答辩文档 副本--项目文档

仓库链接：https://github.com/lingxiao001/Dou-demo/tree/native-first

branch：native-first

一. 项目概览

1.1 项目简介

本项目是一个 **Android Native (Kotlin) + Flutter 混合开发** 的仿抖音客户端 Demo。

项目旨在探索原生 Android 与 Flutter 混合开发的最佳实践，结合了原生开发的高性能列表渲染能力和 Flutter 的高效 UI 构建能力。

- **核心目标**：实现高性能的双列瀑布流视频外流页面（Native）与视频内流播放页面（Flutter + Native Player）。

1.2 核心特性

双列外流 (视频瀑布流界面)：

- 使用原生 `RecyclerView` + `StaggeredGridLayoutManager` 实现高性能双列瀑布流。
- 支持无限滚动加载、下拉刷新。
- 使用 `Coil` 进行高效图片加载与内存管理。
- 使用 `RecyclerViewPoolManager` 优化视图复用。

混合架构：

本项目MVP使用flutter开发，后期改为**原生为主**，flutter仅在视频内流UI与ai悬浮窗处使用。

首页采用 **Native** 实现，保证首屏加载速度和列表滑动流畅度。

视频详情页采用 **Flutter** (`FlutterActivity`) 承载，利用 Flutter 快速构建复杂的交互逻辑（如上下滑动切换、评论面板）。

原生播放器引擎：

视频播放核心采用 **ExoPlayer (Media3)**。

通过 **Platform View (`AndroidView`)** 将原生播放器嵌入 Flutter 内流页面中。

PlayerPool：全局播放器复用池，支持秒开、无缝切换，最多同时维护 3 个播放器实例。

二. 技术栈

2.1 Android Native (Kotlin)

- **UI 组件**：`AppCompatActivity`, `Fragment`, `ViewPager2`, `TabLayout`, `RecyclerView`
- **视频播放**：`androidx.media3.exoplayer` (ExoPlayer)
- **图片加载**：`io.coil-kt:coil`
- **通信**：`MethodChannel` (与 Flutter 通信)

2.2 Flutter (Dart)

* **容器**：`FlutterActivity`
* **渲染**：`AndroidView` (Platform Views)

三. 目录结构

3.1 Android Native (`android/app/src/main/kotlin/...`)

代码块

```
com/example/douyin_demo/
├── NativeHomeActivity.kt     # [Entry] 原生主入口，负责 Tab 管理与 Fragment 调度
├── NativeFeedFragment.kt     # [UI] 双列瀑布流页面，处理列表滚动与数据加载
├── MainActivity.kt           # [Entry] Flutter 容器 Activity，承载视频详情页
│
├── core部分
│   ├── PlayerPool.kt             # [Singleton] ExoPlayer 全局复用池
│   └── RecyclerViewPoolManager.kt # [Singleton] RecyclerView 视图复用池
│
├── views部分
│   ├── NativeVideoView.kt        # [PlatformView] 封装 ExoPlayer 供 Flutter 调用
│   └── NativeFeedView.kt         # [Adapter] FeedAdapter & FeedVH 实现
│
└── ProfileActivity.kt        # [UI] 个人中心页面 (Native 实现)
```

3.2 Flutter (`lib/`)

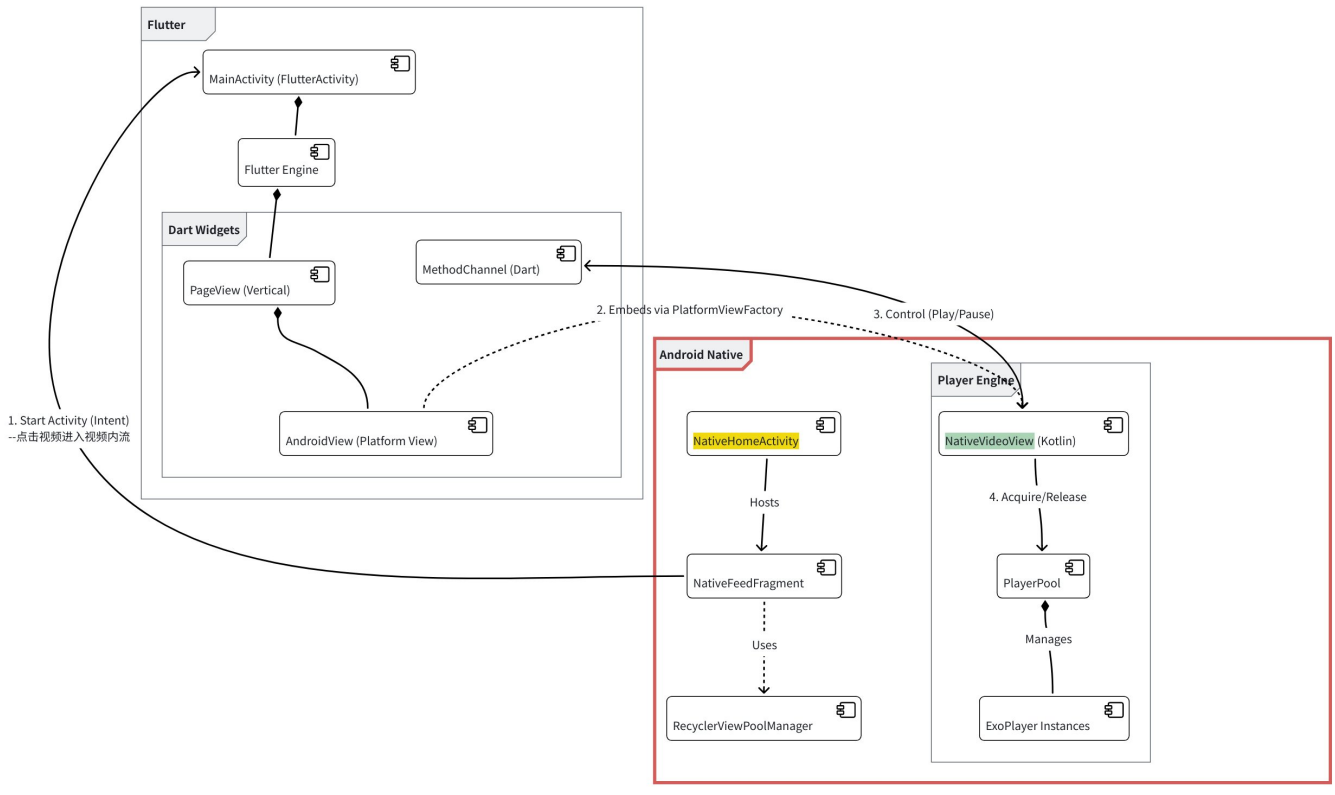
Flutter 侧负责视频播放详情页的交互逻辑。

代码块

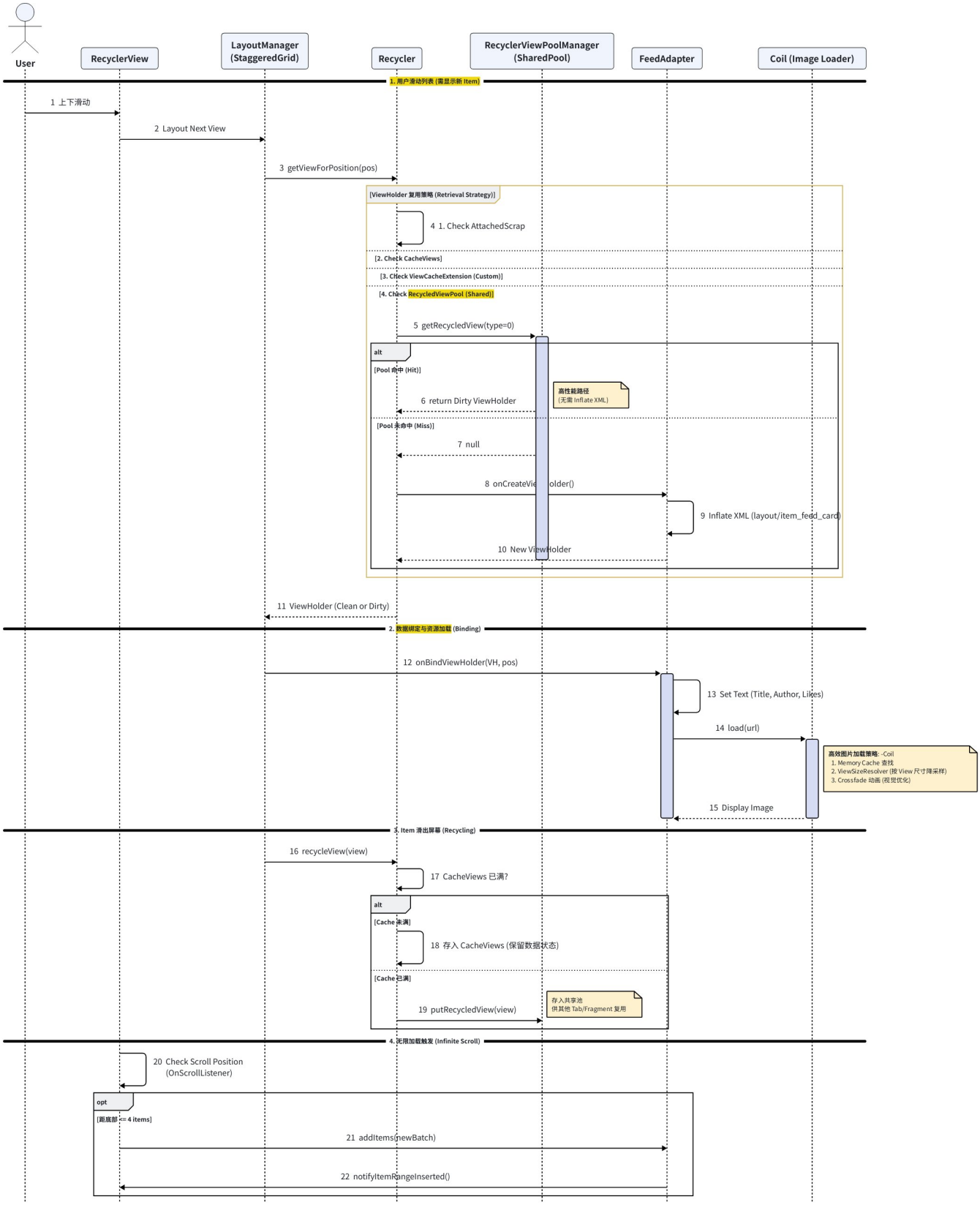
```
lib/
├── main.dart                  # [Entry] Flutter 入口
├── router/                    # 路由配置 (go_router)
├── pages/
│   └── video_viewer_page.dart   # [Page] 视频详情页 (PageView)
└── widgets/
    ├── native_video_player.dart # [Widget] 对应 NativeVideoView 的 Dart 封装
    └── comment_sheet.dart       # [Widget] 评论面板
```

四. 架构设计

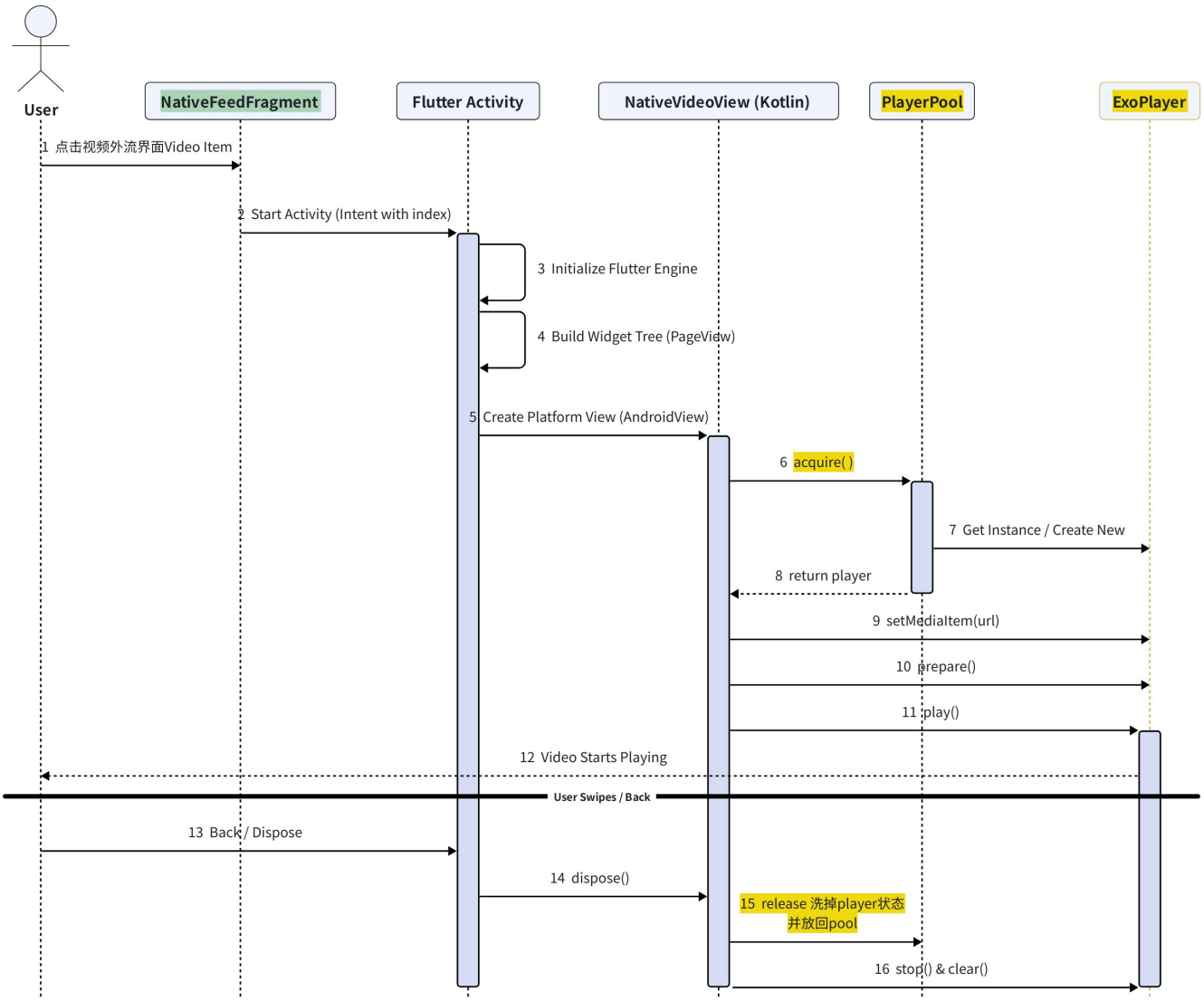
1. 系统组件关系图



2. 视频外流加载时序图


User
RecyclerView
LayoutManager (StaggeredGrid)
Recycler
RecyclerViewPoolManager (SharedPool)
FeedAdapter
Coil (Image Loader)
1. 用户滑动列表 (需显示新 Item)
1 上下滑动
2 Layout Next View
3 getViewForPosition(pos)
[ViewHolder 复用策略 (Retrieval Strategy)]
4 1. Check AttachedScrap
[2. Check CacheViews]
[3. Check ViewCacheExtension (Custom)]
[4. Check RecycledViewPool (Shared)]
5 getRecycledView(type=0)
alt
[Pool 命中 (Hit)]
高性能路径
(无需 Inflate XML)
6 return Dirty ViewHolder
[Pool 未命中 (Miss)]
7 null
8 onCreateVi older()
9 Inflate XML (layout/item_feed_card)
10 New ViewHolder
11 ViewHolder (Clean or Dirty)
2. 数据绑定与资源加载 (Binding)
12 onBindViewHolder(VH, pos)
13 Set Text (Title, Author, Likes)
14 load(url)
高效图片加载策略: -Coil
1. Memory Cache 查找
2. ViewSizeResolver (按 View 尺寸降采样)
3. Crossfade 动画 (视觉优化)
15 Display Image
3. Item 滑出屏幕 (Recycling)
16 recycleView(view)
17 CacheViews 已满?
alt
[Cache 未满]
18 存入 CacheViews (保留数据状态)
[Cache 已满]
存入共享池
供其他 Tab/Fragment 复用
19 putRecycledView(view)
4. 无限加载触发 (Infinite Scroll)
20 Check Scroll Position
(OnScrollListener)
opt
[距底部 <= 4 items]
21 addItems(newBatch)
22 notifyItemRangeInserted()

3. 视频播放器时序图



五. 总结

5.1 核心难点与解决方案

难点	现象	解决方案
视频起播慢	进入详情页或上下滑动切换视频时，画面短暂变黑。	**PlayerPool (复用池)**： 并不销毁 ExoPlayer，而是重置后复用。利用 `PlayerPool` 快速获取播放器实例，结合 Flutter 的预渲染机制减少等待时间。
内存占用过高	瀑布流加载大量高清封面导致 OOM 或卡顿。	**Coil 图片优化**： 使用 ViewSizeResolver 仅加载控件实际大小的图片
多 Tab 列表卡顿	切换底部 Tab 时，Fragment 重建导致列表重新 Inflate。	**RecyclerViewPoolManager**： 全局共享 RecycledViewPool，让不同 Fragment 复用相同的 ViewHolder，大幅减少视图创建开销。