📚 VIDAJOVE - Documentación del Proyecto

Proyecto Intermodular 1º DAM - 2º Trimestre

📋 Índice

🎯 Descripción del Proyecto

VidaJove es una plataforma web diseñada para jóvenes que centraliza información sobre:

- Noticias y eventos juveniles
- Convocatorias y becas
- Recursos formativos
- Actividades culturales y deportivas

Funcionalidades Principales

✅ **Gestión de Noticias**: CRUD completo con categorías, búsqueda y destacados ✅ **Gestión de Eventos**: Inscripciones, calendario y control de plazas ✅ **Sistema de Usuarios**: Roles (admin, editor, usuario) ✅ **Recursos**: Becas, convocatorias y documentación ✅ **Comentarios**: Interacción en noticias

---

💻 Requisitos del Sistema

Software Necesario

- **Java**: JDK 11 o superior
- **MySQL**: 8.0 o superior

- **IDE**: IntelliJ IDEA, Eclipse o NetBeans
- **Git**: Para control de versiones
- **Maven** (opcional): Para gestión de dependencias

Dependencias Java

```xml
<!-- MySQL Connector -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
</dependency>

<!-- JUnit para testing -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

🚀 Instalación y Configuración

Paso 1: Clonar el Repositorio

```bash
git clone https://github.com/tu-usuario/vidajove.git
cd vidajove
```

Paso 2: Configurar la Base de Datos

1. Abrir MySQL Workbench o acceder por terminal:

```bash
mysql -u root -p
```

2. Ejecutar el script SQL:

```bash
```

```
mysql -u root -p < database/vidajove_db.sql
```

3. Verificar la creación:

```sql
SHOW DATABASES;
USE vidajove_db;
SHOW TABLES;
```

Paso 3: Configurar Variables de Entorno

1. Copiar el archivo de ejemplo:

```bash
cp .env.example .env
```

2. Editar `.env` con tus credenciales:

```properties
DB_URL=jdbc:mysql://localhost:3306/vidajove_db
DB_USER=vidajove_app
DB_PASSWORD=TU_CONTRASEÑA
```

3. **IMPORTANTE**: Añadir `.env` al `.gitignore`:

```bash
echo ".env" >> .gitignore
```

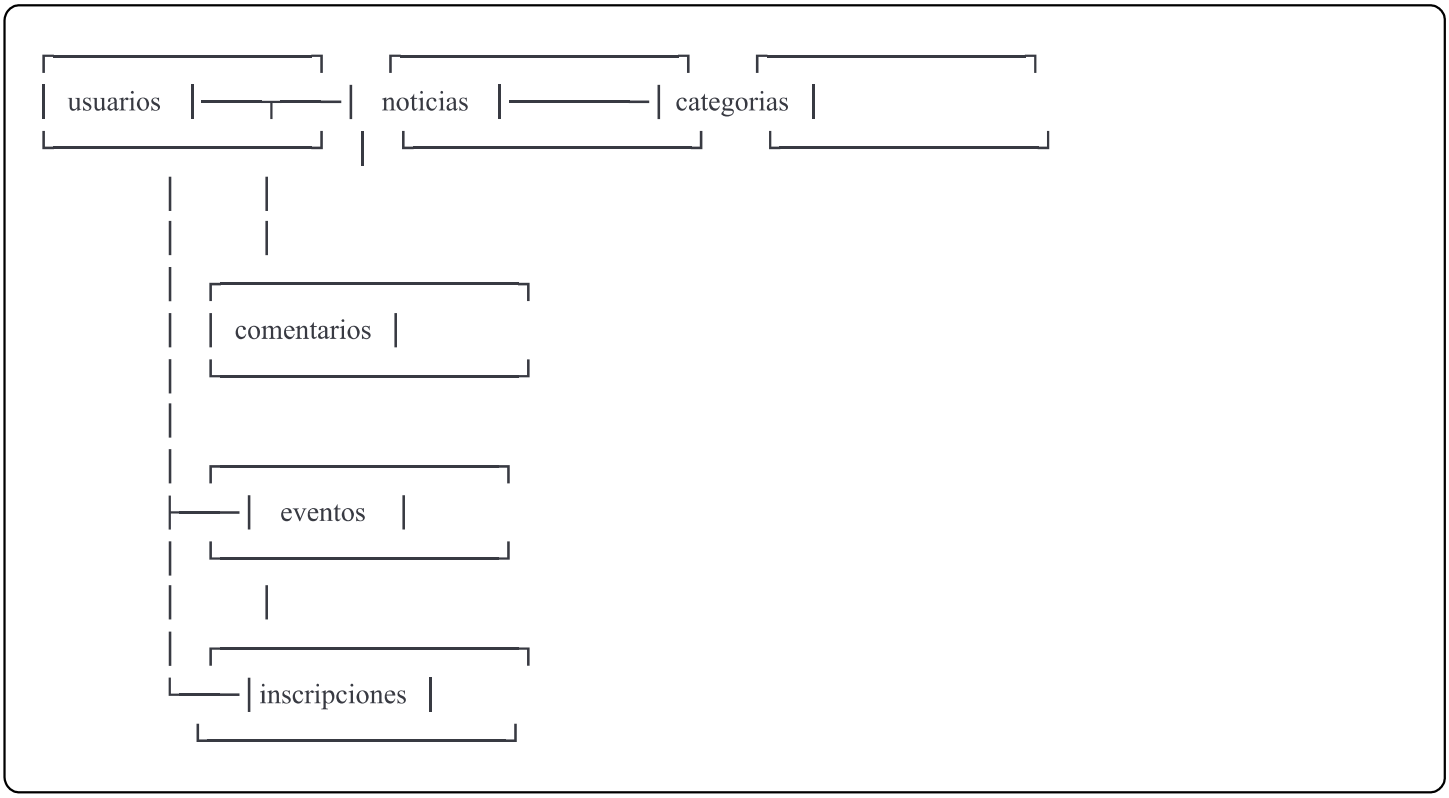
Paso 4: Compilar y Ejecutar

```bash
# Compilar
javac -d bin src/com/vidajove/**/*.java

# Ejecutar
java -cp bin com.vidajove.MainApp
```

🗄️ Estructura de la Base de Datos

Diagrama Entidad-Relación



Tablas Principales

1. usuarios

- Almacena información de todos los usuarios del sistema
- Roles: admin, editor, usuario
- Incluye autenticación y permisos

2. noticias

- Contenido principal de la plataforma
- Estados: borrador, publicada, archivada
- Relación con categorías y autores

3. eventos

- Gestión de actividades y eventos
- Control de inscripciones y plazas
- Información de fecha, lugar y organizador

4. categorias

- Clasificación de noticias
- Configuración visual (color, icono)

5. recursos

- Becas, convocatorias y documentación
- Enlaces externos y archivos adjuntos

Normalización

✅ **1FN**: No hay grupos repetitivos, cada campo contiene valores atómicos ✅ **2FN**: Todas las columnas dependen de la clave primaria completa ✅ **3FN**: No hay dependencias transitivas

---

📁 Estructura del Proyecto

```
vidajove/
│
├── src/
│   └── com/
│       └── vidajove/
│           ├── config/
│           │   └── DatabaseConfig.java
│           ├── model/
│           │   ├── Noticia.java
│           │   ├── Usuario.java
│           │   ├── Evento.java
│           │   └── Categoria.java
│           ├── dao/
│           │   ├── CrudDAO.java (interfaz)
│           │   ├── NoticiaDAO.java
│           │   ├── UsuarioDAO.java
│           │   └── EventoDAO.java
│           ├── service/
│           │   └── NoticiaService.java
│           └── MainApp.java
│
├── database/
│   └── vidajove_db.sql
│
├── docs/
│   ├── manual_usuario.pdf
│   ├── diagrama_clases.png
│   └── casos_prueba.xlsx
```

```
│
├── test/
│   └── com/
│       └── vidajove/
│           └── dao/
│               └── NoticiaDAOTest.java
│
├── .env
├── .gitignore
├── README.md
└── pom.xml (si usas Maven)
```

🔧 Uso del CRUD

Operaciones Básicas

CREATE - Crear Noticia

java

```java
NoticiaDAO dao = new NoticiaDAO();
Noticia noticia = new Noticia(
    "Título de la noticia",
    "Subtítulo descriptivo",
    "Contenido completo de la noticia...",
    1, // id_categoria
    2  // id_autor
);

if (dao.crear(noticia)) {
    System.out.println("Noticia creada con ID: " + noticia.getIdNoticia());
}
```

READ - Leer Noticia

java

```java
// Obtener por ID
Noticia noticia = dao.obtenerPorId(1);

// Obtener todas
List<Noticia> todas = dao.obtenerTodos();

// Obtener por categoría
List<Noticia> cultura = dao.obtenerPorCategoria(1);

// Obtener destacadas
List<Noticia> destacadas = dao.obtenerDestacadas();
```

UPDATE - Actualizar Noticia

```java
Noticia noticia = dao.obtenerPorId(1);
noticia.setTitulo("Nuevo título");
noticia.setContenido("Contenido actualizado");

if (dao.actualizar(noticia)) {
    System.out.println("Noticia actualizada");
}
```

DELETE - Eliminar Noticia

```java
if (dao.eliminar(1)) {
    System.out.println("Noticia eliminada");
}
```

Operaciones Adicionales

Publicar Noticia

```java
dao.publicarNoticia(1);
```

Buscar por Texto

```java
List<Noticia> resultados = dao.buscarPorTexto("becas");
```

Incrementar Visitas

```java
dao.incrementarVisitas(1);
```

🧪 Testing

Casos de Prueba Principales

CP001: Crear Noticia Válida

- **Entrada**: Datos completos y válidos
- **Resultado esperado**: Noticia creada con ID asignado
- **Estado**: ✅ Pasado

CP002: Crear Noticia con Datos Incompletos

- **Entrada**: Título vacío
- **Resultado esperado**: Error de validación
- **Estado**: ✅ Pasado

CP003: Actualizar Noticia Existente

- **Entrada**: ID válido y nuevos datos
- **Resultado esperado**: Noticia actualizada
- **Estado**: ✅ Pasado

CP004: Eliminar Noticia Inexistente

- **Entrada**: ID no existente
- **Resultado esperado**: false, sin errores
- **Estado**: ✅ Pasado

CP005: Buscar por Texto

- **Entrada**: Palabra clave "festival"
- **Resultado esperado**: Lista de noticias relacionadas
- **Estado**: ✅ Pasado

Ejemplo de Test con JUnit

```java
import org.junit.Test;
import static org.junit.Assert.*;

public class NoticiaDAOTest {

    @Test
    public void testCrearNoticia() {
        NoticiaDAO dao = new NoticiaDAO();
        Noticia noticia = new Noticia(
            "Test Noticia",
            "Subtitulo test",
            "Contenido de prueba",
            1, 1
        );

        assertTrue(dao.crear(noticia));
        assertTrue(noticia.getIdNoticia() > 0);
    }

    @Test
    public void testObtenerPorId() {
        NoticiaDAO dao = new NoticiaDAO();
        Noticia noticia = dao.obtenerPorId(1);

        assertNotNull(noticia);
        assertEquals(1, noticia.getIdNoticia());
    }
}
```

🗓️ Planificación del Desarrollo

Semana 1: Base de Datos ✅

- Creación de tablas
- Definición de relaciones
- Creación de usuarios y permisos
- **Evidencia**: Script SQL en GitHub

Semana 2: Normalización ✅

- Aplicación de 1FN-3FN
- Datos de prueba
- **Evidencia**: Documento de normalización

Semana 3: CRUD ✅

- Implementación completa de operaciones
- Clase Noticia y NoticiaDAO
- **Evidencia**: Código en GitHub

Semana 4: Conexión Segura (En curso)

- Variables de entorno
- Pool de conexiones
- **Evidencia**: Archivo .env.example

Semanas 5-6: Interfaz y Testing

- Pantallas principales
- Casos de prueba
- **Entregable**: Demo funcional

Semanas 7-8: Funcionalidades Adicionales

- Búsquedas avanzadas
- Exportación de datos
- **Entregable**: Manual técnico

Semanas 9-10: Revisión y Demo Final

- Documentación completa
- Presentación del proyecto
- **Entregable**: MVP funcional

👥 Roles del Equipo

Coordinador

- Gestión de Trello
- Reuniones de seguimiento
- Control de plazos

Programador

- Desarrollo del CRUD
- Implementación de funcionalidades
- Commits en GitHub

Documentación

- Manuales técnicos
- Diagramas
- Casos de uso

Testing

- Casos de prueba
- Detección de bugs
- Validación de funcionalidades

📊 Criterios de Evaluación

Criterio	Peso	Estado
CRUD funcional	25%	✅ Completado
Conexión BBDD	15%	🟡 En proceso
Interfaz y UX	10%	⏳ Pendiente
Pruebas	15%	⏳ Pendiente
Documentación	10%	🟡 En proceso
Planificación	10%	✅ Completado

Criterio	Peso	Estado
Demo final	15%	⏳ Pendiente

🔐 Seguridad

Buenas Prácticas Implementadas

- ✅ Variables de entorno para credenciales
- ✅ Usuario de BD con permisos limitados
- ✅ PreparedStatement para prevenir SQL Injection
- ✅ Validación de datos de entrada
- ✅ .env en .gitignore

Mejoras Pendientes

- ⏳ Hash de contraseñas (BCrypt)
- ⏳ Sanitización de inputs
- ⏳ Logs de auditoría

📞 Contacto y Soporte

- **Repositorio**: https://github.com/tu-usuario/vidajove
- **Trello**: [Enlace al tablero]
- **Google Drive**: [Enlace a la carpeta]

📝 Changelog

Versión 1.0.0 (Semana 3)

- ✅ Base de datos completa
- ✅ CRUD de noticias implementado
- ✅ Conexión con BD configurada
- ✅ Datos de prueba insertados

Próxima versión (1.1.0)

- ⏳ Interfaz gráfica
- ⏳ Sistema de login

- ⏳ Gestión de eventos

🎓 Recursos de Aprendizaje

- Documentación MySQL
- Java JDBC Tutorial
- Git Cheat Sheet
- Markdown Guide

Última actualización: Enero 2025 **Autor**: Equipo VidaJove - 1º DAM