# Android Evergreening 计划

## 一、代码修改部分

### （一）Gradle 版本升级（7.1.2 → 8.5或更高）

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 检查 Gradle 配置文件 | 查看项目级build.gradle和settings.gradle，确保无废弃 API，如将旧版插件应用方式（classpath）改为plugins块 | 0.5 天 |
| 更新 Gradle Wrapper | 修改gradle/wrapper/gradle-wrapper.properties中的distributionUrl为https://services.gradle.org/distributions/gradle-8.5-bin.zip | 0.5 天 |
| 调整构建脚本 | 根据 Gradle 8.5 的语法要求，更新构建脚本中的任务定义、依赖管理等内容，确保与新版本兼容 | 2 天 |
| 处理依赖变化 | 检查项目依赖，对于 Gradle 8.5 中不再支持的依赖方式或旧版库，进行相应的调整和替换 | 1.5 天 |

### （二）AndroidX 版本升级（2020 年旧版 → 2025 年最新稳定版）

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 依赖分析 | 使用 Android Studio 的依赖分析工具，识别所有旧版 AndroidX 库，如androidx.appcompat:appcompat:1.1.0、androidx.core:core:1.3.0等 | 1 天 |
| 批量更新依赖 | 在模块级build.gradle中，将 AndroidX 依赖版本统一替换为 2025 年 4 月的最新稳定版 | 1.5 天 |
| 处理依赖冲突 | 对于更新后出现的依赖冲突，通过implementation、api等依赖范围的调整，或使用resolutionStrategy进行版本强制统一 | 2 天 |
| 迁移代码 | 根据 AndroidX 新版本的 API 变化，对项目中使用旧版 AndroidX 类和方法的代码进行迁移，确保代码兼容性 | 3 天 |

### （三）Gradle 脚本 KTS 化（Groovy → KTS）

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 转换构建脚本 | 将项目级和模块级的build.gradle转换为 KTS 格式，重命名为build.gradle.kts，同时将settings.gradle转换为settings.gradle.kts | 1 天 |
| 处理语法差异 | 按照 Kotlin DSL 的语法规则，调整构建脚本中的配置，如属性访问使用project、rootProject等对象，函数调用使用正确的参数列表和括号 | 2 天 |
| 引入扩展函数 | 利用 KTS 的扩展函数特性，简化构建脚本的编写，提高代码的可读性和可维护性 | 1.5 天 |
| 验证脚本正确性 | 对转换后的 KTS 脚本进行语法检查和构建测试，确保脚本能够正确执行，无编译错误 | 1 天 |

### （四）Kotlin 版本升级（1.4 → 2.1.0）

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 更新 Kotlin 插件 | 在项目级build.gradle.kts中，将 Kotlin 插件版本从1.4更新为2.1.0 | 0.5 天 |
| 检查代码兼容性 | 使用 Kotlin 编译器的检查工具，扫描项目代码，识别在 Kotlin 2.1.0 中可能出现的兼容性问题，如废弃的 API、语法变化等，并进行相应的修改 | 3 天 |
| 更新 Kotlin 标准库 | 确保项目中使用的 Kotlin 标准库版本与目标版本一致，处理因标准库变化带来的代码调整 | 1.5 天 |
| 重构代码 | 根据 Kotlin 2.1.0 的新特性和最佳实践，对项目中的 Kotlin 代码进行适当的重构，提高代码质量和性能 | 2 天 |

### （五）加入 Build Logic 及常规插件

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 创建 Build Logic 模块 | 在项目根目录下创建build-logic模块，用于存放自定义的 Gradle 插件和构建逻辑 | 0.5 天 |
| 定义常规插件 | 在build-logic模块中，定义常用的 Gradle 插件，如应用插件、依赖管理插件、代码检查插件等 | 3 天 |
| 应用插件 | 在项目级和模块级的build.gradle.kts中，应用定义好的 Build Logic 插件和常规插件，简化构建脚本的配置 | 1.5 天 |
| 配置插件参数 | 根据项目需求，对应用的插件进行参数配置，如设置编译选项、代码检查规则、依赖版本等 | 1.5 天 |

## 二、功能自测部分

### （一）Gradle 版本升级自测

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 构建项目 | 使用升级后的 Gradle 版本构建项目，检查是否能够成功编译，无构建错误 | 0.5 天 |
| 查看构建日志 | 分析构建日志，确认没有与 Gradle 版本相关的警告或错误信息，确保构建过程正常 | 0.5 天 |
| 验证增量构建 | 进行增量构建测试，检查增量构建的速度和准确性，确保与预期一致 | 1 天 |

### （二）AndroidX 版本升级自测

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| UI 界面测试 | 在不同的设备和屏幕尺寸上运行应用，检查 UI 界面是否正常显示，无布局错乱、控件丢失等问题 | 2 天 |
| 功能验证 | 对应用的主要功能进行测试，如 Activity 和 Fragment 的跳转、按钮点击事件、列表显示等，确保功能正常实现，无因 AndroidX 版本升级导致的功能异常 | 3 天 |
| 兼容性测试 | 在不同的 Android 版本上运行应用，检查应用的兼容性，确保在目标支持的 Android 版本上能够稳定运行 | 2 天 |

### （三）Gradle 脚本 KTS 化自测

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 构建脚本语法检查 | 使用 Android Studio 的语法检查功能，确保 KTS 脚本无语法错误，代码格式正确 | 0.5 天 |
| 构建功能验证 | 通过 KTS 脚本构建项目，检查构建结果是否与原 Groovy 脚本构建结果一致，确保构建配置正确生效 | 1 天 |
| 脚本可读性检查 | 检查 KTS 脚本的可读性和可维护性，确保团队成员能够轻松理解和修改脚本内容 | 0.5 天 |

### （四）Kotlin 版本升级自测

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 代码编译测试 | 编译项目代码，确保没有因 Kotlin 版本升级导致的编译错误，所有代码能够正常编译通过 | 1 天 |
| 运行时测试 | 运行应用，进行功能测试和稳定性测试，检查是否存在因 Kotlin 版本变化引起的运行时异常，如空指针异常、类型转换错误等 | 2 天 |
| 新特性验证 | 针对 Kotlin 2.1.0 的新特性，如新增的语言功能、标准库方法等，进行相应的测试，确保项目能够正确使用这些新特性 | 1 天 |

### （五）Build Logic 及常规插件自测

|  |  |  |
| --- | --- | --- |
| 任务名称 | 具体内容 | 预计耗时 |
| 插件应用测试 | 检查项目是否正确应用了 Build Logic 和常规插件，如在构建脚本中是否能够正确引用插件，插件的功能是否正常生效 | 1 天 |
| 插件配置验证 | 验证插件的配置参数是否正确设置，如依赖版本是否统一管理、代码检查规则是否正确应用等，确保插件能够按照预期对项目进行配置和管理 | 1.5 天 |
| 构建行为检查 | 通过应用插件后的构建过程，检查构建行为是否符合预期，如是否能够正确执行自定义的构建任务、是否能够生成所需的构建产物等 | 1 天 |