почтатех

Что хотят работодатели от Android-разработчика

Поиск идеального стека

Назаров Лев

Спикер

почтатех

Назаров Лев

Руководитель Android-сообщества

lev.nazarov@russianpost.ru

t.me/leffsu

Спикер

почтатех

Назаров Лев

- В 12 лет писал моды для Minecraft
- В 18 лет устроился на первую работу
- Мессенджеры, маркетплейсы, продажа авиабилетов, поддержка инфраструктуры для мобильного тестирования, обработка данных
- Был соискателем на 100+ собеседованиях за свою карьеру
- Провел 50+ собеседований
- Ментор, помог 3 разработчикам прыгнуть на x5 от текущей зарплаты
- Отец двух кошек и комнатный стартапер

Эксперт

почтатех

Копылов Михаил

Главный разработчик

M.Kopylov@russianpost.ru

t.me/Mikhval

Эксперт

почтатех

Копытов Михаил

- Начал погружаться в программирование в 35 лет
- Изучал Android самостоятельно и через курсы
- На работу устроился после первого же собеседования
- Был соискателем на 10+ собеседованиях
- Во время смены работы прошёл три собеседования, получил два оффера

почтатех

Этот доклад для вас, если:

1. Вы новичок и не знаете, с чего начать
2. Вы хотите улучшить свою позицию на рынке труда

Но вы хотите что-то поменять

Два подхода

почтатех

Важны инструменты

- Идет улучшение тулинга как в самих библиотеках, так и в IDE
- Улучшение опыта для разработчиков
- Ускорение работы приложений

Важны подходы

- Меняется синтаксис библиотек, но не меняется суть
- Нам не нужно менять стабильно работающий инструмент ради маржинального ускорения работы (не делать изменения ради изменений)

Почему важен выбор инструмента?

почтатех

Стек технологий:

- Наши проекты полностью реализованы на языке, которым вы не владеете;
- Асинхронность на библиотеках, про которые вы впервые слышите, есть небольшая legacy часть на RxJava2;
- Для presentation уровня используем VIPER архитектуру;
- Для внедрения зависимостей используем чистый чистый DI в C++;
- Навигация на NavProMax 15;
- Пишем Unit тесты с использованием Xamarin.

HR-департамент СуперИнтернетРешения 14:52

Уважаемый Лев,

Благодарим вас за проявленный интерес к нашей компании и за отклик на нашу вакансию Senior Android Developer.

Мы рассмотрели ваше резюме и, к сожалению, должны сообщить, что ваш стек технологий не соответствует требованиям, указанным в описании вакансии. Мы ищем кандидата с опытом работы с RxJava2 и Dagger2, которые являются ключевыми для данной позиции.

Мы ценим ваше время и усилия, которые вы вложили в отклик на нашу вакансию. Если у вас есть другие навыки или опыт, которые могут быть интересны для нашей компании, мы будем рады рассмотреть ваше резюме в будущем.

Спасибо за понимание.

почтатех

Эпохи

Какие разделы мы рассмотрим?

почтатех

База

Язык

UI

Данные

Фоновая работа

Сетевое взаимодействие

Постоянное хранилище данных

Dependency Injection

почтатех

Какие разделы мы не рассмотрим?

UI помощники

- Accompanist
- Epoxy
- Дизайн-системы
- Прочее

MV* помощники

- Orbit MVI
- MVIKotlin
- Moxy
- Прочее

Навигация

- Decompose
- Jetpack Navigation
- Cicerone
- Прочее

Как выглядит 71 вакансия?

почтатех

почтатех

Как выглядит 71 вакансия?

Как выглядят 17 собеседований?

почтатех

Как выглядят 17 собеседований?

почтатех

почтатех

Язык

Язык

почтатех

Java™ vs

почтатех

Язык: Kotlin

1. Двумя словами — золотой стандарт
2. Встречается в 65% вакансий,
   но будет использоваться 100%
3. Google сделал его стандартом
4. Android-подразделение Google
   активно продвигает Kotlin, **игнорируя**
   другие языки
5. Топ-3 по ЗП (по данным JetBrains
   Developer Survey 2023)

Вердикт:
однозначно да!

почтатех

Язык: Java

1. Тремя словами — отсюда выросло всё
2. Встречается в 15% вакансий, но будет в 50%
3. Google постепенно выводит его из исходников
4. Android практически полностью состоит из Java-кода

Вердикт:
желательно да

почтатех

Язык: итоги

1. Зная Kotlin, вы так или иначе знаете и Java

2. Если мы не рассматриваем минусы Kotlin, то нет смысла специально изучать Java

3. В плане подходов: разница минимальная

Вердикт:
Kotlin

почтатех

UI

UI

почтатех

Compose VS View

почтатех

UI: Compose

1. Двумя словами — серебряный стандарт
2. Встречается в 17% вакансий, но большая часть активно хочет его внедрять
3. Google сделал его стандартом
4. Очень высокий порог вхождения
5. Знание Compose является для большей части компании навыком «на будущее»
6. Если вы плохо владеете этим инструментом, то работодатели не видят ничего плохого если это навык «на будущее»

Вердикт:
да

почтатех

UI: Compose

1. Тремя словами — отсюда выросло всё
2. Встречается в 11% вакансий явно, **но будет в 90%+ проектах**
3. Google всё меньше и меньше уделяет времени поддержке
4. Android Framework полностью состоит из View

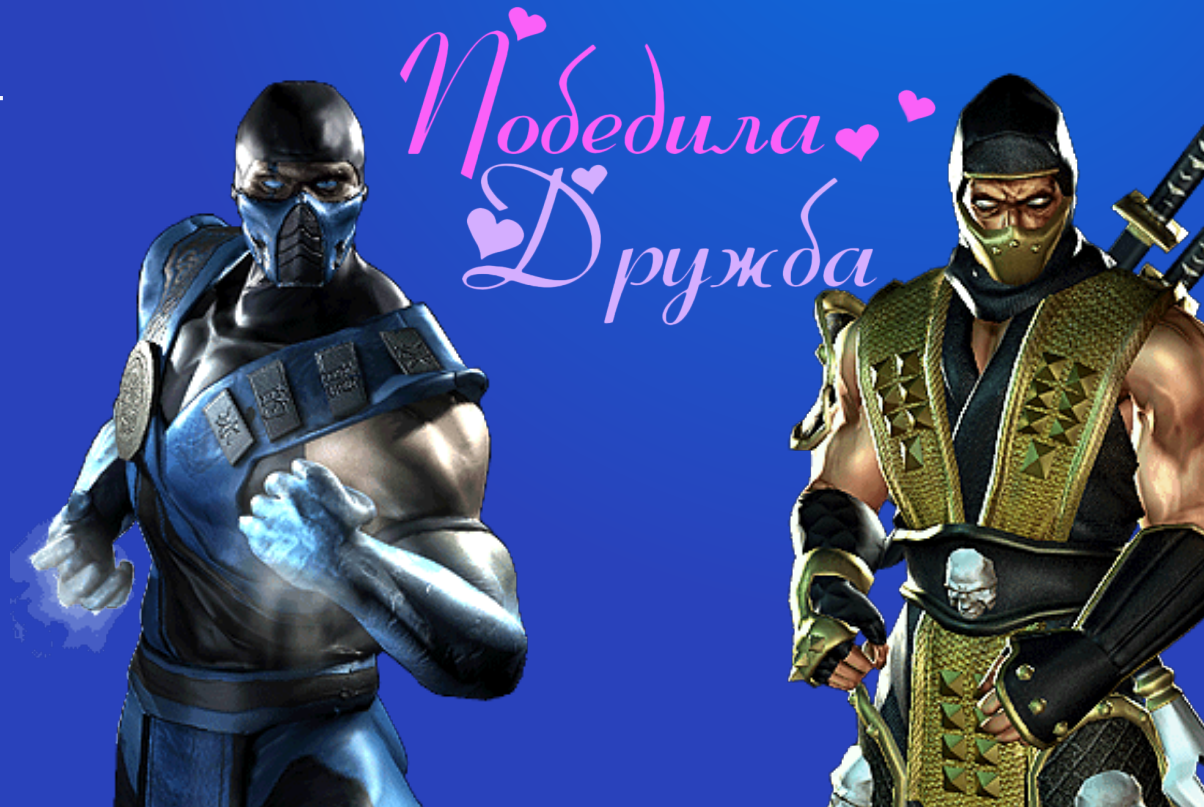
Вердикт:
да

почтатех

UI: Итоги

1. Если бы меня спросили пару лет назад, то я бы сказал View
2. View никуда не делся. Пройдет много лет, пока мы не скажем Compose однозначное «да»
3. В маленьких проектах Compose превалирует
4. В плане подходов: разница фундаментальна
5. Compose — декларативный подход

Вердикт:
View > Compose, но желательно оба



Два подхода

почтатех

Важны инструменты

- Идет улучшение тулинга как в самих библиотеках, так и в IDE
- Улучшение опыта для разработчиков
- Ускорение работы приложений

Важны подходы

- Меняется синтаксис библиотек, но не меняется суть
- Нам не нужно менять стабильно работающий инструмент ради маржинального ускорения работы (не делать изменения ради изменений)

почтатех

Фоновая работа

почтатех

Фоновая работа

RxJava VS Coroutines

почтатех

Фоновая работа: RxJava

1. Четырьмя словами — корутин раньше не было
2. Присутствует активная поддержка сообщества
3. Отсутствие поддержки виртуальных потоков
4. Многие проекты до сих пор используют

Вердикт:
скорее нет, чем да

почтатех

Фоновая работа: Coroutines

1. Двумя словами — **золотой стандарт**
2. Встречается в 40% вакансий, но большая часть активно хочет его внедрять
3. Google сделал его стандартом

Вердикт: однозначно да

почтатех

Фоновая работа: итоги

Не тратьте время на Rx, изучайте Coroutines

Вердикт:
Coroutines

почтатех

Постоянное хранилище данных

почтатех

Постоянное хранилище данных

Realm VS Room

Постоянное хранилище данных: Realm

1. Пятью словами — хороший инструмент, но потерял популярность
2. Присутствует активная проприетарная поддержка
3. Поддержка Kotlin Multiplatform
4. Можно ожидать роста, но сейчас он считается Legacy-инструментом

Вердикт:
неоднозначное
да

почтатех

Постоянное хранилище данных: Room

1. Двумя словами — золотой стандарт
2. Присутствует активная поддержка Google
3. Во времена Realm активно перегнал его из-за реализации киллер-фич

Вердикт:
однозначно да

почтатех

Постоянное хранилище данных: Итоги

1. На проекте с Kotlin Multiplatform от вас попросят Realm
2. На обычном проекте от вас попросят Room
3. Адаптация Kotlin Multiplatform минимальная

Вердикт:
Room

почтатех

Dependency Injection

почтатех

Dependency Injection

Dagger VS Koin

почтатех

Dependency Injection: Dagger

1. Двумя словами — золотой стандарт
2. Присутствует активная поддержка Google
3. Безопасность во время компиляции
4. Встречается в 33% вакансий, **но будет использоваться 65%**

Вердикт:
да

почтатех

Dependency Injection: Koin

1. Шестью словами — это не DI, а сервис-локатор
2. Присутствует активная поддержка сообщества
3. Безопасность во время компиляции появилась только недавно
4. Ускоряет разработку
5. Поддерживает Kotlin Multiplatform
6. Встречается в 8% вакансий, **но будет использоваться 30%**

Вердикт:
да

почтатех

Dependency Injection: Итоги

Вердикт: если прямо сейчас ищем работу,
то Dagger. Если на перспективу, то оба.

Победила Дружба

Что от нас хочет видеть Google сейчас

почтатех

Kotlin

Jetpack Compose

Coroutines

Jetpack Navigation

Room

Dagger2

Espresso

почтатех

Что от нас хотят видеть работодатели

Kotlin	Jetpack Compose	Espresso
Coroutines	Room	Dagger2
Koin	Java	View
RxJava по желанию	Что угодно кроме Jetpack Navigation	Kaspresso

почтатех

Идеальный стек = то, что просит Google?

Kotlin

Jetpack Compose

Coroutines

Jetpack Navigation

Room

Dagger2

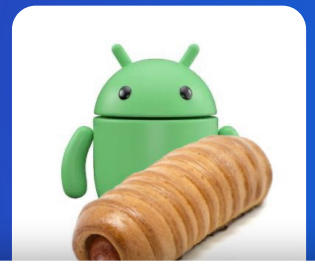
Espresso

почтатех

Спасибо за внимание

почтатех

Почтатех



Android и сосиски в тесте

почтатех

Ниже архив

почтатех

Императивный vs декларативный

Императивный подход

```
List<String> getCatList(House house) {
    if (house == null) {
        return Collections.emptyList();
    }

    List<Animal> animalList = house.animalList();
    if (animalList.isEmpty()) {
        return Collections.emptyList();
    }

    List<Cat> catList = new ArrayList<>();
    for (animal: animalList) {
        if (animal instanceOf Cat) {
            catList(animal);
        }
    }

    return catList;
}
```

Декларативный подход

```
List<String> getCatList(House house) {
    var animalList = Optional.ofNullable(house)
            .map(House::animalList)
            .orElseGet(Collections::emptyList);
    return animalList.stream()
            .filter(Cat.class:isInstance)
            .collect(Collectors.toList());
}
```

почтатех

Императивный vs декларативный

Android View императивный

```
LinearLayout ll = LinearLayout(...)

ll.addView(AddCatView(...).apply {
   textRes(R.string.button_add);
   onClick(new OnClickListener() {
      @Override
      public void onClick(View v) {
         callbacks.onAddCatClicked();
      }
   )
})

if(carousels.size >0) {
   ll.addView(AddCatView(...).apply {
      textRes(R.string.button_add);
      onClick(new OnClickListener() {
         @Override
         public void onClick(View v) {
            callbacks.onClearCatListClicked();
         }
      )
   })
}
```

Airbnb Epoxy декларативный

```
 addButton
       .textRes(R.string.button_add)
       .clickListener((model, parentView, clickedView,
position) → {
        callbacks.onAddCatClicked();
       });

   clearButton
       .textRes(R.string.button_clear)
       .clickListener(v →
callbacks.onClearCatListClicked())
       .addIf(carousels.size() > 0, this);

   shuffleButton
       .textRes(R.string.button_shuffle)
       .clickListener(v →
callbacks.onShuffleCatListClicked())
       .addIf(carousels.size() > 1, this);
```

почтатех

Императивный vs декларативный

Airbnb Epoxy

```
 addButton
       .textRes(R.string.button_add)
       .clickListener((model, parentView, clickedView,
position) -> {
         callbacks.onAddCatClicked();
       });

    clearButton
       .textRes(R.string.button_clear)
       .clickListener(v ->
callbacks.onClearCatListClicked())
       .addIf(carousels.size() > 0, this);
```

Compose

```
CatAddButton(
    textResId = stringResource(id =
R.string.button_add),
    onClick=callbacks:onAddCatClicked
)

if (carousels.size > 0) {
    CatClearButton(
        textResId = stringResource(id =
R.string.button_clear),
        onClick = callbacks:onClearCatListClicked)
}
```