Левченко
Никита Андреевич

Цель:
Android-разработчик

Общая информация о себе:
Возраст: 24
Дата рождения: 7 декабря 1993 года
Семейное положение: не женат
Адрес: г. Санкт-Петербург, Съезжинская 19
Телефон: +79170582199
Email: levko07324@gmail.com

Образование:

2012 – 2016 г.г.	***Ульяновский Государственный Технический Университет*** Специальность: Информационные системы и технологии
2014 г.	***Практика в X-cart. Написание модуля интеграции с платежной системой и модуля карты сайта***
2015 г.	***Участие в серии мастер-классов по направлению «Программирование на Java» в IT.Place на базе ООО «СимбирСофт»***
2015 г.	***Участие в «Летнем интенсиве 2015» на базе IT.Place SimbirSoft***
2016 г.	***Повышение квалификации по 100-часовой программе «Разработчик мобильных приложений» на базе УлГТУ***

Трудовая деятельность:

11.15 - 08.16	***ООО «Июнь-С»*** Должность: Инженер-программист по сопровождению программного обеспечения Обязанности: Android-разработка
09.16 - 12.17	***ООО «АГИМА.мобайл»*** Должность: Разработчик мобильных приложений Обязанности: Android-разработка (в том числе разработка бизнес-логики кросс-платформенных приложений) (13 месяцев), iOS-разработка (2 месяца)

Профессиональный опыт и навыки:

1. Участие в разработке 5 приложений в Google Play и 4 приложений в AppStore
2. Опыт работы с языками: Java (2 года разработки), Kotlin (2 месяца самостоятельного изучения), Objective-C (2 месяца разработки), Swift (2 месяца самостоятельного изучения)
3. Опыт использования архитектурных паттернов:
    a. Понимание подхода Clean Architecture дядюшки Боба
    b. Паттерны Presentation-уровня:
        i. MVP (использование в рабочих проектах библиотеки Moxy)
        ii. MVVM (использование в личных проектах Android Architecture Components)
        iii. MVC «из коробки» в iOS
    c. VIPER (использование в рабочих проектах для построения архитектур на iOS и Android)
    d. Rambler's VIPER (использование в рабочих проектах для построения архитектур на iOS и в личных проектах на Android)
    e. Dependency Injection (использование Dagger 2 на Android и Typhoon на iOS в рабочих и личных проектах)
    f. Понимание паттерна Object-Relational Mapping (использование Realm, OrmLite, Android Room в рабочих и личных проектах)
4. Опыт работы с многопоточным программированием:
    a. Android: RxJava 2, Coroutines
    b. iOS: ReactiveCocoa
    c. Java (для разработки кросс-платформенных приложений): RxJava
5. Опыт работы с СУБД:
    a. Android:
        i. Android Room, Realm
        ii. Работа с SQLite без ORM
    b. iOS: Realm
    c. Java (для разработки кросс-платформенных приложений): OrmLite
6. Опыт использования основных паттернов программирования (Factory, Builder, Singleton, Observable)
7. Опыт работы с библиотеками:
    a. Android: Android Support, Moxy, Android Architecture Components, Android Data Binding Library, RxJava 2, Coroutines, Dagger 2, Retrofit 2, Cicerone, Green Robot's EventBus, Realm, Picasso, Glide, Flurry, Gson, Butter Knife
    b. iOS: Typhoon, Realm, Alamofire
    c. Java(для разработки кросс-платформенных приложений): RxJava, OrmLite
8. Опыт разработки клиент-серверного взаимодействия по REST и SOAP
9. Опыт разработки приложений с протоколом авторизации OAuth

10. Опыт работы по Git flow
11. Опыт работы в командах, использующих Scrum
12. Опыт работы с таск-трекерами:
    a. Продолжительный опыт работы с YouTrack
    b. Небольшой опыт работы с Redmine
    c. Небольшой личный опыт работы с Trello

Интересные задачи:

1. Необходимо было реализовать кэш данных в рамках текущей сессии, доступный с разных экранов, при этом не было известно, какой из экранов инициирует загрузку данных с сервера. В проекте (в тот момент я разрабатывал под iOS, однако это полностью применимо и в разработке под Android) для реализации бизнес-логики использовался ReactiveCocoa (аналог ReactiveX для Objective-C, далее буду оперировать терминами RxJava 2). Очевидно, что для таких целей подходит BehaviorSubject, однако, многие данные необходимо было хранить в базе данных, а не в оперативной памяти устройства. Для достижения данного функционала решил написать обертку над BehaviorSubject, которая принимает и испускает не сами данные, а их *статус*. В качестве параметров конструктора обертка принимала блоки, отвечающие за запрос данных с сервера, сохранение данных в базу, и чтение данных из базы (на Java эквивалентный функционал можно реализовать с помощью интерфейса, содержащего аналогичные методы, на Kotlin – с помощью таких же интерфейсов или делегатов). Дальше дело оставалось за малым – применить оператор map для чтения данных из базы, используя блок из конструктора, или возвращения ошибки (в зависимости от статуса). Таким образом, презентеры (в проекте для Presentation-уровня использовался архитектурный паттерн MVP) подписывались на обертку, как на обычный Observable, оставались независимыми друг от друга, при этом данные грузились только один раз за сессию и хранились в постоянной памяти устройства (или как реализовано в соответствующем блоке/реализации интерфейса/делегате). В случае, если происходила принудительная перезагрузка данных с сервера, все подписанные презентеры автоматически получали обновленные данные.

   P. S. Конкретно в том проекте использовалась архитектура Rambler's VIPER, поэтому, на самом деле, на обертку подписывались интеракторы, но это уже другая история :-)

2. В приложении необходимо было создать локальные напоминания, которые можно было редактировать. При этом напоминания могли быть как ежедневные, так и зависящие от дней недели. Для реализации я использовал AlarmManager, который взаимодействовал с моим

BroadcastReceiver. В методе onReceive() приходилось переназначать время следующего «срабатывания» ресивера в зависимости от актуальных напоминаний. Если же подходящих напоминаний не находилось – переназначал время на полночь. Разумеется, при изменении напоминаний из приложения приходилось снова переназначать время. Таким образом напоминания были реализованы без использования сервисов и практически не нагружали устройство.

P. S. Выяснилось, что на устройствах с прошивкой MIUI BroadcastReceiver срабатывает только при разрешении автозапуска приложения в настройках устройства.

Дополнительная информация:
- Знание английского на уровне чтения технической документации;
- Положительные качества: внимание к деталям, ответственность, стремление к постоянному развитию, желание изучать новые технологии;
- Хобби: игра на электрогитаре, смешанные единоборства;

Пример личного проекта изучения Kotlin и Android Architecture Components (проект находится в процессе разработки):
https://github.com/nikitalevchenko/VK
com/nikitalevchenko/VK