УДК 004.45

ОСОБЛИВОСТІ МЕТОДІВ РОЗРОБКИ ДОДАТКІВ ДЛЯ ПЛАТФОРМИ ANDROID

Годун Дмитро

Науковий керівник: канд. технічних наук, доцент Присяжнюк О.В.

Центральноукраїнський державний педагогічний університет імені Володимира Винниченка, м. Кропивницький, Україна

У статті проаналізовано основні критерії ефективності використання мов Java, Kotlin та JS для розробки Android додатків. З цією метою порівнювались процеси створення ідентичних додатків за допомогою використання SDK для кожної платформи. Досліджено критерії ефективності вибору технології для розробки Android додаткі з урахуванням глибини застосування мовами Java, Kotlin та JS. Робота може бути корисною для студентів та викладачів фізико-математичних факультетів та осіб, які цікавляться розробкою додатків для мобільних платформ.

Ключові слова: Android-додаток, java, мобільна платформа.

Features of application methods for Android platform

Godun Dmitry

Scientific supervisor: Candidate of Technical Sciences, Prysjazhnjuk O.V.

The Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytsky, Ukraine

The article analyzes the main criteria for using Java, Kotlin, and JS languages to develop Android applications. To this end, the processes of creating identical applications were compared using the SDK for each platform. The criteria of the efficiency of the choice of technology for the development of Android applications are considered, taking into account the depth of use in Java, Kotlin and JS. The work can be useful for students and faculty of physics and mathematics faculties and those interested in the development of applications for mobile platforms.

Keywords: Android app, java, mobile platform.

Постановка проблеми

Наразі мобільні пристрої (смартфони, планшети) стають найбільш перспективним каналом комунікації та засобом оптимізації бізнес-процесів. Згідно з останніми дослідженнями Strategy Analytics, у третьому кварталі 2016 року обсяг поставок світових смартфонів досягав 375 мільйонів одиниць. Ринкова частка Android останні два роки тримається у межах 80–88% [5]. Іншими словами, приблизно дев'ять з десяти пристроїв використовують саме цю ОС.

Багато власників бізнесу усвідомили, що присутність їхнього продукту на мобільних платформах є обов'язковою складовою результативної

маркетингової стратегії. Щоб успішно стартувати на ринку мобільних додатків, потрібно прийняти кілька важливих рішення. Одним з них буде вибір правильного технології створення додатку. Для аналізу ефективності вибору технології для розробки Android додатків варто досліджувати наступні критерії:

Час розробки. Якщо взяти фахівців одного рівня в кожній з технологій і дати однакове технічне завдання, скільки часу буде потрібно, щоб розв'язати цю проблему за допомогою кожної з технологій.

Наявність фахівців. Наскільки швидко можна знайти розробників, які здатні створити продукт на високому якісному рівні, а також фахівців, які зможуть його в подальшому супроводжувати.

Зручність розробки та налагодження. Наскільки розвинені інструменти розробки і налагодження в рамках даної технології.

Документація і технічна підтримка. Чи існує регулярна технічна підтримка для даної технології. Наскільки часто виходять оновлення, як швидко виправляються критичні помилки.

Швидкість роботи. Наскільки швидким буде інтерфейс програми. Чи будуть помітні затримки в переходах між екранами і станами додатка.

Аналіз останніх досліджень і публікацій

Аналіз можливостей апаратних мобільних платформ та інструментальних засобів для розробки мобільних додатків висвітлено у роботах К. Харченко [4], Т. Вакалюк, Р. Горбатюка та ін. В той же час, регулярне оновлення апаратного забезпечення, зокрема мобільних пристроїв, актуалізує потребу в постійному аналізі мобільних платформ та технологій розробки мобільних додатків з метою створення більш якісних програмних засобів.

У статті проаналізовано найбільш популярні за останні декілька років технології розробки мобільних додатків, а саме розробка мовою Java, Kotlin та розробка мовою JS .

Кожна сучасна мобільна платформа надає інструментарій для розробників (SDK - software development kit), який дозволяє отримати доступ практично до всіх сервісів пристрої. Розробники SDK прагнуть спростити

процес створення додатків шляхом вирішення типових задач, з якими сторонні розробники стикаються у повсякденній практиці. Технології, які лежать в основі кожної з SDK, як правило, відрізняються досить сильно. Наприклад, додатки для Андроїд за допомогою офіційного середовища Android studio розробляються на мові програмування Java та Kotlin. Підтримка Kotlin почалась із випуском третьої версії середовища розробки 25 жовтня 2017 року [2]. За допомогою SDK React native можна створювати не складні за функціоналом додатки мовою JS.

Постановка завдання

Мета статті полягає у висвітленні ефективності розробки та дослідженні можливостей глибини застосування мов Java, Kotlin та JS для створення мобільних додатків.

Виклад основного матеріалу дослідження

Для дослідження ефективності технологій розробки мобільних додатків у середовищі Android studio було створено ідентичні на вигляд додатки на Java та Kotlin. Для створення додатку за допомогою React Native встановлено менеджер пакетів Chocolatey, node.js та python2[3].

Для додатків сформоване технічне завдання: додаток складається з двох екранів, на першому з верху текст під ним поле вводу і по центру кнопка (рис.1). По натисканню на кнопку екран перемикається на інший у заголовку показуємо те що користувач ввів у поле вводу на першому екрані. По центру другого екрану відображається список з 20 елементів.

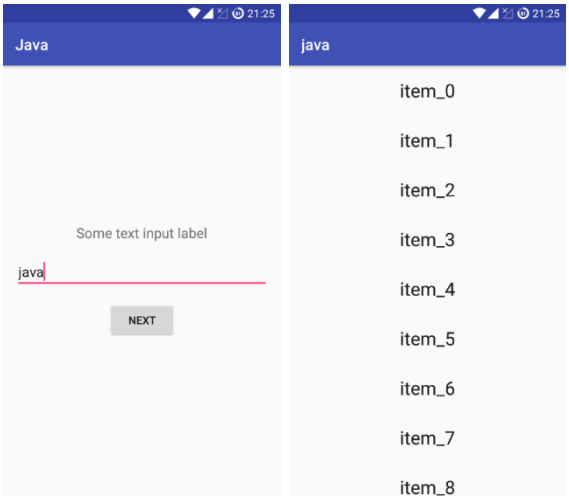

Рис.1. Вигляд додатку

Мовою Java для побудови головного екрану знадобилося 20 рядків коду, для другого екрану знадобилося 17 рядків, а для адаптеру який будує список 36 рядків.

Приклад створення адаптеру для списку мовою Java:

```
public class RvAdapter extends RecyclerView.Adapter<RvAdapter.TextViewHolder> {

    private ArrayList<String> objects;

    public RvAdapter(ArrayList<String> objects) {
        this.objects = objects;
    }

    @Override
    public TextViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        View v = LayoutInflater
                .from(viewGroup.getContext())
                .inflate(R.layout.list_item, viewGroup,  attachToRoot: false);
        return new TextViewHolder(v);
    }

    @Override
    public void onBindViewHolder(TextViewHolder holder, int position) {
        String str = objects.get(position);
        holder.tvText.setText(str);
    }

    @Override
    public int getItemCount() {
        return objects == null ? 0 : objects.size();
    }

    class TextViewHolder extends RecyclerView.ViewHolder {

        TextView tvText;

        public TextViewHolder(View itemView) {
            super(itemView);
            tvText = (TextView) itemView.findViewById(R.id.tvText);
        }
    }
}
```

Мовою Kotlin для побудови головного екрану знадобилося 19 рядків коду, для другого екрану знадобилося 15 рядків, а для адаптеру який будує список 26 рядків. Цікаво виходить – мова Kotlin на 100% сумісна з Java, і Android studio дозволяє java-код в один клік трансформувати в kotlin-код. Потрібно лише натиснути на пункт меню code та обрати Convert java file to kotlin file. На трансформацію коду йде декілька секунд. Отже навіть якщо проект написаний на Java його можна з легкістю трансформувати на Kotlin.

Kotlin економить час програміста на написання повсякденних конструкцій методів і класів. Наприклад для написання моделі потрібна лише один рядок коду:

```
class Message(val text: String, val isRead: Boolean)
```

На Java для створення аналогічної моделі необхідно написати таку громіздку конструкцію:

```
public class Message {
    private String text;
    private boolean isRead;

    public Message(String text, boolean isRead) {
        this.text = text;
        this.isRead = isRead;
    }

    public String getText() {
        return text;
    }

    public boolean isRead() {
        return isRead;
    }
}
```

Приклад створення адаптеру для списку мовою Kotlin:

```
class RvAdapter(private val objects: ArrayList<String>?) : RecyclerView.Adapter<RvAdapter.TextViewHolder>() {

    override fun onCreateViewHolder(viewGroup: ViewGroup, viewType: Int): TextViewHolder {
        val v = LayoutInflater
                .from(viewGroup.context)
                .inflate(R.layout.list_item, viewGroup,  attachToRoot: false)
        return TextViewHolder(v)
    }

    override fun onBindViewHolder(holder: TextViewHolder, position: Int) {
        val str = objects!![position]
        holder.tvText.text = str
    }

    override fun getItemCount(): Int {
        return objects?.size ?: 0
    }

    inner class TextViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

        var tvText: TextView

        init {
            tvText = itemView.findViewById<View>(R.id.tvText) as TextView
        }
    }
}
```

До речі, на Java можна писати код у kotlin-файлах, і все буде працювати - це як ще один бонус стовідсоткової сумісності. За допомогою IDE отриманий

код можна спростити і привести його до kotlin-style. На Kotlin не потрібно явно викликати getter/setter для отримання/присвоєння – звернення відбувається просто по імені.

Встановлювати та оновлювати версії SDK однаково легко для обох мов, так як цим займається Android Studio, користувачу просто потрібно обрати зі списку потрібне та встановити. Так як версії постійно оновлюються з виходом нової версії Android, це дуже зручно для програміста отримувати найновіше SDK перед виходом платформи у масове споживання для користувачів.

React Native пропонує парадигму React для розробки мобільних додатків. Мета полягає не в тому, щоб писати код один раз і запускати його на будь-якій платформі. А навчитися один раз і писати під будь яку платформу. React native дозволяє писати додатки одночасно для Android і iOS. Структура React native проекту кардинально відрізняється від проекту на Java або Kotlin. Створювати додаток не зручно, для розробки додатку необхідно завжди запускати node.js сервер та емулятор або підключати телефон або планшет. Технологія ще досить молода, але все ж таки дозволяє створювати додатки з не складною логікою роботи. Тобто додатки які націлені на тісну роботу з сервером – отримати дані, та відобразити їх. В цьому випадку є деяка перевага над Java та Kotlin. Наприклад код який будує список елементів на екрані:

```
export default class SearchResults extends Component {
    static navigationOptions = {
        title: 'Results',
    };
};

    render() {
        const { params } = this.props.navigation.state;
        return (
            <FlatList
                data={params.listings}
                keyExtractor={this._keyExtractor}
                renderItem={this._renderItem}
            />
        );
    }
}
```

На відміну від Java на JS не потрібно створювати громіздкі допоміжні класи для роботи зі списком. Потрібно тільки передати масив даних, встановити розмітку, стиль для елементів списку і все список готовий. Але є чимало недоліків, наприклад, кастомізація елементу списку, додавання до елементу обробників подій таких як свайпи, не має таких необхідних елементів як Holder – спеціальний кастомізований елемент списку який виконує роль розділювача списку, індикарору завантаження або будь якого іншого елементу який повинен відрізнятися від інших елементів списку але який повинен бути присутнім у списку. До великого мінуса React Native можна віднести те що не можливо перевизначити події життєвого циклу додатку. Тобто не має змоги визначити коли користувач згорнув або розгорнув додаток.

React Native використовує занадто застаріле SDK, велика ймовірність того що додаток буде не правильно, або зовсім не буде працювати на платформі Android з SDK більше 22 (Android 6.0). У додатках створених на React Native цільва аудиторія користувачів значно зменшується порівняно з додатками написаними на Java та Kotlin, оскільки на даний момент цільова версія SDK 27 (Android 8.0).

Наявність документації та навчальних матеріалів по React Native та Kotlin на даний момент досить мало, не завжди можна знайти відповідь поставлені запитання. Розробка додатків мовою Java наразі актуальна як ніколи, написано дуже багато статей, книжок офіційної і не офіційної документації. Завжди можна знайти декілька способів вирішення однієї і тієї ж проблеми.

Висновки

У статті проаналізовано методи розробки мобільних додатків мовами Java, Kotlin та JS. Описано створені Android додатки для кожної мови. Кількість рядків написаного та читабельність коду досліджено на прикладі створення списку. І це показало що для швидкого створення простого додатку краще використовувати технологію React Native з мовою JS. А для додатків які потребують використання складної логіки роботи слід використовувати Java або Kotlin.

Список літератури

1. Strategy Analytics: Android Captures Record 88 Percent Share of Global Smartphone Shipments in Q3 2016 [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/11/02/strategy-analytics-android-captures-record-88-percent-share-of-global-smartphone-shipments-in-q3-2016#.WpbhqejFKUl.
2. Android Studio Release Notes [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://developer.android.com/studio/releases/index.html.
3. Building Projects with Native Code [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://facebook.github.io/react-native/docs/getting-started.html.
4. К.В. Харченко Методи та засоби розробки програмних додатків для операційної системи андроїд.// Збірник наукових праць. Серія: Нові рішення в сучасних технологіях. – Х.: НТУ «ХПІ». – 2014. - № 17. – С. 68-72.
5. Загальна частка ринку мобільних ОС у продажах кінцевим споживачам з 1 кварталу 2009 року до 2-го кварталу 2017 року [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.