开源课程期中作业

葛进 *51215903084*

周文格 *51215903071*

开源课程期中作业
1.任务分工
2.项目的基本背景和发展历程介绍
技术类型
版本发布历史
主要贡献者的构成
CI/CD 的使用
其他有价值的信息
为什么Anko停止更新了?
3.项目的历史轨迹分析
4.洞察项目被归档的可能原因
我们对开源的理解

1.任务分工

周文格：项目的基本背景和发展历程介绍

葛进：项目的历史轨迹分析

共同完成：洞察项目被归档的可能原因

2.项目的基本背景和发展历程介绍

技术类型

Anko是Kotlin官方开发的一个让开发Android应用更快速更简单的Kotlin库, 并且能让我们书写的代码更简单清楚更容易阅读。它包括多个部分：

- Anko Commons：
  内含intents、dialogs、logging 和Resources and dimensions等更简洁快速的语法函式库
- Anko Layouts：
  编写动态Android UI 布局
- Anko SQLite：
  针对Android SQLite 的查询DSL 和解析器集合;

- Anko Coroutines:
  基于kotlinx.corutines 函式库的实用程序

版本发布历史

- Apr 03, 2015 Anko 0.5
- Apr 30, 2015 Anko 0.6
- Sep 18, 2015 Anko 0.7
- Dec 02, 2015 Anko 0.8
- Jun 10, 2016 Anko 0.9
- May 18, 2017 Anko 0.10
- Nov 09, 2018 Anko 0.10.8 (last release)

主要贡献者的构成

- yanex 782 commit 来自Tokyo, Japan
- 4u7 110 commits 来自Saint Petersburg

CI/CD 的使用

CI/CD 是一种通过在应用开发阶段引入自动化来频繁向客户交付应用的方法。CI/CD 的核心概念是持续集成、持续交付和持续部署。作为一个面向开发和运营团队的解决方案，CI/CD 主要针对在集成新代码时所引发的问题（亦称："集成地狱"）。 具体而言，CI/CD 可让持续自动化和持续监控贯穿于应用的整个生命周期（从集成和测试阶段，到交付和部署）。这些关联的事务通常被统称为"CI/CD 管道"，由开发和运维团队以敏捷方式协同支持。

CI/CD 中的"CI"始终指持续集成，它属于开发人员的自动化流程。成功的 CI 意味着应用代码的新更改会定期构建、测试并合并到共享存储库中。该解决方案可以解决在一次开发中有太多应用分支，从而导致相互冲突的问题。 CI/CD 中的"CD"指的是持续交付和/或持续部署，这些相关概念有时会交叉使用。两者都事关管道后续阶段的自动化，但它们有时也会单独使用，用于说明自动化程度。

本项目中没有使用到这个方法。

其他有价值的信息

为什么Anko停止更新了?

以下引用自anko/GOODBYE.md:

> While Anko was quite popular among Kotlin users, we have to admit that the experience was not 100% perfect. Until recently, Android View APIs were highly optimized for inflation, and sometimes it wasn't possible to set some of the attributes programmatically. As a result, the DSL had to rely on hacks or workarounds. Also, it was non-trivial to emulate the reflective approach of widget loading needed for supporting AppCompat. We didn't have enough resources to fix all corner cases in a timely manner.
>
> However, things have changed substantially during the last few years. Google officially supported Kotlin, and later even made Kotlin the preferred language for Android application development. Thanks to JetPack, an extensive set of libraries, the rough edges of the SDK were smoothed over.
>
> Anko is a successful project, and it has played its role in establishing a better Android developer experience with Kotlin. However, there are modern alternatives today, and we feel it's time to say goodbye to Anko.

3.项目的历史轨迹分析

具体见`analysis.ipynb`文件。

4.洞察项目被归档的可能原因

因为Kotli语言本身的飞速发展，Anko作为一种临时的解决方案已经渐渐和发展主流脱节。Kotlin官方给出的Anko的替代解决方案：

Layout DSL

- Jetpack Compose. A reactive View DSL for Kotlin, backed by Google.
- Splitties – Views DSL. An extensible View DSL which resembles Anko.

Generic utilities

- Android KTX. A set of Kotlin extensions for different purposes, backed by Google.
- Splitties. A lot of micro-libraries for all occasions.

SQLite helpers

- Room. An annotation-based framework for SQLite database access, backed by Google.
- SQLDelight A type-safe API generator for SQL queries.

我们对开源的理解

开源是一种去中心化的创造，让我们可以自由的发挥来满足创造欲，我们理想的开源项目应该是没有leader没有核心的，甚至开源可以不是一个项目，它可以是一团代码或者一个半成品，就像自助餐桌上的菜品一样让人随时搭配享用或者二次烹饪，有的只是我们需要“吃”的这种需求。

而说到开源精神，那也许是一种给于和获取的平衡，有给于才能有获取，有获取才会有给于的动力。无需指责别人只会获取，我们应该懂得开源是一种创造方式，一个没有创造欲和创造力的人加入开源也是无用的。

开源是一种文化，而不仅限于开源这一特定的行为。我认为推动整个圈子发展的正是开源，开源是黑客们向技术垄断发起的挑战，是程序员们的饕餮狂欢。

美国的 Steven Weber 写的《开源的成功之路》其中说到一个非常重要的世界观的区别：关于人类的动机，具体到编写软件上，究竟是为了挣钱？还是像真正的艺术家一样就是为了创作和尝试？在比尔盖茨看来，盗版的行为，偷窃软件，让程序员免费干活，最终会抑制创新。而在开源黑客看来，发布软件却不发布代码，限制了合作的范围，也阻断了别人可能的改进和进一步创新。看起来，两边都说的很有道理，而且有趣的是，都在拿创新说事儿。究竟什么样的激励，才能激发更多更好的创新呢？是金钱？还是纯粹的爱好、乐趣和荣誉感呢？ 公平一点说，如果没有软件版权、专利法、代码编译与加密技术，软件产业可能远远没有现在那么庞大，也难以养活像现在那么多的程序员。也许只会剩下一部分真正热爱编程，有没有钱都要编点什么的人了。但是，我更想从另一个角度来提问：“这个世界上，最重要、最伟大、最具有影响力的创新，有多少是金钱激励出来的呢？”