DOI: 10.24412/2413-2527-2022-230-53-58

Разработка кросс-платформенных приложений на языке Python и фреймворке Kivy

А. И. Федотова, к.воен.н. Р. Г. Гильванов
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
sasha.fedotova.01@mail.ru, gilvanov1950@mail.ru

Аннотация. **В статье рассматриваются варианты разработки приложений, такие как Web-приложения, нативные и кросс-платформенные приложения. Рассматриваются достоинства и недостатки каждого варианта. Сравнивается нативная и кросс-платформенная разработка. Подробнее рассматривается вариант кросс-платформенной разработки приложений, популярные фреймворки. В статье дается обзор языка программирования Python, приведены его достоинства и недостатки. Рассмотрена возможность кросс-платформенной разработки на языке программирования Python с помощью фреймворка Kivy.**

Ключевые слова: **разработка приложений, web-приложения, нативные приложения, кросс-платформенные приложения, фреймворки кросс-платформенной разработки, язык программирования Python, фреймворк Kivy.**

Введение

В современном мире существует большое многообразие техники, с которой взаимодействует человек, и соответственно которой нужен графический интерфейс. Эти устройства могут работать под множеством разнообразных операционных систем: Windows, Linux, MacOS, Android, iOS.

Существует множество различных приложений (как на мобильные устройства, так и на компьютеры, ноутбуки), которые решают большое количество разнообразных задач. Например, это приложения для заметок, планирования, разнообразные игры, приложения для обучения, прослушивания музыки, просмотра видео, приложения для доступа к различным социальным сетям, банкам и т.п. Эти приложения постоянно обновляются, а также разрабатываются новые приложения.

Пользователи этих приложений могут использовать устройства с разными операционными системами. Также, часто пользователю хочется иметь доступ к предоставляемым услугам в приложении с нескольких своих устройств. Например, на ноутбуке (с операционной системой Windows), на планшете (с Android) и на телефоне (с iOS), на всех устройствах желателен доступ к приложению.

Таким образом, перед разработчиком стоит вопрос выбора реализации приложения. Обычно рассматриваются такие варианты как, web-приложение, нативные приложения и кросс-платформенные приложения.

Web-приложение

Web-приложение – это прикладное программное обеспечение, которое работает на web-сервере, в отличие от «стационарных» программ, которые запускаются локально в операционной системе устройства. Для доступа к web-приложению пользователю необходимо активное подключение к Интернету.

Любое web-приложение представляет собой набор статических и динамических web-страниц. Статическая web-страница — это страница, которая всегда отображается перед пользователем в неизменном виде. Web-сервер отправляет страницу по запросу web-браузера без каких-либо изменений. В противоположность этому, сервер вносит изменения в динамическую web-страницу перед отправкой ее браузеру. По причине того, что страница меняется, она называется динамической [1].

Несомненным достоинством web-приложений является их доступность с любого устройства с доступом в Интернет.

Однако это же означает что без доступа в сеть пользователь не может воспользоваться приложением, что является недостатком.

Также отметим, что зачастую на мобильных устройствах гораздо удобнее пользоваться «стационарными» приложениями.

Нативная разработка

Под «нативной разработкой» (англ. *native* — родной, естественный) подразумевается использование оригинальных инструментов, предлагаемых каждой платформой — Swift и Cocoa для iOS и Java или Kotlin с набором инструментов для разработки программного обеспечения (Software Development Kit, SDK) от Android Open Source Project (AOSP) для Android [2].

У нативной разработки есть неоспоримые преимущества, как например [3]:

1. Скорость работы приложения. Код получаемый в результате нативной разработки является оптимальным для своей операционной системы.

2. Полная поддержка со стороны магазинов приложения (App Store и Google Play). Магазины приложений могут просто не пропустить приложение, если качество картинки слишком низкое, скорость выполнения слишком медленная и т.п. Всего этого (качества изображения, скорости выполнения) легче достичь с помощью нативной разработки.

3. Использование последних технологий. Так как средства нативной разработки поддерживаются компаниями-производителями устройств и операционных систем, то весь функционал (аппаратный и программный), даже которые только был добавлен, становится доступен для нативных приложений.

Однако имеется и недостаток — сложность (или высокая стоимость). Для разработки уникального, качественного приложения требуется немало времени и высокая квалификация разработчика.

Кросс-платформенная разработка

Кросс-платформенная разработка — это способ создания приложения с возможностью адаптации под несколько операционных систем. Чаще всего кросс-платформенная разработка рассматривается для мобильных приложений (операционные системы Android и iOS). Однако существует также кросс-платформенная разработка и для десктопных (англ. Desktop – рабочий стол) приложений (операционные системы Windows, Linux и MacOS).

Достоинства кросс-платформенной разработки [3]:

1. Требуется меньше ресурсов для реализации приложения сразу под несколько платформ. Программистов, занимающихся проектом, нужно ровно в два раза меньше. Дизайнер делает только один набор графики. Все это снижает количество рабочих часов и бюджет проекта.

2. Меньшее время на разработку. За счет отсутствия уникальных элементов интерфейса и более простых технологий разработки кросс-платформенных приложений, время на создание простых продуктов, как правило, меньше.

3. Упрощенный цикл обновления продукта. Если в проект нужно что-то добавить или исправить какую-то ошибку, это делается сразу для всех платформ, на которые распространяется проект.

4. Возможность использования мобильной версии сайта. В большинстве случаев языки для кросс-платформенной технологии разработки мобильных приложений входят в семейство языков JavaScript. Поэтому если у проекта уже есть мобильная версия сайта, значительная часть кода и материалов может быть использована в приложении без изменений.

5. Использование единой логики приложения. Логика, заложенная в работу приложения, будет работать гарантированно одинаково для всех платформ. Довольно часто это может являться и минусом из-за разной архитектуры операционных систем. Яркий пример — кнопка «Назад» в навигации между экранами. В Android предусмотрена аппаратная кнопка Back для этих целей, в iOS — движение пальцем от левой части экрана или же наличие кнопки в левой части навигационной панели. Если кнопку не делать вовсе, пользователи iOS не смогут вернуться назад. Если сделать, но не на том месте и выглядящую нестандартно, пользователям iOS будет непривычно и неудобно; а если сделать как в iOS, будет непривычно пользователям Android. Однако написанная и отлаженная один раз логика содержит потенциально меньшее количество ошибок и расхождений в своей работе. Поэтому вам не придется проделывать двойную и тройную работу по поиску проблем на каждой платформе.

Недостатки кросс-платформенных приложений:

1. Устаревание. С каждым обновлением операционной системы появляется возможность (необходимость) улучшить приложение, добавить в него новый, доступный функционал. Для нативной разработки это практически не составляет труда, потому средства нативной разработки почти сразу же предоставляют такую возможность. Однако при кросс-платформенной разработке необходимо дождаться обновления фреймворка, на котором написано кросс-платформенное приложение. А это может занять куда больший срок, по сравнению с нативными приложениями.

2. Низкая скорость. Зачастую кросс-платформенные приложения представляют собой веб-страницы, которые не особо расторопны, например, в скроллинге тяжелого контента: картинок, анимаций и т. д.

3. Трудности дизайна**.** Если вы хотите, чтобы вид вашего приложения соответствовал профессиональному и хорошо-проработанному системному дизайну каждой из платформ, будь то iOS или Android, вам придется разрабатывать дизайн для обеих операционных систем по отдельности. iOS и Android приложения имеют свои собственные, уникальные стандарты дизайна, а так как кросс-платформенное приложение не отвечает им, его вид придется «подгонять» под соответствующие рамки. Получается, по окончании работы вы получите только одно приложение, а времени и денег вы потратили как на два.

4. Незащищенность исходного кода**.** Одним из серьезных минусов кросс-платформенных приложений является их небезопасность. В то время как нативное приложение может быть зашифровано перед выходом в официальный магазин, кросс-платформенное приложение остается «голым». Поскольку в основе многих кросс-платформенных приложений лежит HTML-страница, то ничего не стоит посмотреть ее исходный код и понять, как работает само приложение.

Подводя итоги, можно сказать, что обычно кросс-платформенные фреймворки абстрагируют разработчиков от особенностей написания кода под отдельную платформу. Однако, как только вопрос касается производительности или реализации специфического функционала под платформу, так сэкономленное при разработке время съедается на решении этих проблем.

Также стоит отметить, что подходы при разработке приложения можно комбинировать. Например, отрабатывать критичные к производительности экраны (лента новостей в социальной сети) на нативных технологиях, а второстепенные (экран профиля, экран настроек) — на кросс-платформенных.

Сравнение нативных и кросс-платформенных приложений

Как правило, выход любого бизнеса в Интернет протекает по следующему сценарию: сначала компания запускает сайт, затем его адаптируют под мобильные устройства, и если наблюдается прирост трафика, появляется смысл закрепиться среди владельцев мобильных гаджетов, и компания выпускает приложение.

Сравнивать мобильный сайт и приложение нет смысла: второе однозначно выигрывает за счет широты своих возможностей и отзывчивого интерфейса, взаимодействовать с которым через телефон или планшет гораздо комфортнее. Кроме того, приложение может работать без постоянного подключения к Интернету.

В таблице 1 приведено сравнение нативной и кросс-платформенной разработки по основным пунктам [5]. Из нее следует, что по возможности работать с конкретной операционной системой, качеству интерфейса и производительности нативная разработка абсолютно точно лучше кросс-платформенной. Однако для стартапов, маленьких проектов важнее скорость и стоимость разработки, а также возможность выпустить приложение сразу на несколько платформ, для привлечения большей аудитории. Рассмотрим кросс-платформенную разработку подробнее.

Таблица 1

Сравнение нативных и кросс-платформенных приложений

Критерий оценивания	Нативные приложения	Кросс-платформенные приложения
Стоимость разработки	Высокая стоимость разработки	Относительно низкая стоимость разработки
Возможности использования кода	Работает для одной платформы	Работает на нескольких платформах
Доступность интерфейсов устройств	SDK платформы обеспечивает беспрепятственный доступ ко всем интерфейсам устройства	Нет гарантированного доступа ко всем интерфейсам устройства
Согласованность пользовательского интерфейса	Согласованность с компонентами пользовательского интерфейса устройства	Ограниченная согласованность с компонентами пользовательского интерфейса устройства
Производительность	Безупречная производительность, поскольку приложение разработано для ОС устройства.	Высокая производительность, но нередки задержки и проблемы с совместимостью оборудования.

Популярные фреймворки для кросс-платформенной разработки

Существует множество решений для кросс-платформенной разработки. Наиболее популярные из них [6]:

- PhoneGap (JavaScript, HTML5 и CSS3);
- Ionic (HTML, CSS, JavaScript и Angular);
- React Native (JavaScript, Java, C++, Objective-C и Python);
- Flutter (Dart);
- Xamarin (C#).

Выбор конкретного фреймворка зависит во многом и от языка программирования, на котором хочет работать разработчик.

Одним из наиболее популярных языков программирования на данный момент является язык программирования Python. По рейтингу GitHub [7] последние 5 лет язык программирования Python не опускался ниже 2 места. По рейтингу TIOBE же Python занимает 1 место по состоянию на март 2022 [8]. Python — стабильный и распространенный язык. Он используется во многих проектах и в различных качествах: как основной язык программирования или для создания расширений и интеграции приложений. На Python реализовано большое количество проектов, также он активно используется для создания прототипов будущих программ. В данной статье будет рассмотрена возможность кросс-платформенной разработке на языке программирования Python.

Язык программирования Python

Python это высокоуровневый язык программирования общего назначения с динамической строкой типизацией и автоматический управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нем программ.

Примеры использования Python крупными компаниями [9]:

- **Google.** С момента появления языка компания взяла на вооружение лозунг «Python везде, где можем, а C ++ — где должны». Python не только является компонентом поискового движка, но и считается (наряду с C ++, Java и Go) одним из официальных серверных языков Google, приложения на которых разрешено развертывать в производственной среде.
- **Facebook**. Python занимает третье место (после C++ и Hack) среди самых популярных языков разработки, которыми пользуются инженеры технологического гиганта. На нем сделано более 5 000 коммитов для утилит и инфраструктурных приложений Facebook.
- **Instagram**. Платформа социальных сетей целиком создана на базе Python-фреймворка Django. Она ежедневно дает возможность 4 миллионам активных пользователей фотографировать, редактировать, делиться и сохранять свои творения в личном цифровом альбоме.
- **Spotify**. Крупный игрок на рынке и приложение для потоковой передачи музыки использует Python для аналитики данных. На ее основе работают алгоритмы рекомендаций в популярнейших функциях «Радио» и «Открытия недели».
- **Netflix**. Стриминговый сервис высоко оценил возможности стандартной библиотеки Python, чрезвычайно активное сообщество разработчиков и богатый выбор сторонних библиотек, доступных для решения практически любой конкретной проблемы. В своем блоге компания отмечала, что использует Python на протяжении всего жизненного цикла контента — от принятия решения о финансировании проектов, до управления сетью CDN, предоставляющей видео конечным пользователям.
- **Dropbox**. Популярное онлайн-хранилище применяет Python для оптимизации кода как серверной части, так и внешнего интерфейса. Для этой задачи они привлекали самого создателя «змеиного языка» Гвидо ван Розума. А в 2016 году Dropbox выпустили Pyston — свою собственную реализацию Python, совместимую с CPython и библиотекой NumPy.

Достоинства Python:

1. Удобство и простота.

Python легко читается и понимается. Синтаксис ядра языка минималистичен, за счет чего на практике редко возникает необходимость обращаться к документации.

2. Открытый исходный код.

Лицензия Python с открытым исходным кодом делает его легкодоступным, облегчает распространение и создание модификаций. Разработчики со всего мира могут бесплатно использовать язык и вносить свой вклад в его улучшение. К тому же, в случае с Python сами пользователи, а не крупные компании решают, как будет развиваться технология.

3. Встраиваемость и платформонезависимость.

Благодаря своей интерактивности и переносимости Python обладает хорошими возможностями для динамической семантики и быстрого прототипирования. Его можно легко встроить в широкий спектр приложений, даже в те, которые используют разные языки программирования. Поэтому с Python можно легко исправлять новые модули и расширять базовый словарный запас языка.

Python, как C++, Java и другие высокоуровневые языки программирования, может работать с разными типами компьютеров, ОС и баз данных практически без модификаций. Он хорошо интегрирован не только с популярными платформами Windows, Mac и Linux/UNIX, но и со встроенными системами, такими как Raspberry Pi и Gumstix. Программы на Python также позволяют реализовывать переносимые графические интерфейсы.

4. Большое количество фреймворков.

Одним из главных преимуществ языка Python является наличие у него большого числа фреймворков, упрощающих процесс разработки. Большинство фреймворков Python имеют четкую специализацию, в зависимости от типа и масштаба выполняемых с их помощью задач.

5. Динамическая типизация.

Python не знает тип переменной, пока код не запустится. Он автоматически назначает тип данных во время выполнения. Программисту не нужно заранее беспокоиться об объявлении переменных и их типов данных.

Недостатки Python:

1. Слабая поддержка многопроцессорности.

Многопроцессорность — важная часть написания приложения. Python поддерживает многопроцессорность, но из-за отсутствия прямой поддержки многопоточности (задачи выполняются параллельно в один поток), он может быть не таким гибким или удобным, как другие языки.

Это может создать определенные трудности при параллельном выполнении кода. Хотя подобные ограничения во многом снимается за счет многочисленных дополнительных библиотек Python, умеющих полноценно работать с многопоточностью.

2. Ограничение скорости.

Python часто критикуют за его скорость. Это интерпретируемый скриптовый язык, поэтому он работает относительно медленнее своих скомпилированных аналогов (например, C/C++ или Java), которым не нужно тратить время на перевод текста программы. Тем не менее, некоторые тесты на Python работают быстрее, чем на C и C++.

При этом, Python — не единственный, у кого есть потенциальные проблемы со скоростью. Ruby, Perl и даже JavaScript также находятся на более медленном конце «скоростной» шкалы.

3. Большая нагрузка на память.

Python — это язык, известный гибкостью подходов к типизации данных. Эта же динамическая типизация приводит к повышенному потреблению памяти. Поэтому Python будет неидеальным выбором для задач, интенсивно использующих память.

Исторически Python не был лучшим инструментом для написания GUI (Graphic User Interface) приложений. При разработке мобильных приложений стандартными языками считались Java (Android) и Swift (iOS).

Однако мобильная разработка на Python постепенно прогрессирует. Результатом этого прогресса являются несколько современных инструментов, один из них — фреймворк Kivy [10].

Фреймворк Kivy

Фреймворк Kivy был создан в 2011 году, это кросс-платформенный фреймворк, работающий на Windows, Mac, Linux и Raspberry Pi. В дополнение к стандартному вводу через клавиатуру и мышь он поддерживает мультитач, таким образом его можно использовать и для мобильной разработки Android и iOS. Kivy также поддерживает ускорение GPU своей графики, что во многом является следствием использования OpenGL ES2. У проекта есть лицензия MIT, поэтому библиотеку можно использовать бесплатно и вкупе с коммерческим программным обеспечением.

В настоящее время возможности фреймворка Kivy значительно расширена за счет библиотеки KivyMD. Аббревиатура MD означает Material Design (стандарт, созданный Google, которого нужно придерживаться при разработке приложений для Android и iOS) [11].

Во время разработки приложения через Kivy создается интуитивно понятный интерфейс — Natural User Interface, или NUI**.** Его главная идея в том, чтобы пользователь мог легко и быстро приспособиться к программному обеспечению без чтения инструкций.

Kivy не задействует нативные элементы управления, или виджеты. Все его виджеты настраиваются. Виджеты — это отображаемые на экране элементы управления, которыми пользователь может оперировать, например, кнопки, выпадающие списки и т. п. Это значит, что приложения Kivy будут выглядеть одинаково на всех платформах. Тем не менее, это также предполагает, что внешний вид приложения будет отличаться от нативных приложений пользователя. Это может стать как преимуществом, так и недостатком.

Для создания приложения Kivy необходимо создать подкласс App и переопределить метод build(). Данный метод можно считать главным для приложения, в него помещаются код интерфейса, или же в нем вызываются другие методы, определяющие интерфейс.

```
from kivy.app import App
from kivy.uix.label import Label

class MainApp(App):
    def build(self):
        // your App code

if __name__ == '__main__':
    app = MainApp()
    app.run()
```

Во фреймворке Kivy предусмотрено большое количество разнообразных виджетов, комбинируя которые можно создать нужный интерфейс приложения.

Для работы с дизайном приложения Kivy предоставляет свой собственный язык дизайна (разметки). В таком случае дизайн приложения должен быть описан в файле с расширением *.kv*. Это позволяет отделить логику приложения от его дизайна, то есть поддерживает принцип разделения ответственности, или модель Model-View-Controller (Модель-Представление-Контроллер).

Язык разметки Kivy очень прост и похож на язык разметки HTML.

```
<Button>:
    text: 'Press me'
    size_hint: (.5, .5)
    pos_hint: {'center_x': .5, 'center_y': .5}
    on_press: app.on_press_button()
```

Первая строка <Button> указывает на то, какой метод должен вызываться в коде Python для данного виджета. Далее устанавливается текст кнопки, его размер (ширина и высота), позиция и обработчик событий. Обработчик событий указывает на метод, который должен находиться в Python-классе приложения.

Таким образом, фреймворк Kivy представляет возможность разработки кросс-платформенного приложения с разделением логики и представления приложения.

По завершению разработки приложения можно собрать python-приложение для необходимых операционных систем из числа из поддерживаемых (Windows, Linux, MacOS, Raspberry Pi, Android и iOS). Для этого можно использовать различные инструменты, например, pyinstaller, buildozer и др.

Заключение

Выбор конкретных технологий для реализации приложения зависит от многих факторов, например, от того, для каких операционных систем и устройств необходимо разработать приложение, от необходимого функционала приложения, от субъективных предпочтений разработчика и от многого другого.

Нативная разработка предпочтительнее во многих ситуациях, но если Вы можете работать с не нативным набором инструментов пользовательского интерфейса, Вам нужно выйти сразу на несколько платформ, есть ограничение во времени и средствах, или же просто не хочется писать приложение на Java или Swift, то кросс-платформенная разработка станет отличным решением.

Существует множество фреймворков для кросс-платформенной разработки, и при выборе конкретного фреймворка также важную роль играют личные предпочтения, в том числе и в языке программирования.

Язык программирования Python и фреймворк Kivy — это отличный выбор для новичков в программировании, или для тех, кто просто предпочитает данный язык. А также для стартапов, для проектов, которые нужно вывести на рынок быстро и собрать аудиторию для определения дальнейшего развития продукта.

Литература

1. Общие сведения о веб-приложениях // Adobe. — Обновлено 21.05.2021. URL: http://helpx.adobe.com/ru/dreamweaver/using/web-applications.html (дата обращения 25.03.2022).

2. Льюис, Ш. Нативная разработка мобильных приложений: Перекрестный справочник для iOS и Android = Native Mobile Development: A Cross-Reference for iOS and Android / Ш. Льюис, М. Данн; пер. с англ. А. Н. Киселева. — Москва: ДМК-Пресс, 2020. — 376 с.

3. Мурзанаев, А. Технология создания мобильных приложений: нативная или кроссплатформенная разработка // Блог AppCraft. — 2021. — 18 марта. URL: http://appcraft.pro/blog/nativnaja_razrabotka (дата обращения 25.03.2022).

4. Нативные vs гибридные приложения // Блог UmbrellaIT. — 2017. — 03 августа. URL: http://umbrellait.com/ru/blog/native-vs-hybrid-app (дата обращения 25.03.2022).

5. Manchanda, A. The Ultimate Guide to Cross Platform App Development Frameworks in 2022 // Net Solutions. — Обновлено 04.09.2021. URL: http://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019 (дата обращения 25.03.2022).

6. Patel, A. Top 5 Cross-Platform App Development Frameworks You Need to Know in 2021 // Medium. — 2021. — 17 August. URL: http://medium.com/nerd-for-tech/top-5-cross-platform-app-development-frameworks-you-need-to-know-in-2021-405404cf12fd (дата обращения 20.03.2022).

7. GitHut 2.0. GitHub Languages Stats // GitHub. URL: http://madnight.github.io/githut/#/issues/2020/1 (дата обращения 20.03.2022).

8. TIOBE Index for March 2022 // TIOBE. URL: http://tiobe.com/tiobe-index (дата обращения 20.03.2022).

9. Разработка на Python — плюсы и минусы // Блог Eternalhost. — 2022. — 11 января. URL: http://eternalhost.net/blog/razrabotka/nedostatki-i-preimushhestva-python (дата обращения 13.02.2022).

10. Bader, D. Мобильная разработка на Python: обзор двух фреймворков // Proglib. Библиотека программиста. — 2018. — 16 февраля. URL: http://proglib.io/p/mobile-python (дата обращения 13.02.2022).

11. Постолит, А. Разработка кроссплатформенных мобильных и настольных приложений на Python: Практическое пособие. — Ridero, 2022. — 676 с.

DOI: 10.24412/2413-2527-2022-230-53-58

Development of Cross-Platform Applications with Python and Kivy Framework

A. I. Fedotova, PhD R. G. Gilvanov
Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
sasha.fedotova.01@mail.ru, gilvanov1950@mail.ru

Abstract. **This article discusses application development options such as Web applications, native applications, and cross-platform applications. The advantages and disadvantages of each option are considered. Native and cross-platform development is compared. The option of cross-platform application development, popular frameworks is considered in more detail. The article provides an overview of the Python programming language, its advantages and disadvantages. The possibility of cross-platform development in the Python programming language using the Kivy framework is considered.**

Keywords: **application development, web applications, native applications, cross-platform applications, cross-platform development frameworks, Python programming language, Kivy framework.**

REFERENCES

1. General Information About Web Applications [Obshchie svedeniya o veb-prilozheniyakh], *Adobe*. Last update May 21, 2021. Available at: http://helpx.adobe.com/en/dreamweaver/using/web-applications.html (accessed 25 Mar 2022).

2. Lewis S., Dunn M. Native Mobile Development: A Cross-Reference for iOS and Android [Nativnaya razrabotka mobilnykh prilozheniy: Perekrestnyy spravochnik dlya iOS i Android]. Moscow, DMK Press, 2020, 376 p.

3. Murzanaev A. Technologies for Creating Mobile Applications: Native or Cross-Platform Development [Tekhnologiya sozdaniya mobilnykh prilozheniy: nativnaya ili krossplatformennaya razrabotka], *AppCraft*. Published online at March 18, 2021. Available at: http://appcraft.pro/blog/nativnaja_razrabotka (accessed 25 Mar 2022).

4. Native vs Hybrid Apps [Nativnye vs gibridnye prilozheniya]. *UmbrellaIT*. Published online at August 03, 2017. Available at: http://umbrellait.com/ru/blog/native-vs-hybrid-app (accessed 25 Mar 2022).

5. Manchanda A. The Ultimate Guide to Cross Platform App Development Frameworks in 2022, *Net Solutions*. Last update September 04, 2021. URL: http://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019 (accessed 25 Mar 2022).

6. Patel A. Top 5 Cross-Platform App Development Frameworks You Need to Know in 2021, *Medium*. Published online at August 17, 2021. Available at: http://medium.com/nerd-for-tech/top-5-cross-platform-app-development-frameworks-you-need-to-know-in-2021-405404cf12fd (accessed 20 Mar 2022).

7. GitHut 2.0. GitHub Languages Stats, *GitHub*. Available at: http://madnight.github.io/githut/#/issues/2020/1 (accessed 20 Mar 2022).

8. TIOBE Index for March 2022, *TIOBE*. Available at: http://tiobe.com/tiobe-index (accessed 20 Mar 2022).

9. Python Development — Pros and Cons [Razrabotka na Python — plyusy i minusy], *Eternalhost*. Published online at January 11, 2022. Available at: http://eternalhost.net/blog/razrabotka/nedostatki-i-preimushhestva-python (accessed 13 Feb 2022).

10. Bader D. Mobile Development with Python: An Overview of Two Frameworks [Mobilnaya razrabotka na Python: obzor dvukh freymvorkov], *Proglib. Programmer's Library [Proglib. Biblioteka programmista]*. Published online at February 16, 2018. Available at: http://proglib.io/p/mobile-python (accessed 13 Feb 2022).

11. Postolit A. Developing Cross-Platform Mobile and Desktop Applications with Python: A Practical Guide [Razrabotka krossplatformennykh mobilnykh i nastolnykh prilozheniy na Python: Prakticheskoe posobie]. Ridero, 2022, 676 p.