01.功能概述

- **功能ID**：`refactor-20250625-001`
- **功能名称**：OTA弹窗
- **目标版本**：v0.2.0
- **提交人**：@panruiqi
- **状态**：
  - [x] ⏳ 设计中 /
  - [ ] ⏳ 开发中 /
  - [ ] ☑ 已完成 /
  - [ ] ❌ 已取消
- **价值评估**：
  - [x] ★★★★★ 核心业务功能
  - [ ] ★★★★ 用户体验优化
  - [ ] ★★★ 辅助功能增强
  - [ ] ★★ 技术债务清理
- **功能描述**
  - OTA弹窗独立于弹窗管理系统，我们希望将他加入到里面进行管理

02.需求分析

2.1 用户场景

- **主要场景**:

- **边界场景**:

2.2 功能范围

- ☑ 包含:
- ❌ 不包含:

03.技术方案

3.0 可行性分析

☑ 支持放入popup feature的理由

- 统一弹窗管理
  - 所有对话框都在一个地方管理，便于维护
  - 可以统一处理弹窗优先级、队列、生命周期
- 避免弹窗冲突
  - popup feature可以管理多个弹窗的显示顺序
  - 避免OTA弹窗和其他弹窗同时显示
- 复用弹窗基础设施
  - 复用弹窗的基础组件、动画、样式等
  - 统一的弹窗配置管理

❌ 不建议的理由

- 业务职责不匹配
  - OTA是系统级更新功能，业务逻辑复杂
  - popup更多是UI层的通用弹窗管理
- 生命周期差异
  - OTA弹窗可能需要跨Activity显示
  - 普通popup通常和页面生命周期绑定
- 依赖关系
  - OTA功能可能依赖网络、文件下载、安装权限等
  - popup feature引入这些依赖会变得臃

3.1 方案一

- 实现思路：分层设计，推荐
  - ```
    OTA Feature (业务层)
        ↓ 调用
    Popup Feature (UI层)
    ```
  - OTA Feature：负责更新检查、下载、业务逻辑
    ```

- Popup Feature：只负责弹窗的UI展示和管理
- OTA通过接口调用popup来显示弹窗

3.2 方案二

- 实现思路：完全独立
  - OTA和popup完全分离
  - 各自管理自己的弹窗逻辑
  - 也就是现有的逻辑

3.3 方案三

- 实现思路：完全合并
  - 把OTA相关的dialog都放到popup中
  - 需要popup feature承担更多业务责任

04.实现规划

4.1 技术选型

推荐方案一：

- popup feature提供通用的弹窗管理能力（队列、优先级、基础组件）
- OTA feature负责具体的更新业务逻辑
- 通过接口解耦，各司其职

4.2 任务拆解

- 扩展PopupType（在PopupType.kt中添加）
  - ```
    enum class PopupType(val priority: Int, val category: String) {
        // ... 现有类型 ...

        // OTA更新类弹窗 - 高优先级，系统级
        DIALOG_OTA_NORMAL_UPDATE(4, "ota_dialog"),      // 普通更新
        DIALOG_OTA_FORCE_UPDATE(5, "ota_dialog"),       // 强制更新
        DIALOG_OTA_UPDATING(5, "ota_dialog")            // 更新中
    }
    ```

- 在PopupFactory中添加OTA工厂方法
  - ```
    // 在PopupFactory.kt中添加
    object PopupFactory {

        /**
         * 创建OTA普通更新弹窗
         */
        fun createOtaNormalUpdatePopup(
            version: String,
            buildNumber: String,
            updateContent: String,
            onConfirm: () -> Unit,
            onCancel: () -> Unit
        ): PopupConfig {
            return PopupConfig(
    ```

```
            id = "ota_normal_update",
            type = PopupType.DIALOG_OTA_NORMAL_UPDATE,
            layoutRes = R.layout.dialog_ota_normal_update,
            behavior = PopupBehavior(
                autoHideDuration = 0L, // 不自动隐藏
                isModal = true,
                hasOverlay = true,
                overlayClickToHide = false
            ),
            preemptRule = PreemptRule.FORCE, // 强制抢占其他弹窗
            data = mapOf(
                "version" to version,
                "buildNumber" to buildNumber,
                "updateContent" to updateContent,
                "onConfirm" to onConfirm,
                "onCancel" to onCancel
            ),
            bindings = PopupBindings(
                onBind = { view, data -> bindOtaNormalUpdateView(view,
data) },
                onShow = { LogManager.ota("OTA普通更新弹窗显示") }
            )
        )
    }

    /**
     * 创建OTA强制更新弹窗
     */
    fun createOtaForceUpdatePopup(/* 参数 */): PopupConfig { /* 类似实现
*/ }

    /**
     * 创建OTA更新中弹窗
     */
    fun createOtaUpdatingPopup(/* 参数 */): PopupConfig { /* 类似实现 */
}
}
```

- OTA业务层调用

4.3 代码路径

05.兼容性设计

5.1 设备适配

- **屏幕尺寸**：小屏设备折叠布局方案
- **系统版本**：

5.2 冲突检查

现有功能	冲突风险	解决方案
功能A	接口参数变更	版本隔离

06.测试方案

6.1 核心用例

6.2 性能指标

07.发布计划

7.1 阶段发布

7.2 回滚方案

08.文档记录

8.1 技术文档

- 架构设计文档
- 接口API文档

8.2 用户文档

- 功能引导页设计
- 错误代码对照表

8.3 监控埋点