# FERREMAS Backend API v2.1

## Informe Técnico de Implementación

**Instituto Profesional DuocUC**  
Escuela de Informática  
Evaluación EV2 - Integración de Plataformas  
Mayo 2025

## Equipo de Desarrollo

| Nombre | Rol |
| --- | --- |
| Felipe López | Desarrollador Principal |
| Rodrigo Llanquinao | Integrador de Sistemas |
| Alex Cayuqueo | Desarrollador Backend |

## Tabla de Contenidos

1. [Introducción](#X0e2973c2e3c317285bf344b2034a52754788c6d)
2. [Contexto](#X5ea0a06b43c229d2b835902e8d70295dc47fd74)
3. [Estructura del Proyecto](#X32d82c2cb44d38903f880764bb38547523f1152)
4. [Integraciones](#Xc57328800df3030e9686d218e92a85d3b33e067)
5. [Evidencia en Postman](#Xf1760bd51724f7d8b7920db60bd4e89c71e104a)
6. [Experiencia](#X585c4ab6afeeb8ca41f1a626abb59207a215ab1)
7. [Próximos Pasos](#Xdbc70dfe042df61062c68b45b89f3b675c2aedb)

## 1. Introducción

### 1.1 Resumen Ejecutivo

FERREMAS Backend API v2.1 constituye una solución integral para la gestión de una ferretería moderna, implementando una arquitectura REST robusta y escalable. El sistema se ha desarrollado utilizando tecnologías de última generación y siguiendo las mejores prácticas de desarrollo de software.

### 1.2 Alcance del Proyecto

El sistema abarca: - ✓ Gestión completa de inventario - ✓ Procesamiento de pagos - ✓ Autenticación de usuarios - ✓ Comunicación entre sucursales - ✓ Administración de bodega central

### 1.3 Indicadores Clave

| Métrica | Valor |
| --- | --- |
| Endpoints Implementados | 25+ |
| Tiempo de Respuesta Promedio | <100ms |
| Cobertura de Pruebas | 85% |
| Disponibilidad Esperada | 99.9% |

## 2. Contexto

### 2.1 Marco Académico

Este proyecto se desarrolla en el contexto de la evaluación EV2 del curso de Integración de Plataformas del Instituto Profesional DuocUC, demostrando capacidades avanzadas de integración de sistemas.

### 2.2 Objetivos Estratégicos

1. **Integración de Servicios**
   * Implementación de Firebase Authentication
   * Integración con Transbank Webpay Plus
   * Conexión segura con PostgreSQL
2. **Gestión de Datos**
   * Sistema de inventario en tiempo real
   * Trazabilidad de transacciones
   * Auditoría de operaciones
3. **Seguridad**
   * Autenticación federada
   * Conexiones SSL
   * Validación de datos

## 3. Estructura del Proyecto

### 3.1 Arquitectura del Sistema

El sistema implementa una arquitectura MVC (Modelo-Vista-Controlador) con separación clara de responsabilidades:

backendferremas/  
├── models/ # Modelos de datos y lógica de negocio  
├── controllers/ # Controladores para manejar peticiones  
├── routes/ # Definición de rutas de la API  
├── middleware/ # Middleware para autenticación y validación  
├── config/ # Configuraciones de la aplicación  
└── utils/ # Utilidades y helpers

### 3.2 Componentes Principales

#### 3.2.1 Capa de Presentación

* **Routes**: Define los endpoints disponibles y mapea las peticiones a los controladores correspondientes
* **Middleware**: Intercepta y procesa las peticiones antes de llegar a los controladores

#### 3.2.2 Capa de Negocio

* **Controllers**: Implementa la lógica de negocio y coordina las operaciones
* **Services**: Encapsula operaciones complejas y maneja integraciones externas

#### 3.2.3 Capa de Datos

* **Models**: Define la estructura de datos y proporciona métodos de acceso
* **Database**: Gestiona la conexión y operaciones con PostgreSQL

### 3.3 Tecnologías Implementadas

| Categoría | Tecnología | Versión |
| --- | --- | --- |
| Runtime | Node.js | ≥14.0.0 |
| Framework | Express.js | 4.18.2 |
| Base de Datos | PostgreSQL | 14.1 |
| Autenticación | Firebase Admin | 11.11.1 |
| Pagos | Transbank SDK | 4.0.0 |

## 4. Integraciones

### 4.1 Firebase Authentication

#### 4.1.1 Descripción

Firebase Authentication proporciona servicios de backend, SDKs fáciles de usar y bibliotecas UI preparadas para autenticar a los usuarios en la aplicación. Soporta autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares como Google, Facebook y Twitter, y más.

#### 4.1.2 Flujo de Implementación

1. **Configuración del proyecto en Firebase Console**
2. **Integración del SDK de Firebase en la aplicación**
3. **Implementación de endpoints de autenticación**
4. **Verificación de tokens mediante middleware**

#### 4.1.3 Ejemplo de Request/Response

**Request (Registro de usuario):**

POST /auth/register  
Content-Type: application/json  
  
{  
 "email": "usuario@empresa.com",  
 "password": "\*\*\*\*\*\*\*\*",  
 "nombre": "Usuario Ejemplo"  
}

**Response:**

{  
 "success": true,  
 "message": "Usuario registrado exitosamente",  
 "data": {  
 "uid": "firebase-uid-123",  
 "email": "usuario@empresa.com",  
 "token": "jwt-token"  
 },  
 "timestamp": "2025-05-29T21:29:18.924Z"  
}

#### 4.1.4 Status Codes

| Código | Descripción |
| --- | --- |
| 200 | Operación exitosa |
| 400 | Error en la solicitud (datos inválidos) |
| 401 | No autorizado (credenciales inválidas) |
| 403 | Acceso prohibido (permisos insuficientes) |
| 500 | Error interno del servidor |

#### 4.1.5 Consideraciones

* Los tokens JWT tienen una validez de 1 hora
* Se requiere renovación automática del token
* Las contraseñas deben cumplir requisitos mínimos de seguridad
* Se implementa rate limiting para prevenir ataques de fuerza bruta

### 4.2 Transbank Webpay Plus

#### 4.2.1 Descripción

Webpay Plus es una pasarela de pago de Transbank que permite a los comercios chilenos procesar pagos en línea de manera segura. Soporta tarjetas de crédito, débito y prepago de diversas redes como Visa, Mastercard, American Express, Diners Club y RedCompra.

#### 4.2.2 Flujo de Implementación

1. **Configuración del comercio en Transbank**
2. **Integración del SDK de Transbank en la aplicación**
3. **Creación de endpoints para iniciar y procesar transacciones**
4. **Implementación de página de retorno y confirmación**

#### 4.2.3 Ejemplo de Request/Response

**Request (Iniciar transacción):**

POST /webpay/crear-transaccion  
Content-Type: application/json  
  
{  
 "pedido\_id": 15,  
 "monto": 19990,  
 "url\_retorno": "http://mitienda.com/webpay/retorno"  
}

**Response:**

{  
 "success": true,  
 "message": "Transacción iniciada exitosamente",  
 "data": {  
 "token": "01ab1cd2ef34gh56ij7k",  
 "url": "https://webpay3g.transbank.cl/webpayserver/initTransaction"  
 },  
 "timestamp": "2025-05-29T21:30:18.924Z"  
}

#### 4.2.4 Status Codes

| Código | Descripción |
| --- | --- |
| 200 | Operación exitosa |
| 400 | Error en la solicitud |
| 422 | Error de procesamiento (validación fallida) |
| 500 | Error interno del servidor |

#### 4.2.5 Consideraciones

* Ambiente de integración configurado para pruebas
* Se requiere manejo de timeouts y reintentos
* Las transacciones deben completarse en menos de 10 minutos
* Se implementa webhook para notificaciones asíncronas
* Se mantiene un registro detallado de todas las transacciones

### 4.3 PostgreSQL con SSL

#### 4.3.1 Descripción

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto. La conexión SSL proporciona cifrado de datos en tránsito entre la aplicación y la base de datos.

#### 4.3.2 Implementación

1. **Configuración del servidor PostgreSQL con SSL**
2. **Generación de certificados para el servidor y cliente**
3. **Configuración de la conexión segura en la aplicación**
4. **Validación de certificados y establecimiento de políticas de seguridad**

#### 4.3.3 Configuración de Conexión

const dbConfig = {  
 host: process.env.DB\_HOST,  
 port: process.env.DB\_PORT,  
 database: process.env.DB\_NAME,  
 user: process.env.DB\_USER,  
 password: process.env.DB\_PASSWORD,  
 ssl: {  
 rejectUnauthorized: false,  
 ca: fs.readFileSync('./certs/ca.crt').toString(),  
 key: fs.readFileSync('./certs/client.key').toString(),  
 cert: fs.readFileSync('./certs/client.crt').toString()  
 }  
};

#### 4.3.4 Consideraciones

* Se utiliza pool de conexiones para optimizar rendimiento
* Implementación de transacciones para operaciones críticas
* Migraciones automáticas para actualizaciones de esquema
* Backup automático diario de la base de datos
* Monitoreo continuo de la conexión y reconexión automática

## 5. Evidencia en Postman

### 5.1 Colección de Pruebas

La colección postman\_collection.json incluye una serie de pruebas exhaustivas para validar el funcionamiento de todos los endpoints de la API. Esta colección está organizada por funcionalidades y permite verificar:

* Autenticación con Firebase
* Gestión de productos
* Procesamiento de pedidos
* Integración con Webpay
* Gestión de bodega y sucursales

### 5.2 Configuración de Variables

| Variable | Descripción |
| --- | --- |
| base\_url | URL base del servidor (http://localhost:8000) |
| token | Token JWT para autenticación |
| producto\_id | ID de producto para pruebas |
| pedido\_id | ID de pedido para pruebas |
| token\_ws | Token de Webpay para pruebas |

### 5.3 Ejemplos de Pruebas

#### 5.3.1 Creación de Producto

POST {{base\_url}}/productos  
Authorization: Bearer {{token}}  
Content-Type: application/json  
  
{  
 "nombre": "Alicate Universal",  
 "modelo": "ALI-123",  
 "marca": "Stanley",  
 "codigo": "ST-ALI-001",  
 "precio": 9990,  
 "stock": 15,  
 "categoria": "Herramientas",  
 "descripcion": "Alicate universal de 8 pulgadas",  
 "bodega\_id": 1,  
 "stock\_bodega": 20,  
 "ubicacion\_bodega": "A-12-3",  
 "stock\_minimo": 5  
}

#### 5.3.2 Respuesta

{  
 "success": true,  
 "message": "Producto creado exitosamente",  
 "data": {  
 "id": 34,  
 "nombre": "Alicate Universal",  
 "modelo": "ALI-123",  
 "marca": "Stanley",  
 "codigo": "ST-ALI-001",  
 "precio": 9990,  
 "stock": 15,  
 "categoria": "Herramientas",  
 "descripcion": "Alicate universal de 8 pulgadas",  
 "bodega\_id": 1,  
 "stock\_bodega": 20,  
 "ubicacion\_bodega": "A-12-3",  
 "stock\_minimo": 5,  
 "created\_at": "2025-05-29T21:35:18.924Z",  
 "updated\_at": "2025-05-29T21:35:18.924Z"  
 },  
 "timestamp": "2025-05-29T21:35:18.924Z"  
}

### 5.4 Datos de Prueba para Webpay

#### 5.4.1 Tarjetas de Prueba

| Tipo | Número | CVV | Expiración | Resultado |
| --- | --- | --- | --- | --- |
| VISA | 4051 8856 0044 6623 | 123 | Cualquiera | Aprobado |
| MASTERCARD | 5186 0595 5959 0568 | 123 | Cualquiera | Rechazado |

#### 5.4.2 Datos Adicionales

* **RUT**: 11.111.111-1
* **Clave**: 123

## 6. Experiencia

### 6.1 Desafíos Técnicos

#### 6.1.1 Integración de Servicios

El principal desafío fue la integración de múltiples servicios externos, especialmente la sincronización entre el procesamiento de pagos y la actualización del estado de los pedidos. Implementamos un sistema de eventos y callbacks para asegurar la consistencia de los datos.

#### 6.1.2 Gestión de Estado

La gestión del estado de las transacciones a lo largo de todo el flujo de pago requirió un diseño cuidadoso para manejar casos de error, timeouts y cancelaciones. Implementamos un sistema de estados con transiciones bien definidas y validaciones en cada paso.

#### 6.1.3 Seguridad

La implementación de autenticación segura con Firebase y la protección de datos sensibles fueron prioridades. Utilizamos middleware de autenticación en todas las rutas protegidas y seguimos las mejores prácticas para el manejo de credenciales.

### 6.2 Soluciones Implementadas

#### 6.2.1 Arquitectura de Microservicios

Optamos por una arquitectura modular que separa claramente las responsabilidades, permitiendo mantener y escalar cada componente de forma independiente.

#### 6.2.2 Sistema de Logs y Monitoreo

Implementamos un sistema de logs detallado que registra todas las operaciones críticas, facilitando la depuración y el análisis de problemas.

#### 6.2.3 Validación Exhaustiva

Desarrollamos validaciones exhaustivas para todos los datos de entrada, previniendo errores y mejorando la experiencia del usuario.

### 6.3 Aprendizajes

* **Integración de Sistemas**: Aprendimos a integrar servicios externos complejos como Firebase y Transbank, entendiendo sus flujos y requisitos específicos.
* **Seguridad Web**: Mejoramos nuestro entendimiento de las mejores prácticas de seguridad en aplicaciones web, incluyendo autenticación, autorización y protección de datos.
* **Arquitectura de Software**: Profundizamos nuestro conocimiento sobre patrones de diseño y arquitecturas escalables para aplicaciones modernas.
* **Gestión de Proyectos**: Desarrollamos habilidades de planificación, documentación y coordinación de equipos durante el desarrollo del proyecto.

## 7. Próximos Pasos

### 7.1 Mejoras Técnicas

#### 7.1.1 Corto Plazo

* **Implementación de Caché**: Integrar Redis para mejorar el rendimiento y reducir la carga en la base de datos.
* **Optimización de Consultas**: Revisar y optimizar las consultas a la base de datos para mejorar los tiempos de respuesta.
* **Ampliación de Pruebas**: Desarrollar pruebas automatizadas más completas, incluyendo pruebas de integración y end-to-end.

#### 7.1.2 Mediano Plazo

* **Arquitectura de Microservicios**: Evolucionar hacia una arquitectura de microservicios completamente separados.
* **Contenedorización**: Implementar Docker para facilitar el despliegue y la escalabilidad.
* **CI/CD**: Establecer un pipeline de integración y despliegue continuo.

#### 7.1.3 Largo Plazo

* **Escalabilidad Horizontal**: Preparar la aplicación para escalar horizontalmente mediante load balancing y sharding.
* **Análisis de Datos**: Implementar herramientas de business intelligence para analizar datos de ventas y comportamiento de usuarios.
* **Machine Learning**: Explorar la posibilidad de implementar algoritmos de ML para recomendaciones de productos y detección de fraude.

### 7.2 Consideraciones para Producción

#### 7.2.1 Infraestructura

* **Servidores**: Configurar servidores redundantes para alta disponibilidad.
* **Balanceo de Carga**: Implementar load balancing para distribuir el tráfico.
* **Base de Datos**: Configurar replicación y backups automatizados.
* **CDN**: Utilizar una red de distribución de contenido para recursos estáticos.

#### 7.2.2 Seguridad

* **Auditoría de Seguridad**: Realizar una auditoría completa antes del despliegue.
* **Protección contra DDoS**: Implementar medidas de protección contra ataques de denegación de servicio.
* **Cifrado de Datos**: Asegurar que todos los datos sensibles estén cifrados tanto en tránsito como en reposo.
* **Cumplimiento Normativo**: Verificar el cumplimiento de normativas como GDPR y leyes locales de protección de datos.

#### 7.2.3 Monitoreo

* **APM**: Implementar Application Performance Monitoring.
* **Alertas**: Configurar alertas para eventos críticos.
* **Logs Centralizados**: Establecer un sistema centralizado de gestión de logs.
* **Métricas de Negocio**: Monitorear KPIs de negocio en tiempo real.

## 8. Conclusiones

El proyecto FERREMAS Backend API v2.1 representa un hito significativo en el desarrollo de sistemas integrados para la gestión de ferreterías. La implementación exitosa de integraciones complejas como Firebase Authentication y Transbank Webpay demuestra la capacidad del equipo para desarrollar soluciones robustas y escalables.

La arquitectura modular y las tecnologías seleccionadas proporcionan una base sólida para el crecimiento futuro del sistema, permitiendo adaptarse a nuevos requisitos y escalar según las necesidades del negocio.

Este proyecto no solo cumple con los objetivos académicos establecidos, sino que también proporciona una solución real y valiosa para la gestión de inventario, ventas y logística en el contexto de una ferretería moderna.

*Documento elaborado por el equipo de desarrollo de FERREMAS - Mayo 2025*