



EVIDENCIA DE PROMPTS

Orlando Méndez Montes

Evidencia de Prompts e Interacciones con IA

Proyecto: CineVerse (Cinemapedia)

Estudiante: Orlando Mendez Montes

Matrícula: 220512

Fecha: 12 de noviembre de 2025

Propósito: Documentación completa de asistencia con IA durante el desarrollo del Examen Práctico U2

Índice de Interacciones

1. [Configuración Inicial del Repositorio](#)
 2. [Implementación del Splash Screen](#)
 3. [Pantalla de Carga Progresiva](#)
 4. [Mejoras Visuales del Swiper](#)
 5. [Configuración de Iconos y Branding](#)
 6. [Corrección de Bug de Carga Rápida](#)
 7. [Mejoras Finales de Diseño](#)
-

1. Configuración Inicial del Repositorio

Prompt del Usuario:

```
necesito subir mi practica a un repositorio de github, el repositorio ya esta  
creado y se llama  
DMI_ExamenPractico_U2_220512, mi usuario es MendezPro
```

Respuesta de la IA:

La IA proporcionó comandos específicos de Git para inicializar el repositorio, agregar archivos, hacer commit y push:

```
git init  
git add .
```

```
git commit -m "Initial commit: Práctica 05 - Cinemapedia base"
git branch -M main
git remote add origin
https://github.com/MendezPro/DMI_ExamenPractico_U2_220512.git
git push -u origin main
```

Resultado: Repositorio inicializado exitosamente con la práctica base subida.

2. Implementación del Splash Screen

👤 Prompt del Usuario:

Ayudame a realizar mis actividades explicando paso a paso la creación de las mismas para su posterior funcionamiento correcto de la aplicación

🤖 Respuesta de la IA:

Paso 1: Agregar Dependencias

La IA identificó las dependencias necesarias y las agregó al `pubspec.yaml`:

```
dependencies:
  audioplayers: ^6.1.0
  animate_do: ^4.2.0
  intl: ^0.20.2
```

Paso 2: Configurar Assets

```
assets:
  - assets/images/
  - assets/audio/
```

Paso 3: Crear SplashScreen

La IA generó el archivo

completo `lib/presentation/screens/splash/splash_screen.dart`:

```
import 'package:animate_do/animate_do.dart';
import 'package:audioplayers/audioplayers.dart';
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';

class SplashScreen extends StatefulWidget {
  static const name = 'splash';
  const SplashScreen({super.key});
```

```
@override
State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
final AudioPlayer _audioPlayer = AudioPlayer();

@Override
void initState() {
super.initState();
_playIntroAudio();
}

Future<void> _playIntroAudio() async {
try {
await _audioPlayer.play(AssetSource('audio/introaudio.mp3'));

_audioPlayer.onPlayerComplete.listen((_) {
if (mounted) {
context.go('/home');
}
});
}

Future.delayed(const Duration(seconds: 3), () {
if (mounted) {
context.go('/home');
}
});
} catch (e) {
Future.delayed(const Duration(seconds: 3), () {
if (mounted) {
context.go('/home');
}
});
}
}

@Override
Widget build(BuildContext context) {
return Scaffold(
body: Container(
decoration: BoxDecoration(
gradient: LinearGradient(
begin: Alignment.topCenter,
end: Alignment.bottomCenter,
colors: [
Theme.of(context).colorScheme.primary.withOpacity(0.8),
Theme.of(context).colorScheme.secondary.withOpacity(0.6),
Colors.black,
],
),
),
),
child: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
```

```

        children: [
          FadeIn(
            duration: const Duration(milliseconds: 1500),
            child: ZoomIn(
              duration: const Duration(milliseconds: 1500),
              child: Image.asset(
                'assets/images/LogoCineVerse.png',
                width: 250,
                height: 250,
              ),
            ),
          ),
        ],
        const SizedBox(height: 30),
        FadeIn(
          delay: const Duration(milliseconds: 800),
          child: const CircularProgressIndicator(
            color: Colors.white,
            strokeWidth: 3,
          ),
        ),
      ],
    ),
  ),
),
);
},
);
},
);
},
);
}
}

@Override
void dispose() {
  _audioPlayer.dispose();
  super.dispose();
}
}

```

Paso 4: Configurar Router

La IA modificó el `app_router.dart` para incluir el `SplashScreen` como ruta inicial:

```

GoRoute(
  path: '/splash',
  name: SplashScreen.name,
  builder: (context, state) => const SplashScreen(),
),

```

Resultado: Splash Screen implementado con animación `FadeIn/ZoomIn`, reproducción de audio y navegación automática.

3. Pantalla de Carga Progresiva

Prompt del Usuario:

Ayudame a crear la pantalla de carga progresiva: en donde exista el FullscreenLoader mostrando porcentaje numérico (0%-100%) con LinearProgressIndicator animado.

Respuesta de la IA:

La IA rediseñó completamente el archivo lib/presentation/widgets/shared/fullscreen_loader.dart:

Características Implementadas:

1. **Stream de Progreso:** 14 pasos de carga con intervalos de 1.2 segundos
2. **Porcentaje Visual:** Texto grande (72px) mostrando 0-100%
3. **Barra de Progreso:** LinearProgressIndicator animado de 18px de altura
4. **Mensajes Descriptivos:** Sincronizados con cada etapa de carga

```
Stream<Map<String, dynamic>> getLoadingProgress() {
  final progressSteps = [
    {'message': 'Iniciando CineVerse...', 'progress': 0.0},
    {'message': 'Verificando conexión a internet', 'progress': 0.05},
    {'message': 'Estableciendo conexión segura', 'progress': 0.12},
    {'message': 'Conectando con TheMovieDB API', 'progress': 0.20},
    {'message': 'Autenticando credenciales', 'progress': 0.28},
    {'message': 'Validando permisos de acceso', 'progress': 0.35},
    {'message': 'Obteniendo películas en cartelera', 'progress': 0.45},
    {'message': 'Descargando imágenes de películas', 'progress': 0.55},
    {'message': 'Cargando próximos estrenos', 'progress': 0.65},
    {'message': 'Descargando películas mejor valoradas', 'progress': 0.73},
    {'message': 'Cargando películas mexicanas', 'progress': 0.82},
    {'message': 'Organizando catálogo completo', 'progress': 0.90},
    {'message': 'Preparando interfaz visual', 'progress': 0.96},
    {'message': '¡Todo listo! Bienvenido a CineVerse', 'progress': 1.0},
  ];

  return Stream.periodic(const Duration(milliseconds: 1200), (step) {
    if (step < progressSteps.length) {
      return progressSteps[step];
    }
    return progressSteps.last;
  }).take(progressSteps.length);
}
```

Widget de Visualización:

```
// Porcentaje grande y visible
TweenAnimationBuilder<double>(
  duration: const Duration(milliseconds: 500),
  tween: Tween(begin: 0.0, end: progress),
  builder: (context, value, child) {
    return Text(
```

```

        '${(value * 100).toInt()}',
        style: TextStyle(
            fontSize: 72,
            fontWeight: FontWeight.bold,
            color: colors.primary,
            letterSpacing: -2,
        ),
    ),
),
},
),
),

// Barra de progreso animada
SizedBox(
    width: 300,
    child: LinearProgressIndicator(
        value: progress,
        backgroundColor: colors.secondary.withOpacity(0.2),
        valueColor: AlwaysStoppedAnimation<Color>(colors.primary),
        minHeight: 18,
        borderRadius: BorderRadius.circular(10),
    ),
),
),

```

Resultado: Loading screen con porcentaje visible (0-100%), barra de progreso animada y mensajes descriptivos.

4. Mejoras Visuales del Swiper

👤 Prompt del Usuario:

```

verifica que esto este bien... mi carrusel
[Usuario solicitó revisar y mejorar el diseño visual del Swiper]

```

🤖 Respuesta de la IA:

La IA propuso mejoras profesionales para el Swiper en `movie_slidershow.dart`:

Mejoras Implementadas:

1. **Altura Aumentada:** De 250px a 280px
2. **Viewport Fraction:** 0.85 para mejor efecto visual
3. **Autoplay Delay:** 5 segundos entre slides
4. **Clasificación con Iconos:**

```

Map<String, dynamic> _getRatingStyle(String rating) {
    switch (rating) {
        case 'R':
            return {

```

```
        'color': const Color(0xFFDC143C), // Crimson
        'icon': Icons.block,
    );
case 'PG-13':
    return {
        'color': const Color(0xFFFF8C00), // Orange
        'icon': Icons.warning_amber_rounded,
    };
case 'PG':
    return {
        'color': const Color(0xFF4169E1), // Royal Blue
        'icon': Icons.family_restroom,
    };
default:
    return {
        'color': const Color(0xFF32CD32), // Lime Green
        'icon': Icons.check_circle,
    };
}
}
```

5. Formato de Fecha en Español:

```
final dateFormat = DateFormat('d \'de\' MMMM, yyyy', 'es');
```

6. Triple Gradiente para Legibilidad:

```
gradient: LinearGradient(
begin: Alignment.topCenter,
end: Alignment.bottomCenter,
colors: [
    Colors.black.withOpacity(0.1),
    Colors.black.withOpacity(0.4),
    Colors.black.withOpacity(0.9),
],
stops: const [0.3, 0.6, 1.0],
),
```

7. Sistema de Calificación con Estrellas:

```
Row(
children: [
...List.generate(5, (index) {
    final rating = movie.voteAverage / 2;
    if (index < rating.floor()) {
        return const Icon(Icons.star, color: Color(0xFFFFD700), size: 16);
    } else if (index < rating) {
        return const Icon(Icons.star_half, color: Color(0xFFFFD700), size: 16);
    } else {
        return Icon(Icons.star_border, color: Colors.white.withOpacity(0.5),
size: 16);
    }
}),
const SizedBox(width: 8),
Text('${movie.voteAverage.toStringAsFixed(1)}/10'),
],
)
```

8. Efecto Vignette:

```
Container(  
  decoration: BoxDecoration(  
    gradient: RadialGradient(  
      center: Alignment.center,  
      radius: 1.0,  
      colors: [  
        Colors.transparent,  
        Colors.black.withOpacity(0.4),  
      ],  
      stops: const [0.5, 1.0],  
    ),  
  ),  
)
```

Resultado: Swiper con diseño profesional incluyendo clasificación por colores e iconos, estrellas de rating, fecha formateada y mejor legibilidad.

5. Configuración de Iconos y Branding

👤 Prompt del Usuario:

```
quita el icono de flutter y pon el mio este de aqui  
[Usuario compartió LogoCineVerse.png]
```

🤖 Respuesta de la IA:

Paso 1: Agregar Dependencia

```
dev_dependencies:  
  flutter_launcher_icons: ^0.13.1
```

Paso 2: Configurar pubspec.yaml

```
flutter_launcher_icons:  
  android: true  
  ios: true  
  image_path: "assets/icon/LogoCineVerse.png"  
  adaptive_icon_background: "#191919"  
  adaptive_icon_foreground: "assets/icon/LogoCineVerse.png"  
  web:  
    generate: true  
    image_path: "assets/icon/LogoCineVerse.png"  
  windows:  
    generate: true  
    image_path: "assets/icon/LogoCineVerse.png"  
  macos:
```

```
generate: true
image_path: "assets/icon/LogoCineVerse.png"
```

Paso 3: Ejecutar Generación

```
flutter pub get
dart run flutter_launcher_icons
```

Paso 4: Cambiar Nombre de la App

La IA modificó múltiples archivos de configuración:

AndroidManifest.xml:

```
<application
    android:label="CineVerse"
    ...>
```

iOS Info.plist:

```
<key>CFBundleName</key>
<string>CineVerse</string>
```

Web index.html:

```
<title>CineVerse</title>
```

Web manifest.json:

```
{
  "name": "CineVerse",
  "short_name": "CineVerse"
}
```

Resultado: Icono personalizado configurado para todas las plataformas y nombre de app cambiado a "CineVerse".

6. Corrección de Bug de Carga Rápida

 **Prompt del Usuario:**

```
el loading me pasa muy rapido y no se muestran nada los porcentajes del 0 al 100
quiero que se vea
completo aunque ya haya cargado todo
```

 **Respuesta de la IA:**

La IA identificó que el problema era que los providers cargaban muy rápido y el loading desaparecía antes de completar el ciclo visual.

Solución Implementada:

Modificación en `home_screen.dart`:

```
class _HomeViewState extends ConsumerState<_HomeView> {
    bool _isLoading = true;
    bool _minTimeElapsed = false;
    bool _dataLoaded = false;

    @override
    void initState() {
        super.initState();

        // Timer mínimo de 17 segundos garantizado
        Future.delayed(const Duration(milliseconds: 17000), () {
            if (mounted) {
                setState(() {
                    _minTimeElapsed = true;
                });
                _checkDataLoading();
            }
        });

        // Polling cada 500ms para verificar si los datos están listos
        _checkDataLoading();
    }

    void _checkDataLoading() {
        if (!mounted) return;

        final nowPlayingMovies = ref.read(nowPlayingMoviesProvider);
        final popularMovies = ref.read(popularMoviesProvider);
        final upcomingMovies = ref.read(upcomingMoviesProvider);
        final mexicanMovies = ref.read(mexicanMoviesProvider);
        final topRatedMovies = ref.read(topRatedMoviesProvider);

        // Verificar si TODOS los providers tienen datos
        if (nowPlayingMovies.isNotEmpty &&
            popularMovies.isNotEmpty &&
            upcomingMovies.isNotEmpty &&
            mexicanMovies.isNotEmpty &&
            topRatedMovies.isNotEmpty) {
            _dataLoaded = true;
        }

        // Solo ocultar loading cuando AMBAS condiciones se cumplan
        if (_minTimeElapsed && _dataLoaded) {
            setState(() {
                _isLoading = false;
            });
        } else if (!_minTimeElapsed || !_dataLoaded) {
            // Seguir verificando cada 500ms
            Future.delayed(const Duration(milliseconds: 500), _checkDataLoading);
        }
    }
}
```

```

    }

    @override
    Widget build(BuildContext context) {
      if (_isLoading) {
        return const FullscreenLoader();
      }

      // ... resto del código
    }
}

```

Lógica de Control:

1. Timer de 17 segundos mínimo garantizado
2. Polling cada 500ms para verificar datos de providers
3. Loading solo desaparece cuando AMBAS condiciones se cumplen:
 - Timer de 17s completado
 - Todos los providers tienen datos

Resultado: Loading screen ahora muestra el progreso completo de 0% a 100% visible, sin importar qué tan rápido carguen los datos.

7. Mejoras Finales de Diseño

👤 Prompt del Usuario:

cambiale el nombre para que en la pantalla de carga aparezca CineVerse y no diga Cinemapedia

🤖 Respuesta de la IA:

Modificación en `fullscreen_loader.dart`:

```

// Antes:
Text(
  'Bienvenido a Cinemapedia',
  style: TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
    color: colors.primary,
  ),
),

// Despues:
Text(
  'Bienvenido a CineVerse',

```

```
        style: TextStyle(  
          fontSize: 24,  
          fontWeight: FontWeight.bold,  
          color: colors.primary,  
        ),  
      ),
```

Resultado: Pantalla de carga ahora muestra "Bienvenido a CineVerse" en lugar de "Cinemapedia".

Resumen de Tecnologías y Paquetes Utilizados

Paquete	Versión	Uso en el Proyecto
audioplayers	^6.1.0	Reproducir audio de bienvenida en SplashScreen
animate_do	^4.2.0	Animaciones FadeIn y ZoomIn en SplashScreen
card_swiper	^3.0.1	Carrusel de películas con efecto de escala
intl	^0.20.2	Formato de fechas en español ("d 'de' MMMM, yyyy")

Paquete	Versión	Uso en el Proyecto
<code>flutter_launcher_icons</code>	<code>^0.13.1</code>	Generación de iconos para todas las plataformas
<code>go_router</code>	(ya incluido)	Navegación entre screens (splash → home)
<code>flutter_riverpod</code>	<code>^3.0.3</code>	Gestión de estado y providers de películas

⌚ Funcionalidades Completadas con Asistencia de IA

☑ Implementaciones Exitosas:

1. Splash Screen Animado

- Animación FadeIn/ZoomIn con duración de 1.5s
- Reproducción de audio de bienvenida
- Navegación automática después de 3 segundos
- Logo personalizado centrado

2. Loading Screen Progresivo

- Porcentaje visual de 0% a 100% con fuente de 72px
- 14 mensajes descriptivos sincronizados
- Barra LinearProgressIndicator animada
- Duración mínima garantizada de 17 segundos
- Gradiente de fondo personalizado

3. Swiper Mejorado

- Altura de 280px con viewport fraction de 0.85
- Clasificación visual con píldoras de colores (R, PG-13, PG, P)
- Iconos personalizados por clasificación
- Sistema de estrellas para rating (5 estrellas)
- Fecha de estreno formateada en español
- Triple gradiente para legibilidad de texto
- Efecto vignette radial
- Sombras profesionales con color tinting
- Autoplay de 5 segundos entre slides

4. Branding Completo

- Icono personalizado en todas las plataformas
- Nombre "CineVerse" en todas las configuraciones
- Logo consistente en splash y launcher

5. Corrección de Bugs

- Loading demasiado rápido → Timer mínimo de 17s
- Nombre inconsistente → Cambio global a "CineVerse"
- Legibilidad del texto en Swiper → Triple gradiente + vignette



Comandos de Terminal Ejecutados

Instalación de Dependencias:

```
flutter pub get
```

Generación de Iconos:

```
dart run flutter_launcher_icons
```

Ejecución de la App:

```
flutter run
```

Comandos Git:

```
git init
git add .
git commit -m "feat: implementar splash screen con animación y audio"
git commit -m "feat: implementar loading screen progresivo con porcentaje"
git commit -m "feat: mejorar swiper con título, fecha y clasificación"
git commit -m "fix: corregir timing del loading screen para mostrar progreso
completo"
git commit -m "feat: configurar iconos personalizados y cambiar nombre a
CineVerse"
```

```
git push origin main
```

🔍 Archivos Creados y Modificados

Archivos Creados:

- lib/presentation/screens/splash/splash_screen.dart
- assets/images/LogoCineVerse.png
- assets/icon/LogoCineVerse.png
- assets/audio/introaudio.mp3
- PROMPTS_EVIDENCIA.md (este documento)

Archivos Modificados:

- pubspec.yaml - Dependencias y assets
 - lib/config/router/app_router.dart - Rutas del splash
 - lib/presentation/widgets/shared/fullscreen_loader.dart - Loading progresivo
 - lib/presentation/widgets/movies/movie_slideshow.dart - Mejoras visuales
 - lib/presentation/screens/movies/home_screen.dart - Control de timing
 - lib/presentation/providers/movies/initialLoanding_provider.dart - Simplificación
 - android/app/src/main/AndroidManifest.xml - Nombre de app
 - ios/Runner/Info.plist - Nombre de app
 - web/index.html - Nombre de app
 - web/manifest.json - Nombre de app
 - README.md - Documentación del proyecto
-

💡 Lecciones Aprendidas

Desafíos Técnicos Superados:

1. **Sincronización de Timing:**
 - Problema: Loading pasaba instantáneamente
 - Solución: Implementación de timer mínimo + polling de datos
 - Aprendizaje: No depender solo de la velocidad de los providers
2. **Gestión de Audio:**

- Problema: Sincronizar navegación con finalización de audio
 - Solución: Listener `onPlayerComplete` + timeout de seguridad
 - Aprendizaje: Siempre tener fallback para eventos multimedia
3. **Legibilidad de Texto sobre Imágenes:**
 - Problema: Texto blanco no se leía bien en algunas imágenes
 - Solución: Triple gradiente + vignette + sombras múltiples
 - Aprendizaje: Combinar técnicas para garantizar legibilidad
 4. **Configuración Multi-plataforma:**
 - Problema: Icono debía funcionar en 6 plataformas diferentes
 - Solución: `flutter_launcher_icons` con configuración centralizada
 - Aprendizaje: Usar herramientas especializadas ahorra tiempo
-

Próximos Pasos Sugeridos por la IA

1. **Optimización de Rendimiento:**
 - Implementar cache de imágenes
 - Lazy loading de películas adicionales
 - Precarga de assets del splash
 2. **Mejoras UX:**
 - Animación de transición entre splash y home
 - Skeleton loaders en lugar de spinners genéricos
 - Animaciones en los cambios de slide del swiper
 3. **Funcionalidades Adicionales:**
 - Búsqueda de películas
 - Filtros por género y año
 - Sección de favoritos con persistencia local
 - Integración de trailers desde YouTube
 4. **Testing:**
 - Unit tests para providers
 - Widget tests para screens principales
 - Integration tests para flujo completo
-

Estructura Final del Proyecto

```
DMI_ExamenPractico_U2_220512/
├── lib/
│   ├── main.dart
│   └── config/
│       └── constants/
```

```
    └── helpers/
    └── router/
        └── app_router.dart ✎
    └── theme/
    └── domain/
        ├── datasources/
        ├── entities/
        └── repositories/
    └── infrastructure/
        ├── datasources/
        ├── mappers/
        ├── models/
        └── repositories/
    └── presentation/
        ├── providers/
            └── movies/
                └── initialLoanding_provider.dart ✎
        └── screens/
            ├── splash/
                └── splash_screen.dart ♦ NUEVO
            └── movies/
                └── home_screen.dart ✎
        └── widgets/
            ├── shared/
                └── fullscreen_loader.dart ✎
            └── movies/
                └── movie_slidershow.dart ✎
    └── assets/
        ├── images/
            └── LogoCineVerse.png ♦ NUEVO
        ├── icon/
            └── LogoCineVerse.png ♦ NUEVO
        └── audio/
            └── introaudio.mp3 ♦ NUEVO
    └── android/ ✎
    └── ios/ ✎
    └── web/ ✎
    └── pubspec.yaml ✎
    └── README.md ✎
    └── PROMPTS_EVIDENCIA.md ♦ NUEVO
```

♦ = Archivo nuevo creado

✎ = Archivo modificado

🎬 Conclusión

Este documento evidencia todas las interacciones con IA durante el desarrollo del examen práctico. La asistencia de IA fue fundamental para:

- **Acelerar el desarrollo** de funcionalidades complejas
- **Implementar mejores prácticas** de diseño y arquitectura
- **Resolver problemas técnicos** de manera eficiente
- **Mejorar la calidad visual** del proyecto

Todas las implementaciones fueron comprendidas, adaptadas y validadas por el estudiante, utilizando la IA como herramienta de apoyo educativo y no como sustituto del aprendizaje.

 **Documento generado:** 12 de noviembre de 2025

 **Estudiante:** Orlando Mendez Montes (220512)

 **Materia:** Desarrollo Móvil Integral