СТРЮКОВ П. В., ГЕРБЕРТ Д. В. СОВРЕМЕННЫЕ МЕТОДИКИ РАЗРАБОТКИ ANDROID ПРИЛОЖЕНИЙ НА ЯЗЫКЕ PYTHON

УДК 004.4'2, ГРНТИ 50.05.13

Современные методики разработки Android приложений на языке Python

П. В. Стрюков, Д. В. Герберт

Ухтинский государственный технический университет, г. Ухта

В данной статье рассматриваются современные методики разработки Android приложений на языке Python, акцентируя внимание на различных подходах к компиляции Python кода в мобильные приложения. В работе обсуждаются преимущества и недостатки каждого подхода, а также технические аспекты компиляции, такие как требования к системе и размеры выходных файлов. Так же в статье проанализированы перспективы использования Python в мобильной разработке с учетом предстоящего добавления официальной поддержки Android и iOS.

Ключевые слова: *Python, Android, разработка, flutter, kivy, кроссплатформенность*

Modern methods of developing android applications in Python

P. V. Stryukov, D. V. Gerbert

Ukhta State Technical University, Ukhta

This article discusses modern methods of developing Android applications in Python, focusing on various approaches to compiling Python code into mobile applications. The paper discusses the advantages and disadvantages of each approach, as well as technical aspects of compilation, such as system requirements and output file sizes. The article also analyzes the prospects for using Python in mobile development, taking into account the upcoming addition of official support for Android and iOS.

Keywords: *Python, Android, Development, Flutter, Kivy, Cross-platform*

Введение

В современном мире мобильные приложения стали неотъемлемой частью повседневной жизни миллионов пользователей по всему миру. Android, как одна из ведущих мобильных операционных систем, предоставляет обширные возможности для разработки приложений, отвечающих самым разнообразным потребностям пользователей. В последнее время значительно возрос интерес к использованию языка программирования Python для создания приложений под Android. Этот язык известен своей простотой, мощными библиотеками и широким спектром приложений, что делает его привлекательным для разработчиков, стремящихся создавать функциональные и эффективные мобильные решения

Варианты компиляции Python кода в Android приложение

Существует множество вариантов компиляции Python кода в Android приложение. Все варианты можно условно поделить на несколько групп.

К первой группе относятся методики, которые могут скомпилировать только чистый python код. Чистый python код в данном случае – код на языке Python зависящий только от полностью чистых библиотек – таких библиотек, которые включают в себя код только на языке Python, и никакими сериями зависимостей не связаны со сторонними языками программирования. Примерами могут служить текущая Flet сборка [1], и однопроходная сборка python-for-android (p4a), разработанная командой Kivy.

Ко второй группе можно отнести методики, которые теми или иными способами позволяют скомпилировать так называемые собственные пакеты python, содержащие различные языки программирования (например, библиотеки numpy, pyside, pydantic и т.д.). К ним можно отнести такие методики как многопроходная сборка python-for-android (сначала собирается каждый собственный пакет в отдельности, по индивидуальному рецепту, потом все собранные собственные пакеты объединяются с чистыми и собирается окончательное приложение), а также система MobileForge от BeeWare, основанная на системе Crossenv [2].

В отдельную группу стоит вынести системы, основанные на каких-либо движках или связанных с кросс-компиляций python в промежуточный java код. К таким методам можно причислить игровые движки такие как RenPy [3] и GodotEngine, фреймворк Chaquopy и библиотеку Pyjnius.

Особенности компиляции в различных Фреймворках

В зависимости от выбранной методики компиляции при разработке графических приложений на python можно использовать различные GUI фреймворки. Для p4a выбор наиболее велик. Можно использовать библиотеки Kivy и KivyMD [4] или же фреймворк Qt (а именно его версию PySide6). В случае использования одной из Kivy библиотек, итоговый apk файл получается меньше, чем при использовании PySide (для реальной задачи может быть около 30 мб) [5]. Однако есть у этого выбора и некоторые недостатки относительно qt решения. Во-первых, у Kivy нет бесплатного drag-and-drop gui builder’a для размещения интерфейсных компонентов (на данный момент существует лишь нерабочая и не поддерживаемая архивная версия от создателей библиотеки, а также дорогое проприетарное решение от компании LabDeck). Во-вторых, фреймворк Kivy нынешней версии (2.3.0) имеет свою реализацию асинхронности, конфликтующую с библиотекой asyncio. Данная проблема на текущий момент находится в стадии активного решения, однако разработчики не уточняют, к какой версии смогут полностью устранить данный конфликт.

Процесс компиляции для данного набора фреймворков достаточно сложен. В первую очередь, стоит отметить, что он работает только на Linux и только под определенными архитектурами. Для Kivy проще всего использовать Ubuntu, в то время как для PySide, а также для некоторых других особенно сложных библиотек, стоит использовать что-то из семейства Arch-linux (авторы советуют

использовать Manjaro). При этом, в процессе понадобятся такие инструменты как Android SDK, NDK, buildozer, zip, unzip, automake, autoconf, libltdl-dev и некоторые другие. В общей сложности рекомендуется выделить для компиляции минимум 50 Гб дискового пространства 16 Гб оперативной памяти и 8 ядер процессора.

В случае использования инструментов flet-build GUI библиотекой по определению является Flutter. Или, если быть точнее, – Flet, его специальная реализация. Это server-driven UI фреймворк, который позволяет создавать абсолютно одинаковые приложения для iOS, Android, windows, Linux, Macos, а также браузеров. Компиляция чистых пакетов возможна на любой платформе, однако если в приложении находятся собственные пакеты, их необходимо отдельно скомпилировать в p4a и buildozer на Arch-linux. После компиляции итоговые APK становятся по размеру сопоставимы с APK созданными при помощи Kivy. На текущий момент это самый быстроразвивающийся python фреймворк для разработки под android. По этой причине, многие важные составляющие (такие как компиляция под разные платформы, билдер графического интерфейса) находятся в стадии альфа тестирования или даже на стадии закрытой разработки.

Достаточно интересный вариант компиляции предлагает компания BeeWare со своим пакетом разработки, включающим в себя такие инструменты как toga, MobileForge, и briefcase. Данная система предлагает собственный подход к компиляции Python и собственную библиотеку для графического интерфейса. Основная идея заключается в том, что большинство существующих собственных пакетов Python можно будет скомпилировать для iOS и/или Android, просто добавив рецепт meta.yaml только в файл, без каких-либо хаков или исправлений. Сборка на текущий момент возможна только в Linux. В результате собранные приложения примерно на 20% меньше, чем аналогичные, собранные при помощи buildozer, p4a и Kivy (Рисунок 1).

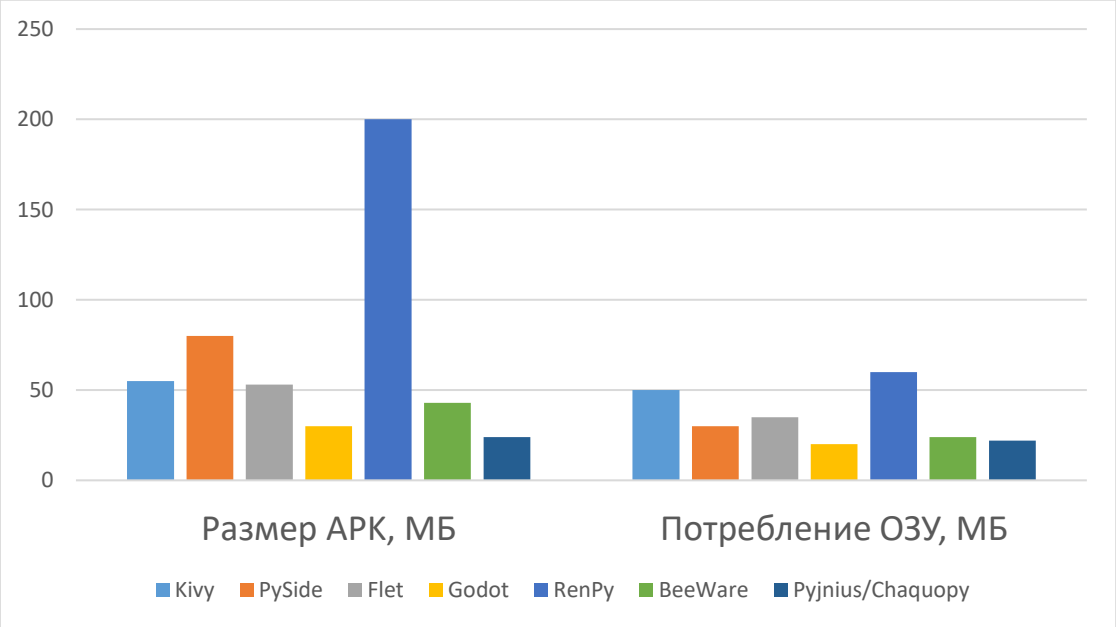

Рисунок 1. Потребление памяти эквивалентными программами в разных сборках

В случае использования Godot Engine, Pyjnius, Chaquopy или RenPy интерфейс проще всего создавать при помощи чистой xml разметки, или одной из её производных (в некоторых случаях это вообще единственный вариант).

При этом при компиляции с использованием Pyjnius (это также обычно верно и для Chaquopy) требуется напрямую работать с Java и Kotlin [6]. Это сильно ограничивает применение Python, и итоговый код проекта чаще всего представляет собой смесь из нескольких языков. Сегодня, в реальной разработке подобная методика применяется довольно часто для создания вставок Python кода со специфическими библиотеками в проекты на Kotlin. В результате итоговые APK файлы получаются значительно меньше, чем при любом другом подходе. Однако в данном случае написать более-менее крупную программу на чистом Python практически не реально.

Компиляция python кода в APK файл на Godot Engine реализована сравнительно недавно [7]. В виду этого существуют определенные проблемы с некоторыми библиотеками. К примеру, с современными версиями библиотеки cryptography, которым для правильной работы нужен компилятор Rust. К плюсам godot-компиляции можно отнести размер исполняемого файла, который в среднем на 25% меньше, чем у аналогичного проекта на Kivy.

Компиляция в RenPy на текущий момент создает самые крупные исполняемые файлы. Их размеры для сложного приложения могут достигать 2 гигабайт. При этом компиляцию можно осуществить даже на windows. Также нет ограничения по используемым библиотекам.

Заключение

Исходя из примеров данного исследования может показаться, что приложения на Python, собранные под Android, занимают колоссальное количество памяти и потому подобный подход к разработке крайне неэффективен. Однако, в реальности основная проблема Python сборки заключается в том, что в конечный файл почти всегда попадают лишние (неиспользуемые) части библиотек. При этом, на текущий момент, эта проблема находится на стадии активного решения. К тому же, при разработке крупных приложений количество неиспользуемых частей снижается и размеры становятся соизмеримыми с приложениями Android, написанными на других языках.

Стоит понимать, что на текущий момент Android и iOS не являются официально поддерживаемыми платформами в Python. При этом сейчас идет процесс (PEP 738 и PEP 730) по добавлению официальной поддержки Android и iOS в Python 3.13 (или по крайней мере в Python 3.14) [8]. Над этим улучшением ядра Python работают, помимо maintainer’ов Python, команды BeeWare, Flet, Kivy, Chaquopy. Все это дает возможность предполагать, что в ближайшем будущем на Android будет появляться все больше программ, написанных на Python.

Список использованных источников и литературы

1. Документация Flet [Электронный ресурс]. – Режим доступа: https://flet.dev/docs/getting-started/ (дата обращения 14.06.2024).

2. Документация BeeWare [Электронный ресурс]. – Режим доступа: https://docs.beeware.org/en/latest/ (дата обращения 14.06.2024).

3. Документация движка RenPy [Электронный ресурс]. – Режим доступа: https://www.renpy.org/doc/html/quickstart.html (дата обращения 14.06.2024).

4. Документация Kivy [Электронный ресурс]. – Режим доступа: https://kivy.org/doc/stable/gettingstarted/index.html (дата обращения 14.06.2024).

5. Документация PySide6 [Электронный ресурс]. – Режим доступа: https://wiki.qt.io/Qt_for_Python#Getting_Started (дата обращения 14.06.2024).

6. Документация Pyjnius [Электронный ресурс]. – Режим доступа: https://pyjnius.readthedocs.io/en/latest/quickstart.html (дата обращения 14.06.2024).

7. Документация Godot [Электронный ресурс]. – Режим доступа: https://docs.godotengine.org/en/stable/getting_started/introduction/index.html (дата обращения 14.06.2024).

8. Предложения по улучшению Python [Электронный ресурс]. – Режим доступа: https://peps.python.org/ (дата обращения 14.06.2024).

List of references

1. Flet documentation [Electronic resource] – https://flet.dev/docs/getting-started/ (Accessed 06/14/2024).

2. BeeWare documentation [Electronic resource] – https://docs.beeware.org/en/latest/ (Accessed 06/14/2024).

3. RenPy engine documentation [Electronic resource] – https://www.renpy.org/doc/html/quickstart.html (Accessed 06/14/2024).

4. Kivy documentation [Electronic resource] – https://kivy.org/doc/stable/gettingstarted/index.html (Accessed 06/14/2024).

5. PySide6 documentation [Electronic resource] – https://wiki.qt.io/Qt_for_Python#Getting_Started (Accessed 06/14/2024).

6. Pyjnius documentation [Electronic resource] – https://pyjnius.readthedocs.io/en/latest/quickstart.html (Accessed 06/14/2024).

7. Godot documentation [Electronic resource] – https://docs.godotengine.org/en/stable/getting_started/introduction/index.html (Accessed 06/14/2024).

8. Python Enhancement Proposals [Electronic resource] – https://peps.python.org/ (Accessed 06/14/2024).