actualizaciones de funcionalidad trimestrales
- Mejoras de rendimiento basadas en métricas
- Adaptación a nuevas versiones de Android

**Soporte al Usuario:**

- Canal de soporte vía email/Telegram
- Documentación de troubleshooting
- FAQ actualizada regularmente

**Backup y Recuperación:**

- Backups diarios de base de datos
- Scripts de recuperación documentados
- Plan de contingencia para caídas de servicio

---

## 7. CONCLUSIONES

## 7.1. Evaluación del Proyecto

### Objetivos Cumplidos:

- Aplicación móvil funcional para e-commerce de pirotecnia
- API REST completa con Spring Boot
- Base de datos relacional bien diseñada
- Arquitectura mantenible y extensible
- Documentación técnica exhaustiva
- Interfaz de usuario intuitiva y atractiva

### Resultados Técnicos:

- Frontend: Implementación exitosa de Clean Architecture con Jetpack Compose
- Backend: API REST robusta con Spring Boot y JPA
- Base de Datos: Esquema relacional normalizado y eficiente
- Integración: Comunicación fluida entre frontend y backend

## 7.2. Cumplimiento de Objetivos y Requisitos

**Requisitos Funcionales Cumplidos: 8/8 (100%)**

**Requisitos No Funcionales Cumplidos: 6/6 (100%)**

### Métricas de Calidad:

- Código: Seguimiento de patrones arquitectónicos
- Documentación: Cobertura completa de todos los aspectos
- Pruebas: Cobertura adecuada para un proyecto académico
- Usabilidad: Interfaz intuitiva validada con usuarios de prueba

## 7.3. Lecciones Aprendidas y Recomendaciones

### Lecciones Técnicas:

- La separación en módulos facilita el mantenimiento
- Clean Architecture mejora la testabilidad del código
- Jetpack Compose acelera el desarrollo de UI
- Spring Boot simplifica enormemente el desarrollo backend

### Lecciones de Gestión:

- La documentación continua evita acumulación al final
- Las revisiones periódicas mejoran la calidad del código
- La planificación detallada reduce imprevistos

### Recomendaciones para Futuros Proyectos:

1. Implementar CI/CD desde el inicio del proyecto
2. Utilizar GraphQL en lugar de REST para APIs complejas
3. Considerar Jetpack Compose para todos los nuevos proyectos Android
4. Implementar pruebas automatizadas desde las primeras fases
5. Utilizar contenedores Docker para homogenizar entornos

## 8.1. Referencias Técnicas

### Documentación Oficial:

- Android Developer Documentation: <https://developer.android.com>
- Spring Boot Reference Documentation: <https://spring.io/projects/spring-boot>
- Kotlin Language Documentation: <https://kotlinlang.org/docs>
- Jetpack Compose Documentation: <https://developer.android.com/jetpack/compose>

### Tutoriales y Guías:

- Android Developers Codelabs: <https://developer.android.com/codelabs>
- Spring Boot Guides: <https://spring.io/guides>
- Retrofit Documentation: <https://square.github.io/retrofit/>
- Hilt Dependency Injection: <https://developer.android.com/training/dependency-injection/hilt-android>

### Herramientas Utilizadas:

- Android Studio: <https://developer.android.com/studio>
- Postman: <https://www.postman.com>
- MySQL Workbench: <https://www.mysql.com/products/workbench/>
- Git: <https://git-scm.com>

## 8.2. Recursos de Aprendizaje

### Libros de Referencia:

- "Kotlin in Action" - Dmitry Jemerov & Svetlana Isakova
- "Effective Java" - Joshua Bloch
- "Spring in Action" - Craig Walls
- "Clean Architecture" - Robert C. Martin

### Plataformas de Aprendizaje:

- Android Developers Training: <https://developer.android.com/courses>
- Spring Academy: <https://spring.io/academy>
- Kotlin Koans: <https://play.kotlinlang.org/koans>

---

**FECHA DE ENTREGA: 4 de Diciembre de 2025**

**AUTOR: Manuel Damas Moreno**

**CICLO FORMATIVO: Desarrollo de Aplicaciones Multiplataforma**

**CENTRO: I.E.S. Juan Bosco - Alcázar de San Juan**