ОСОБЕННОСТИ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ REACT NATIVE

Казанцев С. А., студент гр. ИТм-171, II курс
Протодьяконов А.В., к.т.н., доцент
Кузбасский государственный технический университет
имени Т.Ф. Горбачева

Одной из важнейших задач в сфере тестирования мобильных приложений является автоматизация регрессионного тестирования. Под регрессионным тестированием здесь подразумевается испытание программного обеспечения, направленное на обнаружение ошибок в уже проверенных участках программ. Производится такое тестирование для того, чтобы исправить так называемые регрессионные ошибки — такие ошибки, которые появляются не во время написания программы, а при добавлении новых участков кода или исправлении допущенных ранее промахов в синтаксисе кода. Регрессионные ошибки — это когда после внесения изменений в программу перестаёт работать то, что должно было продолжать работать.[1]

Цель регрессионного тестирования – убедиться в том, что изменения в коде, сделанные с целью добавления новой функциональности или исправления существующих в коде проблем не привело к новым ошибкам в уже проверенных участках программы (в данном случае — мобильного приложения).

Цель автоматизации регрессионного тестирования в том, чтобы повысить эффективность и качество тестирования. Периодически повторяемые однотипные регрессионные проверки отнимают много времени в цикле разработки. Их автоматизация может значительно облегчить работу тестировщиков, проводящих ручное тестирование приложения перед выпуском каждого его обновления.

Особенностью мобильных приложений на платформе React Native является единая кодовая база как в версии приложения для операционной системы (далее ОС) Android, так и для ОС iOS. В обычном случаи приложения для этих ОС были бы написаны разными разработчикам и на разных языках программирования, но при разработке на платформе React Native используется единственный язык — javascript. Таким образом на обоих упомянутых мобильных платформах приложение работает максимально схожим образом. Это делает целесообразным использование единого набора автотестов для обоих приложений.

Выбор устройств для запуска автотестов

В случае с Android можно использовать для тестирования реальные телефоны, планшеты или иные устройства, если они подключены к компьюте-

ру, на котором выполняются тесты, через USB. Также устройства на базе ОС Android не привередливы к тому, какая операционная система установлена на компьютере и отлично работают не только на Windows, но также и на различных дистрибутивах Linux и MacOS. То же касается и среды разработки мобильных приложений для Android. В состав среды разработки Android Studio входит специальное программное обеспечения для создания и запуска эмуляторов с различными версиями ОС. Эмуляторы — наиболее удобный вариант для прогона тестов так как в них без особых усилий можно активировать специальные функции такие как отображение на экране места нажатия и отображение точных координат нажатия.

Для тестирования же iOS-версии мобильного приложения необходимо использовать так называемые симуляторы — специальные программы, в которых запускается компьютерная версия ОС iOS с различными параметрами специально настроенными под эмуляцию конкретной модели iPhone. Симуляторы входят в пакет XCode, предназначенный для разработки приложений для MacOS и iOS. Тот факт, что данный пакет работает исключительно на уже упомянутой операционной системе компании Apple, сужает круг операционных систем, на которых можно было бы развернуть среду тестирования, до одной — **MacOS**.

Подготовка окружения для запуска тестов

```
#!/usr/bin/env bash
PLATFORM=$1
DEVICE=$2

UDID=$(xcrun simctl list devices -j |
 jq -r --arg DEVICE "$DEVICE" --arg PLATFORM "$PLATFORM" '.devices |
 to_entries |
 map(select(.key == $PLATFORM)) |
 .[].value[] |
 select(.name == $DEVICE and .state != "Booted") |
 .udid')

if [[ $UDID < 2 ]]; then
    echo -e "The device doesn't exist or is already open!"
    exit 1
else
    echo -e "Found device with UDID ${UDID}."
fi

xcrun simctl boot ${UDID}
```

Рисунок 1. Скрипт запуска симулятора

В первую очередь перед запуском тестов нужно запустить необходимый симулятор или эмулятор в зависимости от платформы. Предполагается, что эти действия буду выполняться при помощи скрипта или системы непре-

рывной интеграции (далее CI) как например Jenkins или TeamCity, поэтому последовательность действий будет приведена в виде консольных команд.

На рисунке 1 изображён скрипт, запускающий по требованию симулятор определённого iPhone с выбранной версией ОС на борту. Пример его использования:

```
bash launchSim.sh "iOS 11.4" "iPad Air 2"
```

Таким образом после выполнения команды запустится симулятор iPad Air 2 с операционной системой iOS 11.4.

Аналогичный подход возможен для Android, но в отличии от iOS, эмуляторы с нужными параметрами придётся сначала создать вручную.

```
emulator -avd Nexus_5X_API_23
```

где Nexus_5X_API_23 — название нужного вам эмулятора.[2]

Если в процессе тестирования в приложении будет выполнятся авторизация, нужно убедится, что перед каждым запуском тестов статус авторизации будет обновляться, а также все данные — стираться. Это обеспечит большую надёжность тестов так как из-за оставшихся с прошлых запусков данных могу возникнуть сбои. Даже если остающиеся после прохождения или провала тестов объекты не привели к сбою при следующем запуске, сбой может случится через несколько итераций.

В качестве итога после запуска тестов можно получить подробный отчёт об их прохождении, пример на рисунке 2.

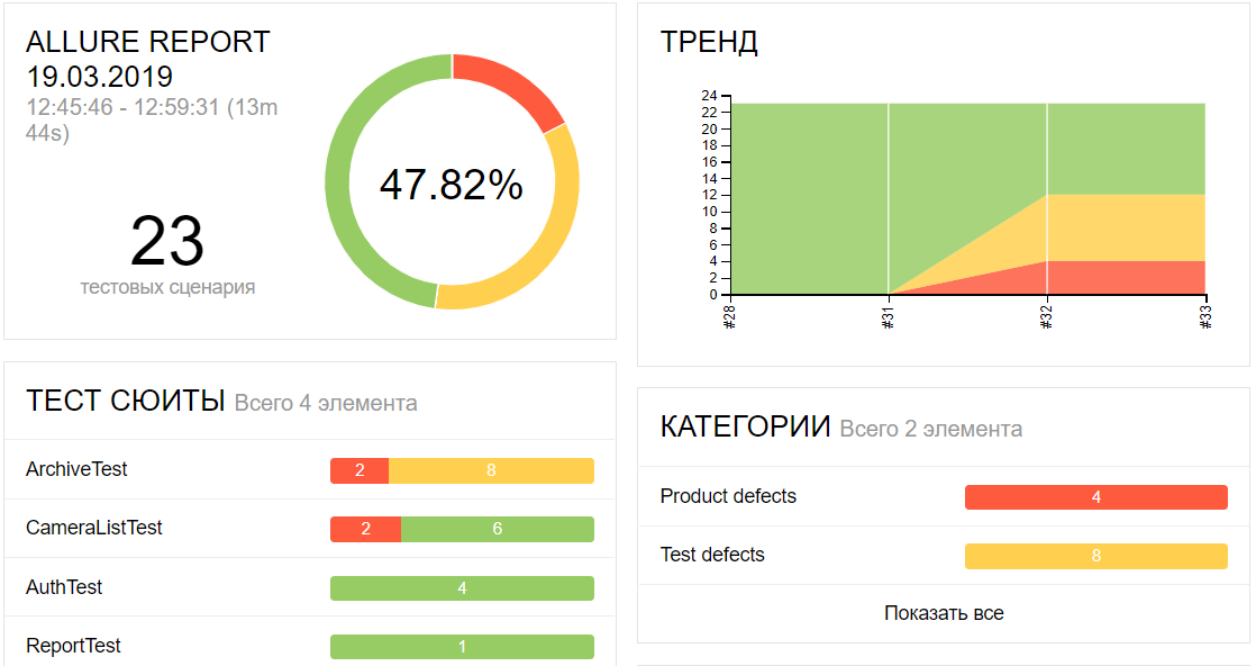

Рисунок 2. Отчёт о результатах прохождения авто-тестов

Таким образом, можно сделать вывод, что несмотря на значительные усилия, необходимые для написания авто-тестов, подготовки окружения для них и обеспечения их корректной работы, это даст следующие результаты:

- качество и эффективность проверки увеличится за счёт того, что проверка будет выполняться по строго заданному алгоритму
- отсутствует необходимость в ручной подготовке отчёта, можно приступать к анализу результатов сразу после окончания тестов

Список литературы

1. Регрессионное тестирование // Национальная библиотека им. Н. Э. Баумана URL: https://ru.bmstu.wiki/Регрессионное_тестирование.
2. Start the emulator from the command line // Android Developers URL: https://developer.android.com/studio/run/emulator-commandline.