УДК 519.688

СОВРЕМЕННЫЕ ПОДХОДЫ К РАЗРАБОТКЕ КРОССПЛАТФОРМЕННЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

© 2025 А. В. Миронов[1], Е. А. Гришанов[2], А. В. Овсянников[3]

[1]студент 2 года обучения
e-mail: samuary150305@gmail.com
[2]студент 2 года обучения
e-mail: marselleze.88@gmail.com
[3]ассистент кафедры программного обеспечения
и администрирования информационных систем
e-mail: ovsyannikovxxx@yandex.ru

Курский государственный университет

Развитие мобильных технологий и растущая конкуренция в сфере мобильных приложений стимулируют разработчиков к поиску эффективных методов создания программных решений, работающих на различных платформах. В данной статье рассматриваются ключевые подходы к разработке кроссплатформенных приложений, такие как гибридный, интерпретируемый, кросс-компилируемый, прогрессивные веб-приложения (PWA) и модельно-ориентированный подход.

Ключевые слова: разработка кроссплатформенных мобильных приложений, сравнение фреймворков, оценка производительности.

MODERN APPROACHES TO THE DEVELOPMENT OF CROSS-PLATFORM MOBILE APPLICATIONS

© 2025 A. V. Mironov[1], E. A. Grishanov[2], A. V. Ovsyannikov[3]

[1]Student 2 years of study
e-mail: samuary150305@gmail.com
[2]Student 2 years of study
e-mail: marselleze.88@gmail.com
[3]Assistant of the Department of Software
and Information Systems Administration
e-mail: ovsyannikovxxx@yandex.ru

Kursk State University

The development of mobile technologies and the growing competition in the field of mobile applications encourage developers to search for effective methods of creating software solutions that work on various platforms. This article discusses key approaches to cross-platform application development, such as hybrid, interpreted, cross-compiled, progressive Web applications (PWA) and a model-oriented approach.

Keywords: development of cross-platform mobile applications, comparison of frameworks, performance evaluation.

С момента выпуска операционной системы Android 1.0 23 сентября 2008 г. компанией Google ее популярность стремительно росла. Другой операционной

системой, которая делит рынок с Android, является iOS, выпущенная в июне 2007 г. Два крупнейших магазина приложений, Playstore от Google и Appstore от Apple, содержат 3,5 миллиона и 2,1 миллиона приложений соответственно [1].

Поскольку платформы Android и iOS делят между собой большую часть пользователей смартфонов, предприятиям и независимым разработчикам мобильных устройств необходимо, чтобы их приложения были представлены на обеих платформах. Одним из решений этой задачи является кроссплатформенная разработка мобильных приложений.

Разработка кроссплатформенного мобильного приложения

Существует несколько подходов к разработке мобильных приложений. Общий и основной подход называется разработкой нативных приложений. Этот метод нацелен на определенную платформу, используя ее SDK (Software Development Kit). Такие приложения определяют платформу по своему визуальному интерфейсу и внешнему виду. Эти приложения разрабатываются с помощью инструментов и языков программирования, которые специфичны для платформы, для которой они разрабатываются. Например, разработчики Android должны знать такие языки, как Java и Kotlin, причем Kotlin является официальным языком разработки для платформы Android. С другой стороны, для разработки приложений для iOS необходимо твердое знание таких языков, как Swift и/или Objective-C.

И Android, и iOS имеют разные наборы языков программирования и комплекты программ для разработки приложений, это означает, что приложение, разработанное для одной платформы, не может работать на другой. Чтобы сделать приложение доступным как для Android, так и для iOS, потребуется отдельное кодирование, что приведёт к чрезмерной трате ресурсов и усилий.

Для борьбы с этими проблемами появился другой подход, цель которого – разрабатывать приложения для нескольких платформ, имея всего лишь единую кодовую базу. Этот подход называется кроссплатформенной разработкой мобильных приложений. Кроссплатформенные решения должны в конечном итоге достигать производительности, равной той, которую обеспечивают "родные" приложения, или даже лучшей.

Кроссплатформенная разработка – это общий термин, который может подходить ко многим сценариям, помимо разработки мобильных приложений (например, планшеты, телевизоры и т.д.). Данная статья посвящена применению кроссплатформенной разработки в контексте разработки мобильных приложений.

Существует множество подходов, которые обсуждаются в различных исследованиях, для разработки кроссплатформенных приложений на основе единого исходного кода. Некоторые из обсуждаемых подходов: Web (также называемый m-site), гибридный, интерпретируемый, кросс-компилируемый (также называемый генерируемым), модельно-ориентированный, PWA, облачный и App factory. В рамках данной статьи рассмотрим пять распространенных подходов в CPD.

Гибридный подход

Гибридный подход – это сочетание технологий разработки веб- и нативных приложений. Он объединяет веб-технологии, такие как HTML, CSS и JavaScript, с функциональными возможностями нативных приложений. Гибридные приложения используют компонент WebView (называемый WebView в Android и WKWebView в iOS), инициализируя новый проект нативного приложения, который включает компонент WebView и код для взаимодействия WebView и нативного кода. Это означает, что разработчики гибридных приложений могут создавать приложение

с ёиспользованием веб-технологий, а затем обернуть этот код в нативное приложение, которое свяжет и отобразит его в компоненте WebView. Приложение также имеет доступ к нативным функциям через абстрактный JavaScript Bridge. Гибридные приложения также могут быть опубликованы в магазинах приложений.

Популярным фреймворком, попадающим под этот подход, является Ionic Framework от Drifty Co. Ionic имеет открытый исходный код и хорошо документирован. Разработчики могут использовать HTML, CSS и JavaScript для создания приложений в Ionic. В Ionic также есть интеграции для фреймворков Angular, React и Vue. Ionic обеспечивает адаптивную стилизацию в зависимости от используемого устройства, в результате чего приложения выглядят как родные. Ionic поставляется с Ionic CLI, который предоставляет ряд команд для настройки и конфигурирования проектов Ionic. Для нативного развертывания Ionic использует Capacitor или Cordova. Capacitor – это новый подход Ionic, который действует как мост, позволяя веб-приложениям превращаться в нативные приложения. В Capacitor есть библиотека нативных плагинов, которые предоставляют доступ к функциям нативных устройств. Приложения Ionic также могут подключаться к AWS, Azure, Firebase и т.д. В отличие от традиционных веб-приложений, Ionic поддерживает парные истории навигации [2].

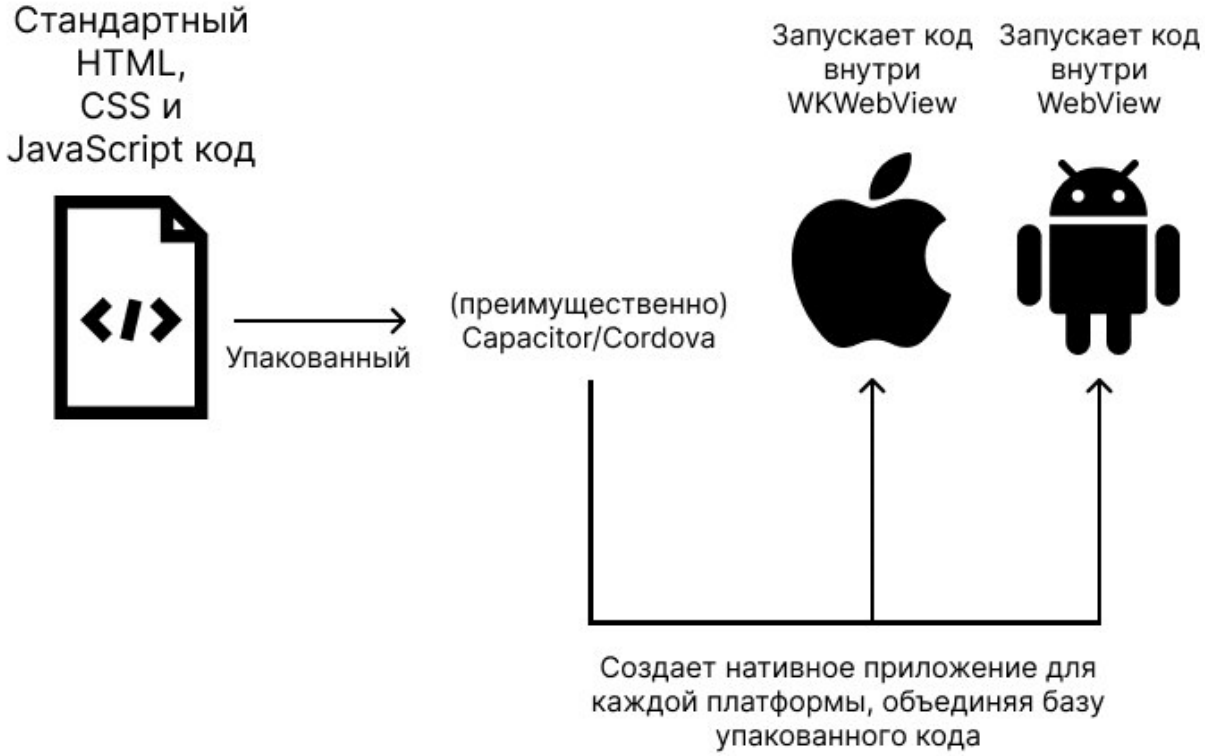

Рис. 1. Рабочий процесс создания гибридного подхода

Интерпретированный подход

Популярным фреймворком для разработки мобильных приложений, подпадающим под этот подход, является фреймворк React Native от Facebook. React Native – это фреймворк с открытым исходным кодом, используемый для создания мобильных приложений с помощью React и нативных возможностей для платформ iOS и Android. В React Native JavaScript используется для доступа к API платформы и для разработки пользовательского интерфейса с помощью компонентов React. Эти

компоненты включают в себя компоненты React Core, компоненты сообщества и собственные компоненты разработчика

Начиная с версии 0.68, React Native представила "Новую архитектуру", которая представляет собой набор обновлений, направленных на повышение производительности и отзывчивости приложений. Основными элементами новой архитектуры являются:

- **турбо-модули** – это обновление нативных модулей React Native, которое теперь предоставляет возможность писать код на C++, ленивую загрузку модулей и использование JSI. Интерфейс JavaScript (JSI) – это дополнение в React Native, технически заменяющее «мост», который использовался для взаимодействия между «потоком JavaScript» и «нативным потоком». JSI позволяет разработчикам использовать любой JavaScript-движок вместо JavaScriptCore. Следует отметить, что начиная с версии 0.70, Hermes является движком по умолчанию в React Native;
- **Fabric** – это новая система рендеринга React Native, обеспечивающая синхронный рендеринг, общее ядро C++, лучшую производительность и меньшую сериализацию данных между JavaScript и хост-платформой;
- **Codegen** – это инструмент, представленный в React Native, который позволяет разработчикам писать меньше повторяющегося кода. Он генерирует код, который можно написать и вручную, но Codegen генерирует код-скаффолдинг. Он также генерирует больше нативного кода во время сборки.

Эта «новая архитектура» значительно изменила React Native по сравнению с предыдущими исследованиями и делает эту диссертацию современным анализом фреймворка React Native [3].

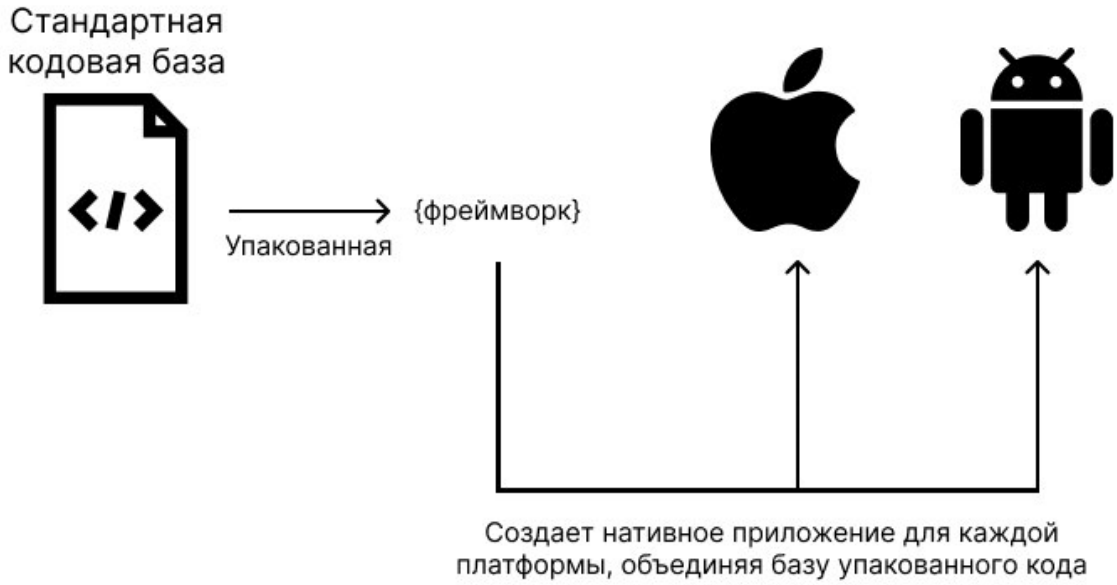

Рис. 2. Интерпретированный рабочий процесс построения подхода

Кросс-компиляция

Кросс-компилятор создает байт-код для платформы, отличной от той, на которой работает компилятор, генерируя код для нескольких платформ. Кросс-компиляция применяет кросс-компиляторы вместо WebView или интерпретации во время выполнения, чтобы отобразить пользовательский интерфейс и получить доступ к функциям родной платформы. Этот подход работает путем компиляции общего языка

программирования, такого как C#, в нативный байт-код, таким образом создавая различные нативные приложения для каждой платформы. В этом подходе доступ к функциям родного устройства осуществляется через комплект средств разработки программного обеспечения (SDK), основанный на компиляции, что означает отсутствие понятия «мост», о котором говорилось в предыдущих подходах.

Популярным фреймворком, работающим на основе концепции кросс-компиляции, является Flutter. В документации по Flutter содержится исчерпывающая информация о том, как он работает, а ниже приводится обзор его архитектуры.

Flutter — это фреймворк Google с открытым исходным кодом, который напрямую компилирует приложения в машинный код. Flutter имеет многоуровневую архитектуру:

- **уровень embedder** – написан на Java и C++ для Android и на Objective-C/Objective-C++ для iOS. Embedder разработан специально для каждой платформы, например, Android и iOS. По сути, он является точкой входа в приложение Flutter, обеспечивая координацию с сервисами хост-платформы, такими как ввод, доступность и рендеринг. Он также отвечает за такие задачи, как управление жизненным циклом и сообщения платформы;
- **уровень движка –** написан преимущественно на C++, C и Java. Он обеспечивает реализацию основных API Flutter, таких, как графика Skia, компоновка текста, файловый и сетевой ввод/вывод, поддержка доступности, архитектура плагинов, среда выполнения Dart и цепочка инструментов компиляции. Движок выполняет растеризацию скомпонованных сцен, когда требуется отрисовка нового кадра;
- **уровень фреймворка** – написан на языке Dart и предоставляет набор библиотек, разделенных на фундаментальные классы, слои рендеринга, слой виджетов и библиотеки Material/Cupertino.

Flutter выделяется среди других кроссплатформенных фреймворков и нативных решений благодаря своей уникальной модели рендеринга. Flutter обходит системные библиотеки виджетов пользовательского интерфейса и включает свой собственный набор виджетов. По сути, Flutter компилирует код Dart в нативный код, используя Skia для рендеринга [4].

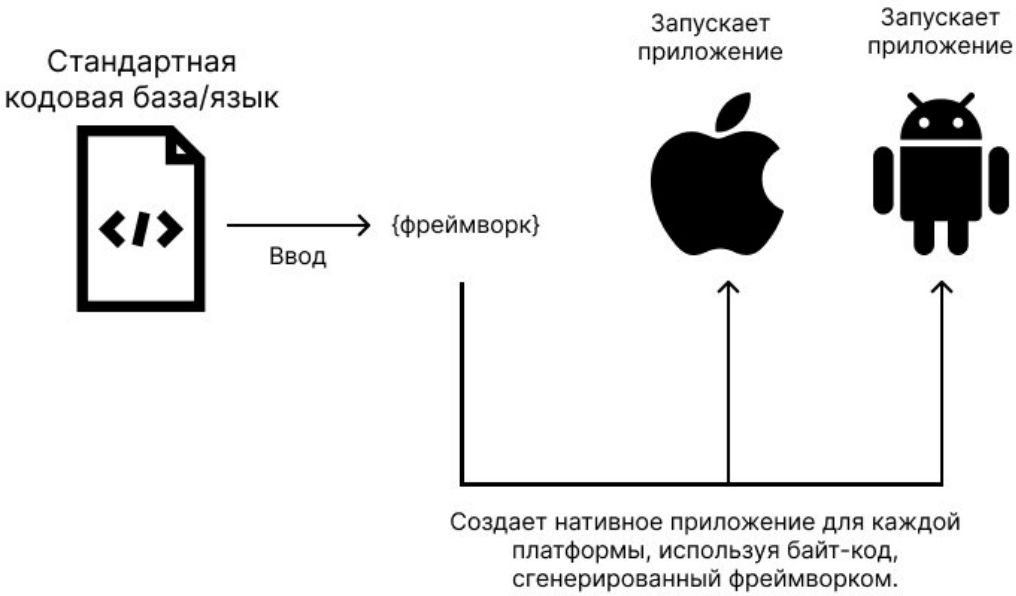

Рис. 3. Рабочий процесс сборки в рамках кросс-компиляции

Подход прогрессивных веб приложений (PWA)

Веб – это мощный инструмент, позволяющий охватить множество аудиторий с помощью одной кодовой базы. Несколько лет назад было невозможно получить доступ к веб-приложениям в автономном режиме или установить их на мобильные телефоны, как другие нативные приложения. Прогрессивные веб-приложения (PWA) – это довольно новый подход к решению этих проблем. PWA – это веб-приложение с усовершенствованиями существующей веб-технологии, предоставляющее пользователям расширенные возможности, такие как:

- иконка запуска приложений;
- появление в поиске приложений;
- автономное окно, отдельное от пользовательского интерфейса браузера;
- более высокий уровень интеграции с операционной системой;
- возможность работы в автономном режиме;
- доступность в магазинах приложений;
- push-уведомления.

PWA можно устанавливать как на iOS, так и на Android. На iOS пользователи Safari (> iOS 11.3) могут запускать PWA и загружать их из AppStore. На Android совместимость с PWA обеспечивают многие популярные браузеры, такие как Firefox, Google Chrome и Opera. PWA также можно устанавливать из Google Play Store [5].

Модельно-ориентированный подход

Модельно-ориентированный подход — это кроссплатформенный подход к разработке мобильных приложений, который использует философию разработки на основе моделей (Model Driven Development, MDD). MDD — это техника генерации кода, в которой высокоуровневые модели уточняются до моделей более низкого уровня и на их основе генерируется исполняемый код. В контексте CPD эти модели могут быть преобразованы в нативный код для Android и iOS. Большинство подходов MDD используют язык, специфичный для конкретного домена (Domain-Specific Language, DSL). Архитектура, управляемая моделями, имеет три типа моделей:

1) модель, независимая от вычислений (CIM);
2) модель, независимая от платформы (PIM);
3) модель, специфичная для платформы (PSM).

Как правило, в MDD CIM определяется экспертами домена. Путем применения преобразований модели создается подробная PIM. Эта модель уточняется и превращается в модель для конкретной платформы (PSM), содержащую детали платформы реализации. Затем эта PSM используется для генерации кода путем применения преобразований модели в текст.

Модельно-ориентированная разработка мобильных приложений все еще остается академической концепцией, учитывая качество фреймворков и документации к ним. Ни один фреймворк или подход не соответствует уровню принятия и четкой документации других фреймворков, рассмотренных в предыдущих разделах. Учитывая это, в литературе обсуждается несколько методов, которые при правильной реализации имеют потенциал масштабируемости и широкого распространения.

App Inventor от Массачусетского технологического института предоставляет набор строительных блоков для создания приложений для Android и iOS. Для создания этих приложений имеется онлайн IDE. Инструмент не поддерживает высокие уровни абстракции и не имеет большого количества практических применений в промышленности со сложными проектами [6].

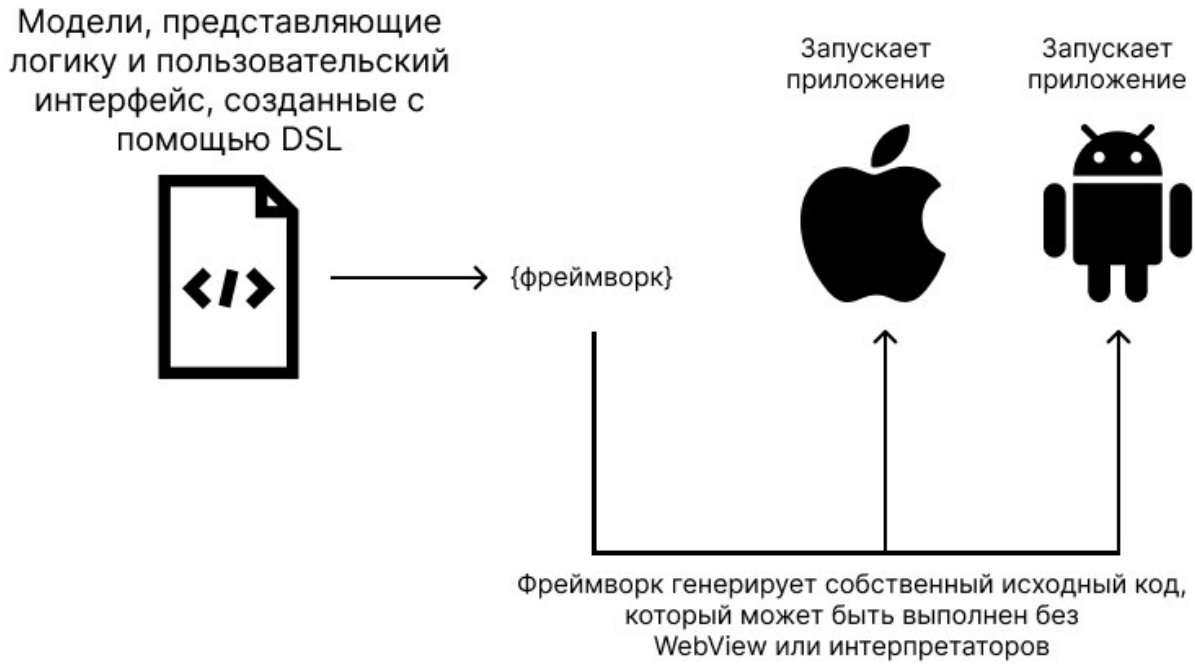

Рис. 4. Рабочий процесс построения MDD-подхода

В статье представлен обзор и анализ современных подходов к кроссплатформенной разработке мобильных приложений. Гибридный и интерпретируемый подходы, реализуемые с помощью таких фреймворков, как Ionic и React Native, обеспечивают быстрый старт разработки, удобство обновлений и интеграцию с веб-технологиями. Кросс-компиляция, реализуемая во Flutter, выделяется высокой производительностью и возможностью компиляции в нативный код, что делает его предпочтительным для ресурсоёмких приложений. Прогрессивные веб-приложения (PWA) предлагают решение для создания лёгких, доступных и автономных приложений, а модельно-ориентированный подход остаётся перспективной, но пока менее распространённой технологией. Результаты анализа демонстрируют, что выбор подхода и фреймворка должен быть обусловлен требованиями проекта, а приведённые рекомендации помогут разработчикам создать кроссплатформенные решения с оптимальными характеристиками.

Библиографический список

1. Statista [Электронный ресурс]. – URL: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ (дата обращения: 15.10.2024)
2. Ionicframework [Электронный ресурс]. – URL: https://ionicframework.com/ (дата обращения: 21.09.2024)
3. Reactnative [Электронный ресурс]. – URL: https://reactnative.dev/docs/getting-started (дата обращения: 25.09.2024)
4. GitHub. Flutter. [Электронный ресурс]. – URL: https://github.com/flutter/flutter/blob/master/docs/README.md (дата обращения: 03.10.2024)
5. GitHub. Reactjs. [Электронный ресурс]. – URL: https://github.com/reactjs/react.dev (дата обращения: 06.10.2024)
6. MIT App Inventor [Электронный ресурс]. – URL: https://appinventor.mit.edu/explore/library (дата обращения: 21.10.2024)

7. Eisenman, B. Learning React Native: Building Modern Mobile Applications with JavaScript. O'Reilly Media, Inc. 2016. [Электронный ресурс]. – URL: https://books.google.ru/books/about/Learning_React_Native.html?id=274fCwAAQBAJ&redir_esc=y (дата обращения: 03.11.2024)

8. Windmill, E. Flutter in Action. Manning Publications. 2020. [Электронный ресурс]. – URL: https://books.google.ru/books?id=EzgzEAAAQBAJ&printsec=frontcover&hl=ru#v=onepage&q&f=false (дата обращения: 15.11.2024)