📘 MANUAL DE INGENIERÍA Y DESARROLLO INTEGRAL: SPOT-LIGHT

Versión: 3.0 (Definitiva) **Alcance:** Base de Datos + Backend API + App Móvil + Web Admin **Fecha:** 21 Enero 2026

🛑 FASE 0: PRE-REQUISITOS OBLIGATORIOS

Antes de escribir código, cada integrante debe tener esto instalado:

1. **Alexander (Backend):**
   - Visual Studio 2022 (Community). Cargas de trabajo: "ASP.NET y desarrollo web".
   - .NET 8.0 SDK.
   - Postman (Para probar endpoints).
2. **Javier y Emily (Móvil/Web):**
   - VS Code.
   - Flutter SDK (Instalado y agregado al PATH).
   - Emulador de Android (Android Studio) o un celular físico con "Depuración USB" activa.
   - Extensión "Live Server" en VS Code (Para la Web).
3. **Gustavo (Datos):**
   - MongoDB Compass (Cliente de escritorio para ver la BD).
   - Cuenta en MongoDB Atlas.

🏛️ FASE 1: BASE DE DATOS (MongoDB Atlas)

Responsable: Gustavo **Objetivo:** Crear el servidor en la nube y la estructura de datos.

1.1 Creación del Cluster (Paso a Paso)

1. Ir a mongodb.com/atlas y crear cuenta gratuita.
2. Hacer clic en **+ Create** (Crear nuevo cluster).
3. Seleccionar opción **M0 Sandbox** (GRATIS, sin tarjeta de crédito).
4. Proveedor: **AWS**. Región: **N. Virginia (us-east-1)**.
5. Clic en **Create Deployment**.

1.2 Configuración de Seguridad (CRÍTICO)

Sin esto, Alexander no podrá conectarse.

1. En el menú lateral, ir a **Security > Database Access**.
    - Clic **Add New Database User**.
    - Username: `admin_spotlight`.
    - Password: `PasswordSeguro123`.
    - Role: "Atlas Admin" o "Read and write to any database".
    - Clic **Add User**.
2. En el menú lateral, ir a **Security > Network Access**.
    - Clic **Add IP Address**.
    - Clic en el botón **Allow Access from Anywhere** (`0.0.0.0/0`).
    - Clic **Confirm**. *(Esto permite que la App Móvil y la PC de Alex se conecten).*

1.3 Obtener la "Connection String"

1. Ir a **Deployment > Database**.
2. Clic en el botón **Connect**.
3. Seleccionar **Drivers**.
4. Copiar la cadena que aparece. Se verá así:
   `mongodb+srv://admin_spotlight:PasswordSeguro123@cluster0.p8qrs.mongodb.net/?retryWrites=true&w=majority`
5. **ACCIÓN:** Enviar este string a Alexander por WhatsApp/Slack.

1.4 Definición de Colecciones (Esquemas)

Gustavo debe entrar a **Browse Collections** y crear la base de datos `SpotLightDB` con estas 3 colecciones:

A. Colección: `projects`

JSON
```
{
```

```
  "title": "Eco-Dron",
  "category": "Sustentabilidad",
  "description": "Dron que planta árboles...",
  "videoUrl": "https://commondatastorage.googleapis.com/gtv-videos-
bucket/sample/BigBuckBunny.mp4",
  "members": ["Juan", "Ana"],
  "stats": {
    "averageScore": 0.0,
    "totalVotes": 0
  },
  "createdAt": "2026-02-01T10:00:00Z"
}
```

B. Colección: `evaluations`

JSON

```
{
  "projectId": "ID_DEL_PROYECTO_AQUI",
  "evaluatorId": "ID_DEL_USUARIO",
  "scores": {
    "innovation": 5,
    "functionality": 4,
    "ui": 5,
    "impact": 5
  },
  "finalScore": 19.0,
  "feedback": "Buen trabajo",
  "aiAnalysis": { "sentiment": "Positivo" }
}
```

C. Colección: `users` (Para login).

⚙️ FASE 2: BACKEND (.NET 8 API)

Responsable: Alexander **Objetivo:** Crear el cerebro que conecta BD, App y Web.

2.1 Creación del Proyecto

1. Abrir Visual Studio 2022.
2. Crear nuevo proyecto: **ASP.NET Core Web API**.
3. Nombre: `SpotLight.API`.
4. Framework: **.NET 8.0**.
5. Desmarcar "Use HTTPS" (para evitar problemas de certificados locales al inicio).

2.2 Instalación de Librerías (NuGet)

Abrir la consola del Administrador de Paquetes y ejecutar:

PowerShell
```
Install-Package MongoDB.Driver
Install-Package Swashbuckle.AspNetCore
```

2.3 Configuración (`appsettings.json`)

Abre este archivo y pega la conexión que te dio Gustavo:

JSON
```
{
  "SpotLightDatabase": {
    "ConnectionString": "TU_CADENA_DE_MONGO_ATLAS_AQUI",
    "DatabaseName": "SpotLightDB",
    "ProjectsCollectionName": "projects",
    "EvaluationsCollectionName": "evaluations",
    "UsersCollectionName": "users"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

2.4 Habilitar CORS (Para que la Web funcione)

En `Program.cs`, antes de `var app = builder.Build();`:

C#
```
builder.Services.AddCors(options =>
{
```

```
    options.AddPolicy("AllowAll", policy =>
        policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
});

// ... luego del build ...
app.UseCors("AllowAll");
```

FASE 3: APP MÓVIL (Flutter)

Responsables: Javier (Lógica) & Emily (Diseño) **Objetivo:** La interfaz para los Jueces.

3.1 Creación del Proyecto

1. Abrir terminal en carpeta de trabajo.
2. Ejecutar: `flutter create spot_light_app`.
3. Abrir carpeta en VS Code.

3.2 Dependencias (`pubspec.yaml`)

Agreguen estas líneas bajo `dependencies:`:

YAML
```yaml
dependencies:
  flutter:
    sdk: flutter
  http: ^1.1.0           # Conexión a API
  google_fonts: ^6.1.0   # Tipografía
  video_player: ^2.8.1   # Video
  chewie: ^1.7.0         # UI de Video
  flutter_rating_bar: ^4.0.1 # Estrellitas (Opcional)
  provider: ^6.1.1       # Gestión de estado simple
```

Luego ejecutar `flutter pub get` en la terminal.

3.3 Permisos de Internet (Android)

¡OJO! Si no hacen esto, la app se cierra al intentar conectar. Ir a `android/app/src/main/AndroidManifest.xml` y agregar justo antes de `<application>`:

XML
```xml
<uses-permission android:name="android.permission.INTERNET" />
```

3.4 Estructura de Carpetas

Plaintext
```
lib/
├── config/
│   └── theme.dart       (Definir Colores: Primary Blue #137fec)
├── models/
│   └── project.dart     (Clase idéntica a la de C#)
├── services/
│   └── api_service.dart (Donde vive el http.get)
├── screens/
│   ├── login_screen.dart
│   ├── home_screen.dart (Lista de proyectos)
│   ├── detail_screen.dart (Video)
```

```
│   └── evaluate_screen.dart (Formulario)
└── main.dart
```

3.5 Configuración de IP (El error común)

En `api_service.dart`, la URL base depende de dónde prueben:

- Si usan **Emulador Android**: `static const baseUrl = "http://10.0.2.2:5000/api";`
- Si usan **Celular Físico**: `static const baseUrl = "http://192.168.1.X:5000/api";` (La IP de la compu de Alexander).

💻 FASE 4: WEB ADMIN (HTML + JS)

Responsables: Gustavo & Emily **Objetivo:** Panel rápido para subir proyectos.

4.1 Estructura

Carpeta `SpotLight.Web/`:

- `index.html` (Login falso / Entrada)
- `admin.html` (Dashboard)
- `js/app.js` (Lógica)

4.2 Código Base (`admin.html`)

Usaremos Tailwind vía CDN para no instalar nada complejo.

HTML
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Spot-Light Admin</title>
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-slate-900 text-white p-10">
    <h1 class="text-3xl font-bold mb-5">Panel de Administración</h1>

    <div class="bg-slate-800 p-6 rounded-xl mb-10">
        <h2 class="text-xl mb-4">Nuevo Proyecto</h2>
        <input type="text" id="title" placeholder="Título" class="w-full
p-2 mb-2 bg-slate-700 rounded">
        <input type="text" id="video" placeholder="URL Video (mp4)"
class="w-full p-2 mb-2 bg-slate-700 rounded">
        <button onclick="crearProyecto()" class="bg-blue-500 px-4 py-2
rounded font-bold">Guardar</button>
    </div>

    <div id="lista-proyectos" class="grid grid-cols-1 md:grid-cols-3 gap-
4">
        </div>

    <script src="js/app.js"></script>
</body>
</html>
```

4.3 Lógica JS (`js/app.js`)

JavaScript
```
const API_URL = "http://localhost:5000/api"; // URL de Alexander

async function crearProyecto() {
```

```
    const title = document.getElementById('title').value;
    const videoUrl = document.getElementById('video').value;

    const data = {
        title: title,
        videoUrl: videoUrl,
        category: "General",
        members: []
    };

    await fetch(`${API_URL}/projects`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data)
    });

    alert("Proyecto Creado!");
    cargarProyectos(); // Recargar lista
}

// ... función cargarProyectos() ...
```

🚦 RESUMEN DEL FLUJO DE TRABAJO

1. **Día 1:** Gustavo crea la BD en Atlas y pasa el string.
2. **Día 1:** Alexander crea la API y pone ese string en `appsettings.json`.
3. **Día 2:** Alexander corre la API (F5). Le sale una pantalla Swagger.
4. **Día 2:** Javier configura Flutter apuntando a la IP de Alexander.
5. **Día 3:** Gustavo abre `admin.html`, llena el formulario y da clic en "Guardar".
6. **Resultado:** El dato viaja a la API de Alex -> Se guarda en Mongo Atlas -> Javier recarga la App en el celular y ¡PUM! Aparece el proyecto.

Este manual cubre el **100%** de los aspectos técnicos. No hay excusa para no empezar. ¡Éxito equipo!