Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 19.Б10-мм

Вильданов Эмир Флоридович

Автоматическое создание скриншотов экрана телефона на операционной системе Android

Отчёт по учебной практике

Научный руководитель:
доцент каф. СП, к.т.н., Литвинов Ю. В.

Санкт-Петербург
2021

Оглавление

1. Введение

В этом семестре была поставлена задача создания десктопного приложения, позволяющего в автоматическом режиме создавать скиншоты экрана телефона на операционной системе Android.

Соблюдение безопасности всегда было важной задачей. В разное время специалистам в этой сфере приходилось сталкиваться с разнообразными трудностями. В современности многие задачи обеспечения безопасности стали опираться на решение проблем, связанных с развитием цифровых технологий.

Так сейчас очень большую роль в обеспечении безопасности играют эксперты-криминалисты, работающие с личными цифровыми устройствами: телефонами, компьютерами, планшетами. При получении доступа к устройству перед ними встаёт ряд задач по извлечению из него полезной информации и улик для досудебного следствия. Одной из таких задач является создание скриншотов сетевых чатов: такие снимки могут стать дополнительным доказательством для суда, поскольку содержат зафиксированные участки переписки людей.

Проблема, возникающая при решении данной задачи – это рутинность действий криминалистов. Для получения скришнотов необходимо совершать ряд однообразных действий: прокрутка экрана, нажатие на кнопки, создание и сохранение скриншотов.

Программисты компании Belkasoft, занимающейся разработкой специализированного программного обеспечения для цифровой криминалистики, предложили проект, который бы облегчил эту рутинную работу. Было принято решение создать приложение, которое будет использовать инструменты тестирования пользовательских интерфейсов для того, чтобы автоматически делать скриншоты всех чатов.

2. Постановка задачи

Целью работы является создание приложения, позволяющего в автоматическом режиме создавать снимки экрана телефона. Для выполнения целей разработки были поставлены следующие задачи:

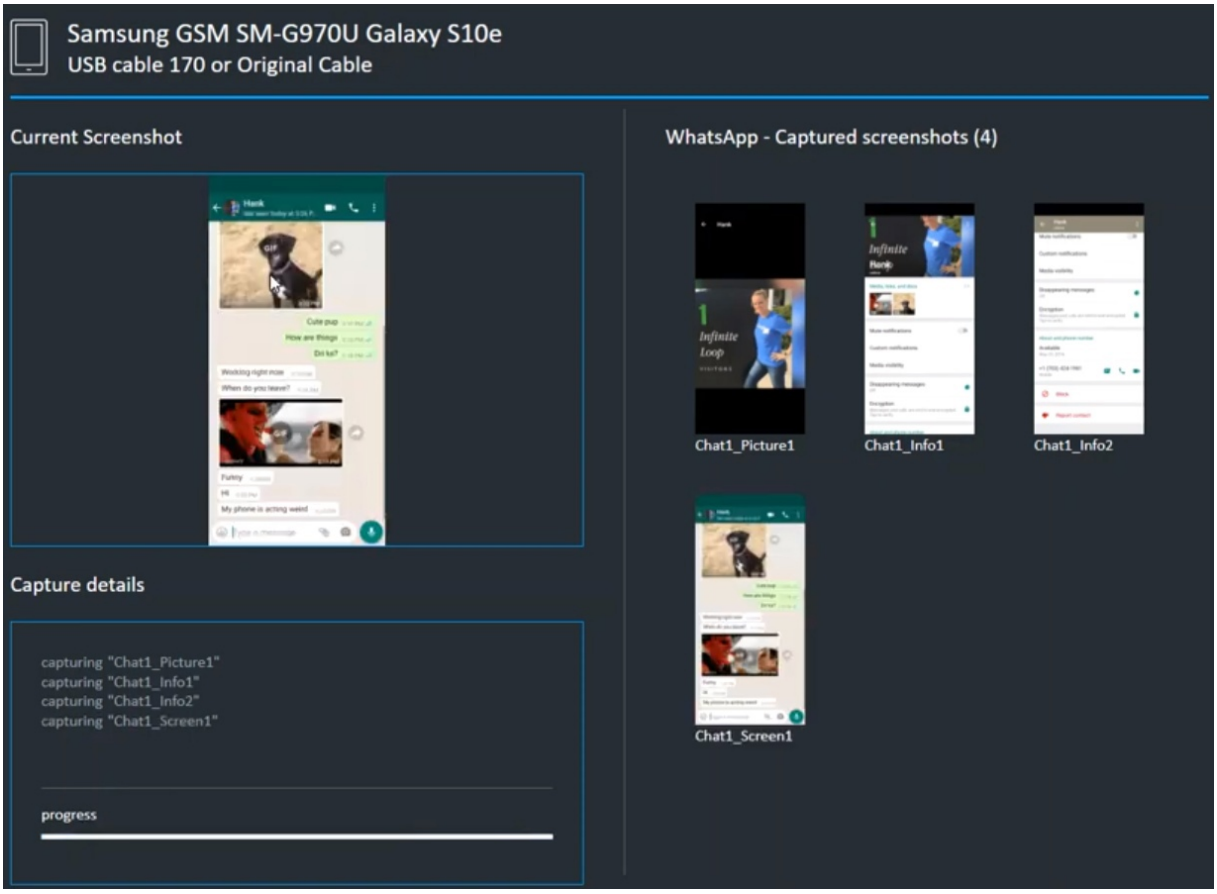
1. Провести обзор предметной области, изучить приложения-аналоги, изучить используемые инструменты;
2. Разработать архитектуру приложения:
   (a) поддержать возможность создания скришотов в ручном режиме;
   (b) поддержать возможность создания скришотов в автоматическом режиме;
   (c) сделать возможным переключение между приложениями, контент которых нужно запечатлеть на скриншотах;
3. Протестировать работу приложения на мобильных устройствах с разными версиями операционной системы Android.

3. Обзор

3.1. Обзор приложений-аналогов

В 2020 году компания Cellebrite показала новую возможность своего инструмента UFED (Universal forensic extraction device, универсальный прибор для извлечения криминалистических данных), заключающуюся в автоматическом создании скриншотов телефона. Его интерфейс можно увидеть на рис. 1. Это приложение было основным ориентиром при разработке данного проекта.

Рис. 1: Интерфейс устройства Cellebrite UFED



Ещё одним примером продукта, позволяющего создавать скриншоты экрана телефона в автоматическом формате, является приложение «МК Агент» от компании Oxygen Software.

Также есть несколько приложений, позволяющих создавать скрин-

шоты экрана телефона в ручном формате. Примером такого приложения служит Droid-at-screen.

3.2. Обзор технологий

В качестве инструмента, который бы позволил общаться с телефоном через USB кабель, был выбран инструмент под называнием ADB [1] (Android Debug Bridge). Данный инструмент разработан компанией Google и доступен в пакете «Android SDK Platform-Tools». Это единственный инструмент, который позволяет посылать на телефон команды без использования клиент-серверного подхода и общения через HTTP/HTTPS запросы. Этот инструмент позволяет совершать такие операции, как:

1. открытие приложения по указанному уникальному идентификатору;
2. скроллинг экрана;
3. создание скриншота экрана.

Для работы с ADB под C# существует NuGet пакет под названием SharpAdbClient.

Однако в ходе разработки скоро стало понятно, что возможности ADB не позволяют эффективно (точно) работать с графическими элементами приложений для автоматического воспроизведения действий «нажатие» и «скроллинг». В качестве связующего элемента был выбран инструмент для автоматического тестирования Appium [2]. Он позволяет обращаться к элементам Android-приложения по id, имени, имени класса или другим атрибутам. Осуществляет он эти действия через Android-сервис UIAutomator (далее драйвер).

Другим возможным вариантом реализации автоматического взаимодействия с телефоном была установка на смартфон собственного приложения, использующего возможности UIAutomator, которое бы общалось с десктопным приложением через веб-сокеты. Appium был выбран по той причине, что он избавляет программиста от необходимости

реализовывать эту функциональность и предоставляет взаимодействие с телефоном через удобное API.

Работа Appium устроена следующим образом:

1. происходит запуск сервера;
2. осуществляется запуск сессии (в контексте Appium сессия – это сеанс запуска конкретного приложения на телефоне) с необходимыми параметрами (такими как имя устройства и имя уникального ключа запускаемого приложения);
3. десктопное приложение через API отправляет запросы на сервер Appium;
4. сервер отправляет этот запрос на драйвер;
5. драйвер осуществляет необходимые действия и отправляет ответ обратно.

Передача запросов клиент-сервер происходит в формате JSON по протоколу HTTP. Такой подход позволяет работать с инструментом на любом языке программирования. В частности, для работы с Appium под C# существует NuGet пакет под названием Appium.WebDriver.

4. Реализация

4.1. Ограничения

В условиях подключения телефона к сети интернет, велика вероятность возникновения нежелательных уведомлений и всплывающих окон, которые могут помешать корректной работе инструмента для автоматического взаимодействия с телефоном. Поэтому при создании приложения разработчиками компании Belkasoft было поставлено условие: соединение с телефоном происходит по USB-кабелю, а работа осуществляется в авиарежиме.

Основное приложение компании Belkasoft, Belkasoft X, написано на языке C#. Предполагалось, что в случае успешного выполнения всех поставленных задач, реализованная утилита будет встроена в основное приложение, поэтому для простоты внедрения кода было принято решение писать проект на языке C#.

4.2. Подключение телефона через USB-кабель

После подключения телефона к компьютеру через USB-кабель необходимо подготовить его для корректной работы программы:

1. перевести в авиарежим;
2. разрешить отладку через USB;
3. разрешить установку приложений через USB.

4.3. Архитектура приложения

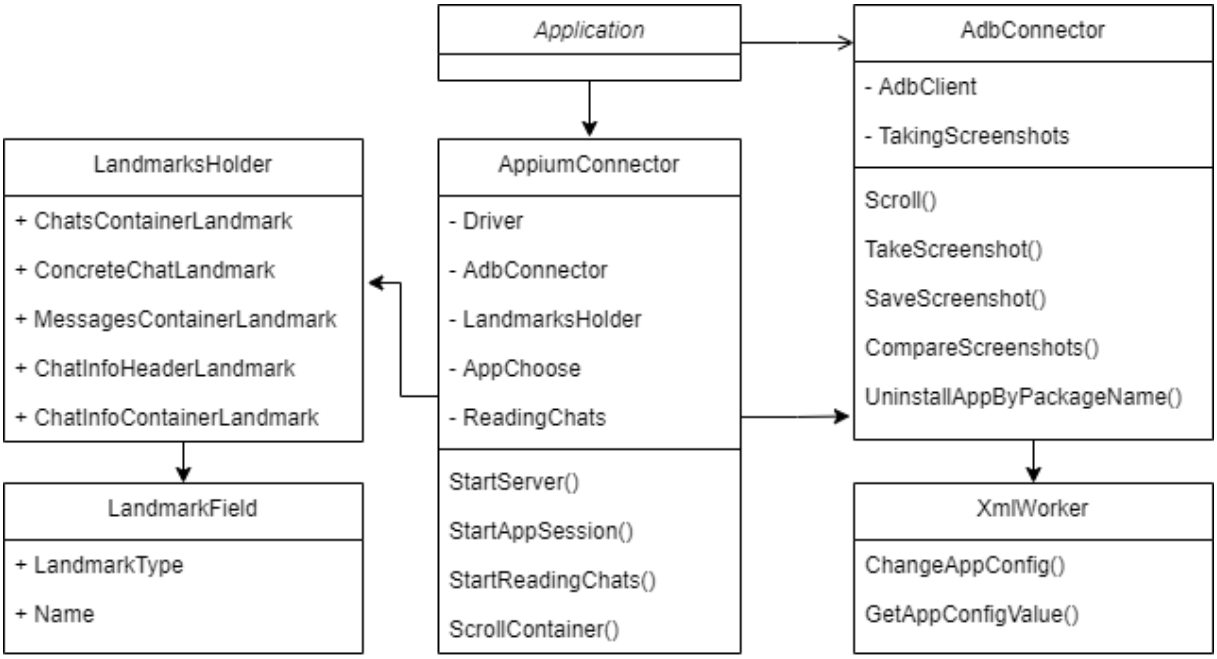
На рис. 2 можно видеть архитектуру приложения.

Основными классами являются **AppiumConnector** и **AdbConnector**: они отвечают за взаимодействие соответсвенно с API Appium и Adb.

Класс **XmlWorker** отвечает за работу с конфигурационным файлом приложения, в котором хранится информация о папке сохранения скриншотов, пути к файлу **adb.exe** и порядковом номере последнего созданного скриншота.

Класс **LandmarksHolder** отвечает за хранение уникальных идентификаторов графических компонентов выбранного для обработки приложения.

Рис. 2: Архитектура приложения



4.4. Создание скриншотов в ручном режиме

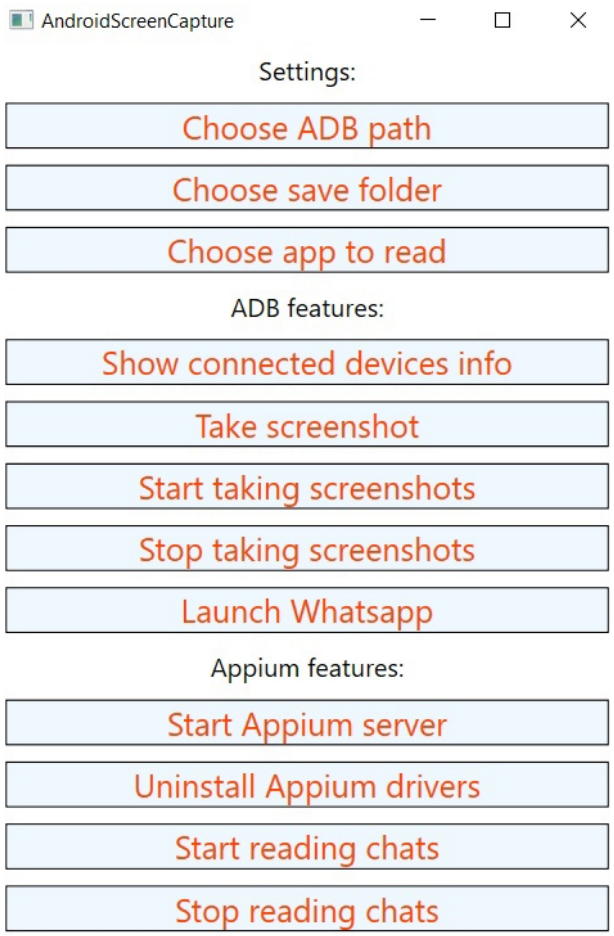
Ручной режим предполагает, что пользователь выполняет какие-то действия на телефоне, а приложение в это время с определённой периодичностью создаёт скриншоты экрана. Эта функциональность полностью реализована через ADB.

Для работы с ADB через C# требуется запустить ADB сервер с указанием пути до файла **adb.exe** и создать объект-клиент ADB.

На рис. 3 можно видеть интерфейс приложения. Для запуска ручного режима создания скриншотов необходимо нажать на кнопку «Start

taking screenshots» в секции «ADB features»

Рис. 3: Интерфейс приложения



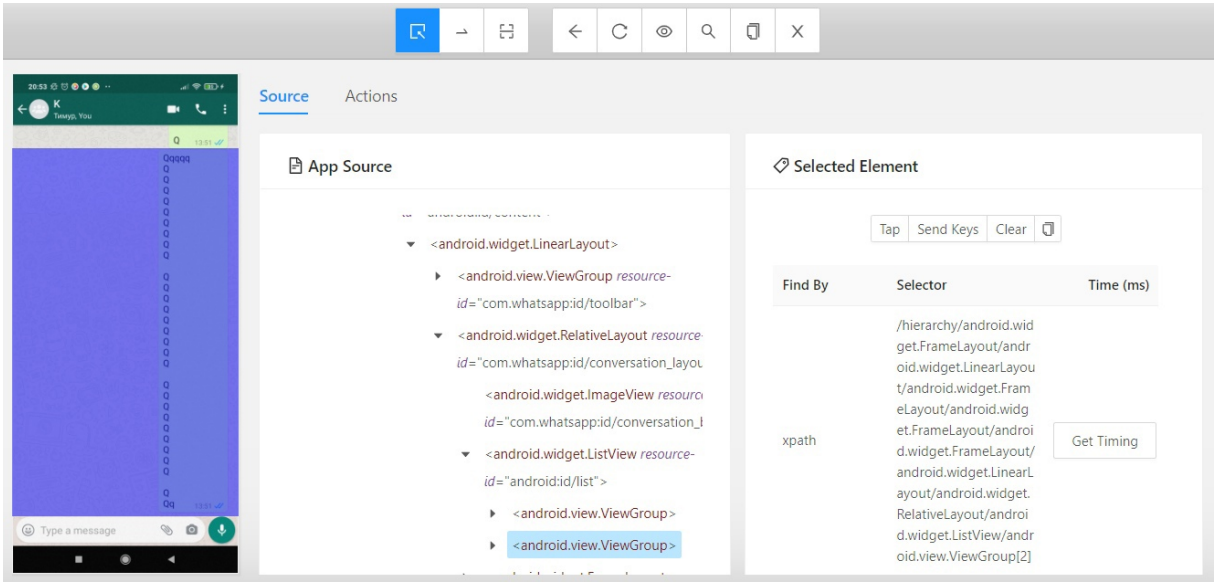
4.5. Создание скриншотов в автоматическом режиме

Автоматический режим предполагает, что приложение осуществляет навигацию по компонентам Android-приложения самостоятельно. Такой подход возможен только в случае, когда заранее известны приложение, обработка которого будет автоматизирована, и идентификаторы компонентов этого приложения.

4.6. Нахождение уникальных идентификаторов компонентов Android-приложения

В десктопном приложении Appium есть возможность запустить сессию, используя графический интерфейс. При старте сессии открывается возможность работы с инспектором. На рис. 4 можно увидеть утилиту, которая по открытому на телефоне приложению позволяет находить уникальные идентификаторы желаемых UI-компонентов. С её помощью осуществлялась обработка мессенджеров.

Рис. 4: Пользовательский интерфейс инспектора



4.7. Запуск приложения (сессии)

Для запуска приложения (и дальнейшей работы с ним) на смартфоне необходимо запустить сервер Appium и подключиться к нему с указанием имён устройства и приложения. Библиотека Appium для C# не позволяет запустить сервер через API, поэтому запуск осуществляется через терминальную команду.

Для запуска автоматического режима создания скриншотов необходимо нажать на кнопку «Start reading chats» в секции «Appium

features». Перед этим необходимо выбрать из предлагаемого списка приложение, которое должно быть обработано.

4.8. Подбор оптимальных параметров работы приложения в автоматическом режиме

У разных мобильных устройств разная производительность и время отклика на события вроде «нажатие» и «скроллинг». По этой причине необходимо выставлять таймеры ожидания отклика этих действий, подходящие для самых малопроизводительных устройств. Путём серии запусков приложения на мобильном устройстве с характеристиками: процессор Qualcomm Snapdragon 615 MSM8939 1500 МГц, видеопроцессор Adreno 405, оперативная память 2 ГБ, которые обеспечивают задержку при взаимодействии с графическими компонентами, было выяснено, что оптимальное время ожидания отклика – около 3 секунд.

Также в процессе работы были подобраны параметры скорости пролистывания экрана и величины погрешности пролистывания, связанной с несовершенностью определения размеров окон приложения.

5. Тестирование

5.1. Устройства

В качестве устройств для тестирования были выбраны два мобильных телефона с операционными системами Android 5.0 и Android 10.0. Вариант с эмуляторами не рассматривался, потому что была необходимость работы с реальными приложениями и реальным контентом.

5.2. Подход

Суть проекта заключается в автоматизации создания скриншотов, и поэтому всё, что было необходимо сделать – это обеспечить программе условия для корректного исполнения, а именно:

1. отключить системные анимации;
2. перейти в авиарежим;
3. включить режим отладки по USB;
4. предоставить разрешения на установку необходимых приложений.

После этого оставалось лишь следить за процессом работы программы, то есть проверять качество создаваемых скриншотов: было необходимо вручную проверять, что скриншот с правильным порядковым номером сохранился в указанную папку и что он соответствуют содрежимому экрана телефона.

5.3. Процесс тестирования

Поскольку основной упор в работе над проектом был сделан на обработку мессенджеров, далее приведены пункты процесса тестирования приложений этого типа:

1. Открытие приложения.
2. Скроллинг и создание скриншотов окна чатов;

3. Открытие чата;

4. Открытие информации о чате, скроллинг и создание скриншотов открытого окна информации;

5. Скроллинг и создание скришотов окна чата (сообщений).

6. Заключение

В ходе учебной практики были выполнены следующие задачи:

1. Проведён обзор предметной области, изучены приложения-аналоги, изучены используемые инструменты;
2. Разработана и реализована архитектура приложения:
   (a) поддержана возможность создания скришотов в ручном режиме;
   (b) поддержана возможность создания скришотов в автоматическом режиме;
   (c) сделано возможным переключение между приложениями, контент которых нужно запечатлить на скриншотах;
3. Протестирована работа приложения на мобильных устройствах с двумя разными версиями операционной системы Android.

На данный момент для корректной работы приложения на телефон требуется установить три сервиса-помощника Appium. В ближайших планах – упаковать три устанавливаемых на телефон приложения в одно под названием BelkasoftScreenCapture.

Код проекта находится в приватном Github репозитории разработчика-консультанта компании Belkasoft. Сделано это по той причине, что написанную утилиту планируется встраивать в коммерческий продукт компании.

Список литературы

[1] ADB. *ADB документация*. URL: https://developer.android.com/studio/command-line/adb (дата обр. 30.05.2021).

[2] Appium. *Appium документация*. URL: https://appium.io/docs/en/about-appium/getting-started (дата обр. 30.05.2021).