Documentación Técnica - Survey Platform Backend

Versión: 1.0.0
Fecha: Noviembre 2025
Autor: Oscar Ortiz
Tecnología: Spring Boot 3.4.0 + Firebase Realtime Database

---

1. Resumen Ejecutivo

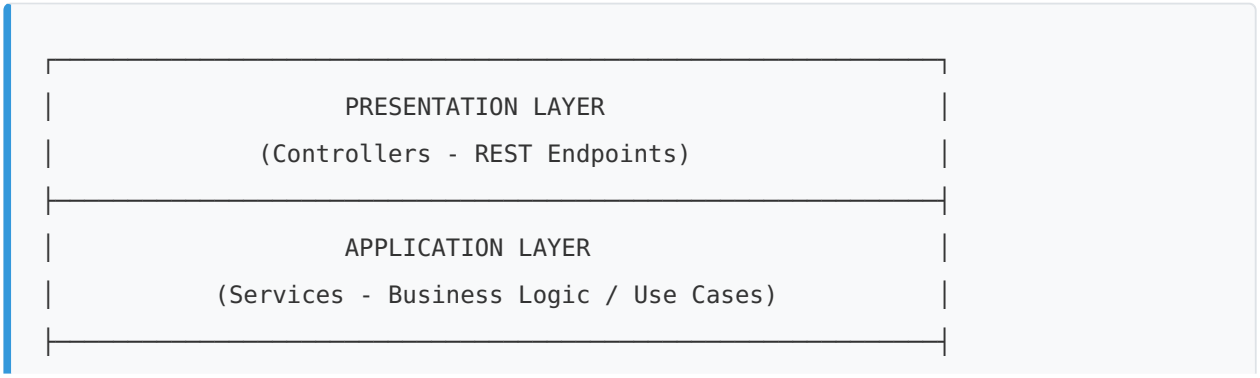
API REST para plataforma de encuestas empresariales desarrollada con arquitectura hexagonal, implementando autenticación JWT y almacenamiento en tiempo real con Firebase.

Características principales:
- Autenticación y autorización segura con JWT
- CRUD completo de encuestas con preguntas dinámicas
- Almacenamiento en tiempo real con Firebase Realtime Database
- Documentación interactiva con Swagger/OpenAPI
- 17 tests unitarios (100% passing)

---

2. Arquitectura del Sistema

2.1 Arquitectura Hexagonal (Clean Architecture)



```
│                      DOMAIN LAYER                                 │
│            (Entities, Ports/Interfaces)                           │
├──────────────────────────────────────────────────────────────────┤
│                   INFRASTRUCTURE LAYER                            │
│    (Adapters, Firebase, Security, Configuration)                  │
└──────────────────────────────────────────────────────────────────┘
```

2.2 Estructura de Carpetas

```
src/main/java/com/davivienda/survey/
├── domain/
│   ├── model/
│   │   ├── User.java              # Entidad Usuario
│   │   ├── Survey.java            # Entidad Encuesta
│   │   ├── Question.java          # Entidad Pregunta
│   │   ├── Answer.java            # Entidad Respuesta
│   │   └── SurveyResponse.java    # Respuesta completa
│   └── port/
│       ├── UserRepository.java         # Puerto Usuario
│       ├── SurveyRepository.java       # Puerto Encuesta
│       └── ResponseRepository.java     # Puerto Respuestas
│
├── application/
│   ├── service/
│   │   ├── AuthService.java       # Lógica autenticación
│   │   ├── SurveyService.java     # Lógica encuestas
│   │   └── ResponseService.java   # Lógica respuestas
│   └── dto/
│       ├── RegisterRequest.java   # DTO registro
│       ├── LoginRequest.java      # DTO login
│       └── AuthResponse.java      # DTO respuesta auth
│
├── infrastructure/
│   ├── adapter/
│   │   ├── FirebaseUserAdapter.java      # Implementación Firebase User
│   │   ├── FirebaseSurveyAdapter.java    # Implementación Firebase Survey
│   │   └── FirebaseResponseAdapter.java  # Implementación Firebase Response
│   ├── config/
│   │   ├── FirebaseConfig.java           # Configuración Firebase
│   │   ├── SecurityConfig.java           # Configuración Security
```

```
│   │   ├── CorsConfig.java              # Configuración CORS
│   │   └── OpenApiConfig.java           # Configuración Swagger
│   ├── security/
│   │   ├── JwtService.java              # Servicio JWT
│   │   ├── JwtAuthenticationFilter.java  # Filtro JWT
│   │   └── CustomUserDetailsService.java # Servicio UserDetails
│   └── exception/
│       └── GlobalExceptionHandler.java   # Manejo global errores
│
└── presentation/
    └── controller/
        ├── AuthController.java        # Endpoints autenticación
        ├── SurveyController.java      # Endpoints encuestas
        ├── QuestionController.java    # Endpoints preguntas
        └── ResponseController.java    # Endpoints respuestas
```

---

3. Stack Tecnológico

Tecnología	Versión	Propósito
Java	21	Lenguaje base
Spring Boot	3.4.0	Framework principal
Spring Security	6.x	Autenticación/Autorización
Firebase Admin SDK	9.2.0	Base de datos en tiempo real
JJWT	0.12.3	Generación/validación JWT
Lombok	Latest	Reducción boilerplate
JUnit 5	5.x	Testing framework
Mockito	5.x	Mocking para tests
Springdoc OpenAPI	2.7.0	Documentación Swagger
Maven	3.9.11	Gestión dependencias

4. Modelo de Datos

4.1 Entidades Principales

User

```
public class User {
    private String id;              // UUID único
    private String name;            // Nombre completo
    private String email;           // Email (único)
    private String password;        // Contraseña (BCrypt)
    private Date createdAt;         // Fecha creación
    private Date updatedAt;         // Fecha actualización
}
```

Survey

```
public class Survey {
    private String id;                      // UUID único
    private String title;                   // Título encuesta
    private String description;             // Descripción
    private String createdBy;               // ID usuario creador
    private boolean isPublished;            // Estado publicación
    private Date createdAt;                 // Fecha creación
    private Date updatedAt;                 // Fecha actualización
    private List<Question> questions;       // Lista preguntas
}
```

Question

```
public class Question {
    private String id;              // UUID único
    private String surveyId;        // ID encuesta padre
    private String title;           // Texto pregunta
    private QuestionType type;      // Tipo pregunta
    private List<String> options;   // Opciones (si aplica)
    private boolean required;       // Obligatoria
```

```
    private int order;              // Orden presentación
    private String imageUrl;        // URL imagen (opcional)
}

enum QuestionType {
    TEXT,              // Texto libre
    MULTIPLE_CHOICE,   // Opción múltiple
    CHECKBOX,          // Múltiples respuestas
    DROPDOWN,          // Lista desplegable
    SCALE              // Escala numérica
}
```

4.2 Estructura Firebase Realtime Database

```
{
  "users": {
    "user-uuid-123": {
      "id": "user-uuid-123",
      "name": "Juan Pérez",
      "email": "juan@example.com",
      "password": "$2a$10$...",
      "createdAt": "2025-11-08T10:30:00.000Z",
      "updatedAt": "2025-11-08T10:30:00.000Z"
    }
  },
  "surveys": {
    "survey-uuid-456": {
      "id": "survey-uuid-456",
      "title": "Encuesta Satisfacción",
      "description": "Evaluación servicio",
      "createdBy": "user-uuid-123",
      "isPublished": true,
      "createdAt": "2025-11-08T11:00:00.000Z",
      "updatedAt": "2025-11-08T11:00:00.000Z",
      "questions": [...]
    }
  },
  "responses": {
    "response-uuid-789": {
      "id": "response-uuid-789",
```

```
      "surveyId": "survey-uuid-456",
      "respondentId": "user-uuid-999",
      "completedAt": "2025-11-08T12:00:00.000Z",
      "answers": [...]
    }
  }
}
```

---

5. API REST Endpoints

5.1 Autenticación

POST /api/auth/register

Descripción: Registrar nuevo usuario
Autenticación: No requerida
Request Body:

```
{
  "name": "Juan Pérez",
  "email": "juan@example.com",
  "password": "password123"
}
```

Response: 200 OK

```
{
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "user": {
    "id": "user-uuid",
    "name": "Juan Pérez",
    "email": "juan@example.com"
  }
}
```

POST /api/auth/login

Descripción: Iniciar sesión
Autenticación: No requerida
Request Body:

```
{
  "email": "juan@example.com",
  "password": "password123"
}
```

Response: 200 OK (mismo formato que register)

GET /api/auth/me

Descripción: Obtener usuario actual
Autenticación: JWT requerido
Response: 200 OK

```
{
  "id": "user-uuid",
  "name": "Juan Pérez",
  "email": "juan@example.com"
}
```

5.2 Encuestas

GET /api/surveys/my-surveys

Descripción: Listar encuestas del usuario autenticado
Autenticación: JWT requerido
Response: 200 OK - Array de Survey

POST /api/surveys

Descripción: Crear nueva encuesta
Autenticación: JWT requerido
Request Body:

```
{
  "title": "Encuesta Satisfacción",
  "description": "Evaluación servicio"
}
```

PUT /api/surveys/{id}

Descripción: Actualizar encuesta existente
Autenticación: JWT requerido (solo creador)

DELETE /api/surveys/{id}

Descripción: Eliminar encuesta
Autenticación: JWT requerido (solo creador)

PUT /api/surveys/{id}/publish

Descripción: Publicar encuesta
Autenticación: JWT requerido

GET /api/surveys/published

Descripción: Listar encuestas publicadas
Autenticación: JWT requerido

5.3 Preguntas

POST /api/surveys/{surveyId}/questions

Descripción: Agregar pregunta a encuesta
Autenticación: JWT requerido
Request Body:

```
{
  "title": "¿Cómo califica el servicio?",
  "type": "SCALE",
  "options": ["1", "2", "3", "4", "5"],
  "required": true,
  "order": 1
}
```

PUT /api/surveys/{surveyId}/questions/{questionId}

Descripción: Actualizar pregunta
Autenticación: JWT requerido

DELETE /api/surveys/{surveyId}/questions/{questionId}

Descripción: Eliminar pregunta
Autenticación: JWT requerido

---

6. Seguridad

6.1 Autenticación JWT

Flujo:
1. Usuario se registra/inicia sesión
2. Backend valida credenciales
3. Genera token JWT con claims: email, rol, expiración
4. Cliente almacena token (localStorage/cookies)
5. Cliente incluye token en header: `Authorization: Bearer {token}`
6. `JwtAuthenticationFilter` intercepta requests
7. Valida token y extrae usuario
8. Spring Security autoriza acceso

Configuración JWT:

```
jwt.secret=clave_secreta_minimo_256_bits
jwt.expiration=86400000  # 24 horas
```

6.2 Encriptación de Contraseñas

- **Algoritmo:** BCrypt
- **Rounds:** 10 (default)
- **Salt:** Generado automáticamente

6.3 CORS Configuration

```
@Configuration
public class CorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration();
        config.setAllowedOrigins(List.of("http://localhost:5173"));
        config.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE"));
        config.setAllowedHeaders(List.of("*"));
        config.setAllowCredentials(true);
        source.registerCorsConfiguration("/**", config);
        return new CorsFilter(source);
    }
}
```

---

7. Testing

7.1 Cobertura de Tests

Total: 17 tests unitarios - 100% passing

Clase	Tests	Estado
AuthServiceTest	4	✅ Passing
SurveyServiceTest	8	✅ Passing
JwtServiceTest	5	✅ Passing

7.2 Estrategia de Testing

- **Unit Tests:** Mockito para aislar dependencias
- **Integration Tests:** Spring Boot Test (planificado)
- **Coverage Target:** >80%

Ejemplo de test:

```
@Test
void register_ShouldCreateUserSuccessfully() {
    // Given
    RegisterRequest request = new RegisterRequest("Juan", "juan@test.com", "pass123");
    when(userRepository.existsByEmail(anyString())).thenReturn(false);
    when(passwordEncoder.encode(anyString())).thenReturn("encodedPass");

    // When
    AuthResponse response = authService.register(request);

    // Then
    assertNotNull(response.getToken());
    assertEquals("Juan", response.getUser().getName());
    verify(userRepository).save(any(User.class));
}
```

---

8. Configuración y Despliegue

8.1 Variables de Entorno

```
# Firebase
FIREBASE_DATABASE_URL=https://proyecto.firebaseio.com
FIREBASE_CREDENTIALS_PATH=/path/to/firebase-config.json

# JWT
JWT_SECRET=tu_secreto_256_bits
JWT_EXPIRATION=86400000

# Server
SERVER_PORT=8080
SPRING_PROFILES_ACTIVE=prod
```

8.2 Despliegue con Docker

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app
COPY target/survey-platform-1.0.0.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Construcción y ejecución:

```
mvn clean package
docker build -t survey-platform-backend .
docker run -p 8080:8080 -e JWT_SECRET=secret survey-platform-backend
```

8.3 Profiles de Spring

- **dev:** Desarrollo local, logs verbose
- **test:** Para tests automatizados
- **prod:** Producción, optimizado

---

9. Documentación API (Swagger)

URL: http://localhost:8080/swagger-ui.html

Características:
- Documentación interactiva de todos los endpoints
- Pruebas en vivo desde el navegador
- Autenticación JWT integrada
- Exportación OpenAPI (JSON/YAML)

OpenAPI Spec:
- JSON: http://localhost:8080/v3/api-docs
- YAML: http://localhost:8080/v3/api-docs.yaml

---

10. Mantenimiento y Monitoreo

10.1 Logs

```
# application.properties
logging.level.root=INFO
logging.level.com.davivienda.survey=DEBUG
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n
```

10.2 Health Check

Endpoint: GET /actuator/health
Response:

```
{
  "status": "UP"
}
```

10.3 Métricas (Planificado)

- Spring Boot Actuator
- Prometheus metrics
- Grafana dashboards

---

11. Troubleshooting

11.1 Errores Comunes

Error: Firebase credentials not found

```
# Solución: Verificar archivo firebase-config.json
ls src/main/resources/firebase-config.json
```

Error: JWT Secret too short

```
# Solución: Generar secret de 256 bits
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"
```

Error: Port 8080 already in use

```
# Solución Windows:
netstat -ano | findstr :8080
taskkill /PID <PID> /F
```

---

12. Roadmap

Fase 1 (Completado)

- ✅ Arquitectura hexagonal
- ✅ Autenticación JWT
- ✅ CRUD encuestas
- ✅ Firebase integration
- ✅ Tests unitarios
- ✅ Swagger documentation

Fase 2 (Planificado)

- ⏳ Analytics y reportes
- ⏳ Exportación a PDF/CSV
- ⏳ Notificaciones email
- ⏳ Rate limiting
- ⏳ Caché con Redis

Fase 3 (Futuro)

- 📋 Encuestas programadas
- 📋 Templates de encuestas
- 📋 Integración con BI tools

- 📋 API versioning

---

13. Referencias

- **Spring Boot:** https://spring.io/projects/spring-boot
- **Firebase Admin SDK:** https://firebase.google.com/docs/admin/setup
- **JWT:** https://jwt.io/
- **OpenAPI:** https://swagger.io/specification/
- **Repository:** https://github.com/oscarortiz1/prueba_davivienda

---

Documento generado: Noviembre 2025
Versión: 1.0.0
Estado: Producción