**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Android приложение за управление на принтери в търговски обекти

Дипломант : Научен ръководител:

Симеон Чакъров Ивайло Велков

СОФИЯ

2019

Дипломната работа на Симеон Чакъров отговаря на всички изисквания и добри практики за програмиране. Оценката за свършената работа е много висока и определено мога да твърдя, че Симеон Чакъров е положил труд и старание за качественото решаване на проблема.

# **Увод**

Във всеки търговски обект, независимо дали в него се извършват продажби на стоки или услуги, има ЕКАФП (Електронен Касов Апарат с Фискална Памет) или (не)фискален принтер, свързан с компютър. И нека все пак да обобщим какво значи понятието „търговски обект“ и каква е разликата между ЕКАФП и (не)фискален принтер.

Търговски обект е всяко помещение или съоръжение (например магазини, ресторанти, сергии и др.) на открито или под навеси, от което се извършват продажби на стоки и услуги, в които заплащането не става по банков път, независимо от факта, че помещението или съоръжението може да служи същевременно и за други цели (офис, жилищно помещение), да е част от притежаван жилищен имот (гараж, мазе, стая) или да е производствен склад или превозно средство, от което се извършват продажби.

Разликата между (не)фискалния принтер и ЕКАФП е това, че касовият апарат функционира самостоятелно – купуваш касов апарат и работиш с него – това е. Докато принтерът не работи самостоятелно – свързан е с компютър и работи със специална програма – ресторантьорска или склад-магазинска. Работата с тази програма има предимства, разбира се – поддържа складовите наличности, сигнализира за изчерпващи се стоки, улеснява ревизиите.

Примерна система предназначена за работа в бар/ресторант се състои от: Компютър, монитор, периферни устройства, фискален принтер, каса.

Какво става когато се развали част от системата? Или например е много трудно или невъзможно инсталирането на системата заради местоположението на различните компоненти? Вадиш касовия апарат? А как ще продължиш да следиш складовите наличности? На листче? Как ще покажеш подробна сметка на своите клиенти, които не отиват в магазинчето отсреща, само защото ти издаваш подробни бележки, за разлика от него, и сега ще ги разочароваш? Ако притежаваш заведение ще ги накараш да ровят в менюто за да си разделят сметката? Не! Реших да се захвана с решението на подобен проблем именно защото освен за самите клиенти, е удобно и за самия търговец. Затова започнах с конструирането на Smart Printer for Android - Android service, през който човек може да управлява своя (не)фискален принтер.

# ПЪРВА ГЛАВА

# Обзор за съществуващите решения

## **1.1 Подобни съществуващи решения**

След подробно и дълго проучване открих, че няма друг аналог на Smart Printer for Android, освен Desktop приложение, с което се запознах по време на лятната си производствена практика, организирана от училище. След запознаване с документацията, прочитане на коментари и много въпроси от моя страна към разработчиците на това приложение, дойде идеята да се осъществи подобна апликация, но за Android, която освен функциите на Desktop приложението, да поддържа и функция за управление на повече от един принтер. Освен това, като се замислих, при разположение на уредбата би било много по-лесно и по-практично да се управлява през телефон, отколкото през компютър.

# ВТОРА ГЛАВА

# ПРОЕКТИРАНЕ НА СТРУКТУРАТА

## **2.1. Функционални изисквания**

Smart Printer for Android няма да е абсолютно автоматизиран продукт, който сам да работи. То ще очаква да получи някаква заявка с това, което трябва да принтира самия принтер.

То е Android приложение, което само в себе си има много на брой драйвери за различни принтери. В самите драйвери са описани характеристиките на самите принтери(Baud rate – скоростта, с която печата принтера, code page-a му и др.), както и различни функции – общи и такива, използващи се в зависимост от това дали принтера е фискален или нефискален.

След като получи заявка, дали от настроен търговски софтуер, пригоден за Smart Printer for Android, или Postman, или лентата в браузъра, то започва да я раздробява(има в себе си Parser), обработва и след това я изпраща към принтера.

Приложението се дели на две основни части:

* Първият панел

Първият панел има етикет, изписващ името на приложението, а под него има две полета за въвеждане. В едното поле, ако си свързан към локална мрежа, показва IP-то, а другия – порта, през който се очаква заявката, съдържаща XML-а. Под двете полета има още един етикет, който изписва съответно със зелено “Connected”, ако си свързан към локална мрежа, и съответно с червено „Not connected“, ако не си. Също така под гореизброените има бутон, водещ към втория панел.

* Вторият панел

Вторият панел има три падащи менюта, единият от които е да се избере в коя държава се използва принтера, другият дава възможност да се избере езикът, на който искате да бъде преведено приложението.

Чрез третото падащо меню може да се избере активния принтер, тоест този, с който работите, и да се добави. Може да работите с повече от един принтер едновременно. При добавянето на принтерите, могат да се зададат скриптове:

* + - Headers – текст, който ще се изписва отгоре на бележката
    - Details – по средата
    - Totals – там, където пише общата сума на покупката
    - Footers – най-отдолу на касовата бележка

Има първоначален шаблон, който управлява данните, които са ни изпратени и разпределя кое къде да стои на бележката. Ако клиентът има изискване към подредбата на текста в бележката, то има опция за осъществяването му чрез тези скриптове.

При добавяне на скрипт, шаблона по подразбиране, в дадената секция се изтрива и се изписва само това, което е зададено. Има по един бутон за всеки вид скрипт, който добавя по още едно поле за въвеждане, в което може да пише човек, т.е. всеки вид скрипт може да има по повече от един ред/команда – колкото ние поискаме.

Можем да изтриваме и редактираме добавените принтери. Под самото падащо меню, в което се намират активните принтери, има още два бутона – този за редактиране, обозначен с моливче и такъв за изтриване на избрания принтер, обозначен с минус.

При натискане на бутона за редактиране, излиза плавно панел от едната страна, в който можем да редактираме скриптовете, зададени при добавяне на устройството и настройките на самото устройство.

Идеята на тези скриптове е ако искаме да променим нещо, да го направим, например от коя страна да е името на продукта , къде да се намира общата дължима сума и др.

Има и още един бутон, който като се цъкне, отново излиза от едната страна плавно панел, в който биват изписани активните принтери и това дали те са в изправност и готови за печатане.

## **2.2. Съображения за избор на програмни средства и развойната среда**

Smart Printer for Android трябва да е лесно за използване от всеки потребител, търговец, на различна възраст. Затова то трябва да е направено така че човек директно да може да се ориентира. Много семпло и просто откъм визуалната част. Поради тази причина избрах да използвам един много функционален език за създаване на Android приложения, а именно Basic4Android.

Едни от предимствата на езика са:

* Първото предимство на езика и средата за разработка е простият, но в същото време много мощен RAD (rapid application development) tool, който позволява на програмисти на всякакво ниво бързо да разработват реални приложения, не е трудно да се започне и да се научи – IDE-то и framework-а сами ще ти помогнат да завършиш работата си.
* Цялостна среда за разработка и език за програмиране, фокусиран 100% върху развиването на Android приложения.
* Компилира се до native bytecode и не са нужни Runtime библиотеки.
* APK файловете са същите като APK файловете, направени с Java / Eclipse / Android Studio.
* Можеш да използваш Java код, обгръщайки го като библиотека и да го свържеш от средата за разработка на B4A приложения.
* Представянето/производителността са много близки до тези на приложение, създадено с Android Studio Java.
* Бърз debugger, поддържащ бърз deployment (процес, който подготвя приложението да работи в съответната среда – включва инсталация и конфигурация, оптимизиране).
* Debugger-ът е step-by-step и поддържа expressions watcher (минавайки през кода, може да се следи стойността в даден израз/променлива).
* Изключително разтегателен език – не те ограничава по никакъв начин, освен големия набор от библиотеки, той поддържа и custom Java библиотеки.
* Визуален редактор, поддържащ няколко екрана и резолюции.
* Мощна функционалност за дизайнерски скриптове, чрез която лесно се създават сложни макети (layouts).
* Поддържано е от всички Android устройства с Android над 1.6.
* Модерна среда за разработка с вградено автоматично довършване.
* Голяма по обем документация и форум, в който при зададен въпрос, се отговаря адекватно и бързо.
* Можеш да използваш B4J за да имплементираш back-end частта и можеш да използваш B4I за да създадеш същото решение за iOS. 80% от кода от всяка от средите за разработка може да се използва в другите две.

Езикът е алтернатива на програмирането с Java и Android SDK. Платформата компилира до Java и след това отново го връща като Basic, използва смесен (Cross) компилатор. Резултатът е изпълним код, предназначен за платформа различна от тази, на която е извършена компилацията. Използва се например при създаване на код за вградени системи (embedded systems), както и в Basic4Android, където няма възможност, или е по-трудно, да се създават програми.

B4A включва визуален дизайнер, който опростява процеса на изграждане на потребителски интерфейси, насочени към телефони и таблети с различни размери на екрана. Компилираните програми могат да бъдат тествани в емулатори на AVD Manager или на истински Android устройства, използващи Android Debug Bridge и B4A Bridge.

Самият език е подобен на Visual Basic и Visual Basic .NET, но е адаптиран към Android средата. Има разлики и прилики с Java, най-големия плюс е, че се пише много бързо и е много функционално, не е гъвкаво колкото Java, но по отношение на приложението, което е нужно за системата и ще се използва, е нужно просто бързодействие и лесно разгадаване на програмата.

За да може да работи приложението освен компилатора, който се предоставя с B4A платформата е нужно да се свърже към нея Java JDK, за да може да се установи компилация, Android SDK за да се инсталират платформите и инструментите за версиите на Android-а, който ще се използва. Също така най-новата версия на B4A е платена и е нужен лиценз.

По желание може да се използва B4A-Bridge, което компилира на устройството безконтактно чрез безжична връзка, ако двете устройства са в една и съща мрежа. Може да се изтегли от Google Play безплатно.

B4A има пробна версия, която може да се използва 30 дни, но не с всички функционалности, както ако се закупи. Може да се закупи стандартна версия, която включва поддръжка само 2 месеца, а може и Enterprise версия, включваща актуализиране на софтуера 2 години.

Другия малко по-гъвкав, но не практичен вариант, поради усложняване на работата за създаването на Android приложението е писането на Java на Android Studio. Но използването на Android Studio ще създаде повече работа по програмирането на това приложение, без да има нужда от нея.

Android Studio е официалната интегрирана среда за разработка (IDE) за операционната система Android на Google, изградена върху софтуера IntelliJ IDEA на JetBrains и специално разработена за разработка на Android. Той е достъпен за изтегляне за операционните системи Windows, MacOS и Linux. Това е заместител на Eclipse Android Development Tools (ADT) като основно IDE за разработка на естествени Android приложения. Android Studio е обявено на 16 май 2013 г. на конференцията Google I/O. То е било в начален етап на предварителен достъп, като се започне от версия 0.1 през май 2013 г., след което е въведен бета етап, като се започне от версия 0.8, която е пусната през юни 2014 г. Първото стабилно изграждане е пуснато през декември 2014 г., като се започне от версия 1.0. Сегашната стабилна версия е 3.2.1, пусната през октомври 2018 г.

## **2.3. Преглед на част от принтерите, свързващи се с приложението и работата с тях**

* + SPP-R300

Фигура 2.3.1



**Изисквания за Bluetooth принтерите:**

За да се свърже принтерът с Android мобилен телефон, използвайки Bluetooth, устройствата не трябва да са на повече от 10метра разстояние.

Това разстояние може да се скъси, ако например има някакви препятствия между двете устройства (да кажем стени). Ако друго устройство (като микровълнова печка или рутер за безжичен интернет), което използва същите честоти на предаване, е в близост до тях, предаването на информация може да бъде прекъснато. Преместете принтера и мобилното устройство от причинителя, имащ същите честоти на вълните на поне пет метра.

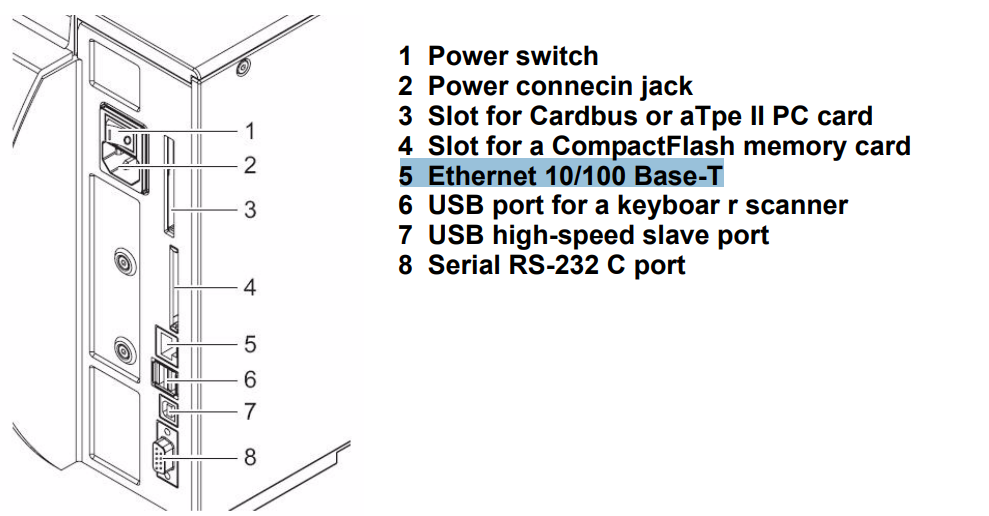
**Как да свържем принтера към устройство:**

1. Проверяваме дали всички устройства са включени, ако не са – ги включваме.
2. Пускаме Bluetooth-а на телефона.
3. Намираме принтера, който трябва в случая да се казва SPP-R300.
4. Свързваме се с него, избирайки го. Ще се покаже неговия MAC address, зададен от драйвера в приложението.
5. Тестваме връзката, като принтираме.
   * PR300

Фигура 2.3.2



Фигура 2.3.3



**Изисквания за LAN принтерите:**

При LAN принтерите идеята е да се свърже чрез Ethernet cable към мрежата и през нея да се изпращат от Android Service-a инструкции, подадени преди това от даден софтуер или проста заявка, като XML структура. Стойностите по подразбиране(IP адрес и порт) се показват, зададени от драйвера и ако се сменят в принтера, то може да се сменят и в приложението, така че да изпрати информацията на правилното място.

* Rongta RP326 COM

Фигура 2.3.4



**Изисквания за Serial принтерите:**

Този конкретен принтер може да се свърже и чрез мрежа, и чрез сериен порт(Serial Port), но в случая ще го използваме като пример за свързване на принтера към мобилното устройство(Android Service-a) чрез сериен порт.

Свързва се с телефона чрез кабел(нужен е преходник от Serial Port към USB и след това OTG (On – The - Go) USB - преходник от USB-то към мобилното устройство), като през всичко това преминават инструкциите, подадени в XML формат. Има стойности по подразбиране за baud rate – скорост на информацията, която се предава(bit per second). Има възможност от приложението да се даде различна стойност, но преди това трябва и принтера да е настроен на тази ‚бързина‘, така че да не стават грешки.

OTG е съкращение от USB OTG (Universal Serial Bus On – The - Go) и представлява стандарт, който позволява на устройствата да ‘разговарят’ помежду си. Например, може да свързвате телефона си към четци на SD карти, камери, дори мишка или клавиатура и други различни периферни устройства.

USB (Универсалната серийна шина) е серийна шина за свързване на компютър с огромен набор от различни периферни устройства.

Фигура 2.3.5 – USB OTG



Има и драйвер за виртуален принтер, който визуализира касовата бележка на екрана на устройството, използвайки библиотеката ScrollView2D.

## **2.4. Принцип на софтуерния модел**

При стартиране на Android приложението, се показва екран, информиращ ни за статуса на връзката ни с мрежата. След натискане на бутона за настройки влизаме в следващия панел, в който вече можем да изберем език на приложението и в коя страна се използва то. Също така вече започваме да работим с принтерите –добавяме към тях и скриптовете им, редактираме ги, трием ги при нужда. Също така има екран, показващ статуса на принтера – дали успешно се е свързал, приел и обработил заявката, подадена от нас.

Фигура 2.4.1

C:\Users\simeon\Downloads\Untitled Diagram.png

## **2.5. Преглед на библиотеките използвани за Android приложението**

* JavaObject

Целта на библиотеката JavaObject е подобна на целта на библиотеката за Reflection. И двете библиотеки ви позволяват да се ‚обаждате‘ директно на Java APIs въз основа на функциите за отразяване на Java. JavaObject е различен от библиотеката Reflection и в повечето случаи е по-лесен за използване. JavaObject обаче не замества библиотеката за Reflection, тъй като не поддържа всичките й функции. В много случаи можете да използвате двете библиотеки заедно (и двете са леки библиотеки). Подходът JavaObject е по-ориентиран към обекта. Декларирате обект JavaObject, който след това "обвива" всеки друг обект и предоставя три метода: SetField, GetField и RunMethod. Променливата JavaObject е подобна на променлива Object с добавянето на методите за отражение.

* Random Access File

Този обект ви позволява да осъществявате последователно достъп до файловете и байтовите масиви. Можете също така да го използвате, за да кодирате номера до байтове (и обратно). Имайте предвид, че файловете с активи (файловете, добавени с файловия мениджър) не могат да се отварят с този обект, тъй като тези файлове всъщност са пакетирани във файла APK.

* Reflection

Тази библиотека съдържа обект Reflection, който позволява достъп до методи и полета от обектите на Android, които не са изложени на езика Basic4Android. Прави това чрез съоръжение, наречено "Reflection", което използва метаданни за обекти, които са включени в пакета за приложения и позволява динамичен достъп до полета и методите по време на изпълнение.

* XmlSax(SaxParser)

Анализатор, който последователно прочита поток и повдига събития в началото и края на всеки елемент.

* OkHttpUtilis2

HTTP е начинът, по който модерните приложения комуникират. По този начин ние разменяме данни и средства.

Използвайки HTTP по ефикасен начин, пратените данни биват заредени по-бързо и спестяваме честотната лента.

OkHttp е HTTP client, който е ефикасен по презумпция:

* HTTP/2 поддръжката позволява всички заявки към един домакин да споделят socket.
* Обединяването на свързвания намалява латентността на заявката (ако HTTP/2 не е достъпен)
* Прозрачният GZIP намалява размерите на изтегляне
* Кеширането на отговорите избягва напълно мрежата за повторни заявки.

OkHttp продължава, когато в мрежата изникнат проблеми- тя тихо ще се възстанови от усложненията с връзката. Ако вашият service има множество IP адреси, OkHttp ще опита алтернативни адреси, ако първото свързване не успее.

Това е необходимо за IPv4 + IPv6 и за service-и, хоствани в резервни центрове за данни. OkHttp поддържа модерни TLS характеристики (TLS 1.3, ALPN, certificate pinning).

Използването на OkHttp е лесно. Неговото API за заявка / отговор е проектирано с безупречни ‘builders’ и ‘immutability’. Той поддържа както синхронни блокиращи повиквания, така и асинхронни разговори с обратни повиквания.

HttpUtilis е базирана на Android версията на Apache HttpClient. Google престана да актуализира този SDK преди много време и те го премахнаха от Android M. HttpUtils2 също се основава на Http библиотека.

Библиотеката OkHttp се основава на SDK на OkHttp.

Тъй като това е по-нов SDK, той поддържа нови функции като SSL SNI, SPDY, Patch заявки и по-добра производителност.

OkHttp API wrapper-ът е почти идентичен на Http API library. Основната разлика е, че имената на обектите започват с Ok (OkHttpClient, OkHttpResponse и OkHttpRequest).

Тази библиотека изисква Android 2.3+.

Модифицираната версия на библиотеката HttpUtils2 (OkHttpUtils2) е базирана на OkHttp вместо Http.

V1.01 - Съвместим с B4J.

V1.00 - Добавя поддръжка за проверка на автентичността.

OkHttpUtils2 v2.20 добавя поддръжка за многостранни заявки.

* USB

Дава достъп до свързаните USB устройства. Също така съдържа съответните константи. За тази библиотека е необходим Android SDK 12 или по-нова версия (Android 3.1 или по-нова версия)

* felUsbSerial

Това е алтернатива на UsbSerial2(просто поддържа една идея повече устройства), което е пък разширена версия на основната UsbSerial библиотека.

Добавено е поддържане на Prolific PL2303 USB към serial конвертори, Android ADK устройства и USB permissions. Всички устройства използват един и същи прост интерфейс, предназначени за използване с AsyncStreams и AsyncStreamsText. Базирана е на същата ’Android USB host serial driver library’, като съществуващата UsbSerial, но вече не се нуждае от отделен jar файл, тъй като изходният код на проекта е включен в библиотеката.

Специфичните подобрения на библиотеката над оригиналната са:

* UsbPresent, HasPermission и RequestPermission са добавени, за да идентифицират всяко свързано устройство или аксесоар, достъпни за библиотеката и да се справят с разрешенията за достъп до него.
* SetParameters, използващи се след Open(), и константите за SetParameters осигуряват достъп до всички параметри на серийните линии, вместо само до скоростта на предаване.
* DeviceInfo предоставя низ, съдържащ информация за устройството. Това работи само за ‘slave’ устройства.
* Android аксесоари, които са устройства в режим ‘host’, се разпознават и могат да се използват по същия начин като другите устройства в ‘slave’ режим.

## **2.6. Получени резултати**

Полученият резултат е Android service управляващ принтери, с който се работи лесно, без притеснения за разположение на уредбата и без притеснения, че нещо може да се повреди от нея. Работи за трите вида свързване на принтерите, успешно заменя компютъра, с който трябва да се съобразяваш къде трябва да го поставиш и да има известни притеснения, че нещо може да се повреди. Освен съображенията за мястото на използване на компютъра, с това приложение печелим бърза и лесна инсталация на софтуера на самото устройство.

Удобен и лесен за използване дизайн на приложението.

Лесно пренасяне и монтиране на хардуерната част, съставяща се от едно Android устройство.

* + Мнението на хората

Проектът е представен пред магазин “Twins” в кв. Павлово. Мениджърът на магазина беше изненадан от идеята и постигнатия резултат. Мнението му беше категорично положително. Той заяви, че тази система ще улесни него и персонала в магазина, както и че няма да са нужни много кабели, които да се влачат из целия магазин, в които постоянно се препъват и че най-вероятно изобщо няма да е необходимо да извиква поддръжка от фирмата, от която е купена инсталацията и софтуера, през месец. Както и това, че ако реши да направи преаранжиране на магазина, няма да го има проблема с това къде трябва да стои компютърът.

* Плюсове
* Улесняване на касиерите
* Бързо и лесно инсталиране на уредбата
* Работа с повече от един принтер, едновременно
* Манипулиране на текста, както и структурата на бележката, както ние пожелаем
* Подобряване самочувствието на работещите в магазина поради иновативните технологии, използващи се от тях спрямо други магазини
* Оставя впечатление у клиентите
* Минуси - засега не са установени, поради простата причината, че все още не е пуснато в употреба в реална среда на работа.

# ТРЕТА ГЛАВА

# ПРОГРАМНА РЕАЛИЗАЦИЯ

Продуктът за управление на принтери в търговски обекти, наречен “Smart Printer for Android” е Android приложение, състоящо се от два главни екрана.

Панел, върху който се построява екран с данни за устройството – името му най-отгоре, IP-адреса и порта, на който той слуша.

Под тях има малък етикет, оказващ дали сте свързани към локална мрежа.

При цъкането му, той опреснява информацията за връзката.

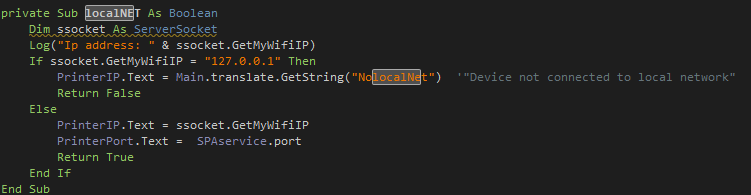
До него има бутон за преминаване към втория екран на приложението.

Фигура 3.1



Ето и частта от кода, проверяваща свързаността на устройството:

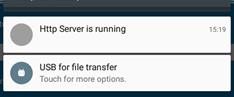
Фигура 3.2



Във функцията localNet се определя дали устройството е свързано към локална мрежа. Чрез ServerSocket(той позволява чрез него други машини да се свържат с тази, слушайки даден порт и позволява комуникация с един или повече клиенти) се проверява дали е свързано към локалната мрежа, в която трябва да е свързан и принтерът и следователно известява търговеца.

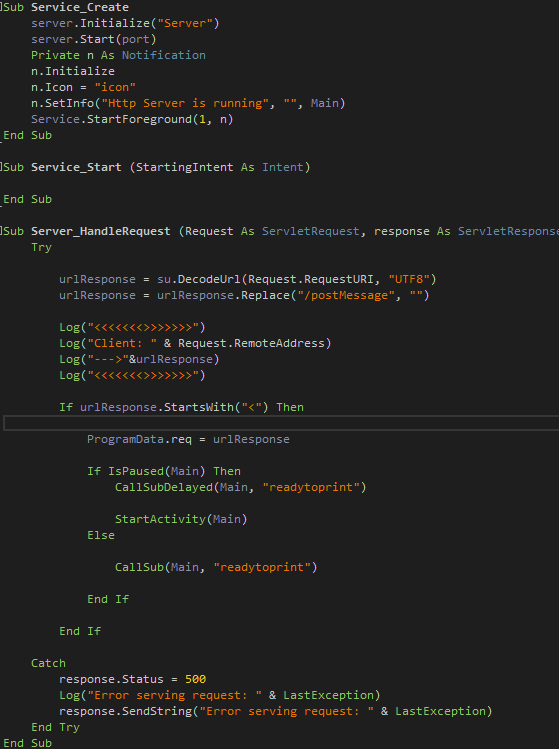
Взета е превантивна мярка – когато приложението се изчисти от полето с работещи апликации (търговецът случайно го изключи), то няма да спре Service-а, а напротив, ще стои, докато не го изключим. Също така то ще ни е оставило в центъра за уведомления съобщение, че все още работи:

Фигура 3.3



При получаване на заявка, service-a ще пусне отново Activity-то и ще изпрати данните за принтиране.

Фигура 3.4



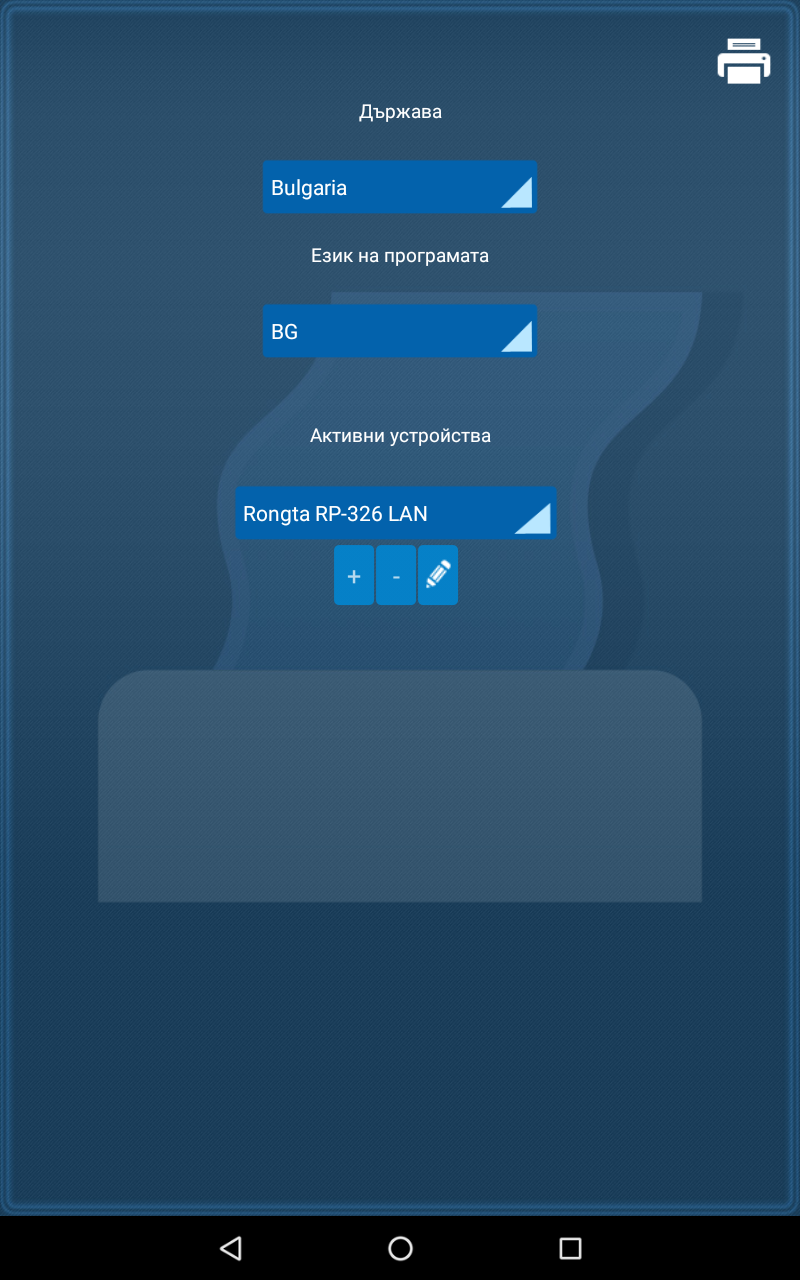
Проверката дали взетата нужна част от заявката започва с „<“, е защото ако има някакъв extension(за блокиране на реклами например) в браузъра, то се изпраща заявка от типа “favicon.ico/” – бъг в браузъра.

След като се натисне бутонът, обозначен с общоприетата картинка за настройки, се влиза във втория панел.

Там се поместват няколко падащи менюта.

* От първото се избира държавата, от която се работи с приложението.
* От второто се избира езикът, на който търговецът, работещ с приложението, иска да му бъдат изписвани надписите.

Фигура 3.5

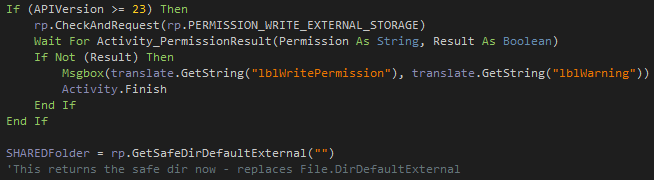


Когато се изберат тези настройки, биват запазени в RAF(random access file). Поведението на RAF е като голям масив от байтове, държащи се в системата, те могат да бъдат и прочетени, и презаписани.

Ако targetSdkVersion е по-малък от 23, тогава системата за стандартни разрешения ще се използва на всички устройства, включително Android 6+, но скоро всички приложения на Google Play ще трябва да зададат targetSdkVersion на 26+.

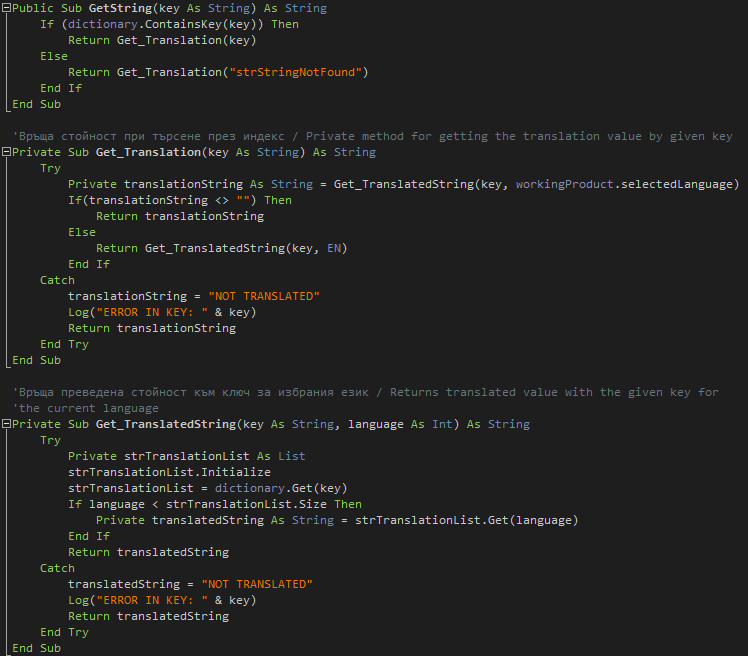
B4A v6.0 добавя поддръжка за разрешения за изпълнение. Хубавото на разрешенията по време на изпълнение е, че потребителят няма да бъде попитан за него, когато инсталира приложението ви от Google Play. Вместо това те ще бъдат помолени да одобрят "опасни" разрешения по време на изпълнение. А записването в RAF е такова.

Фигура 3.6



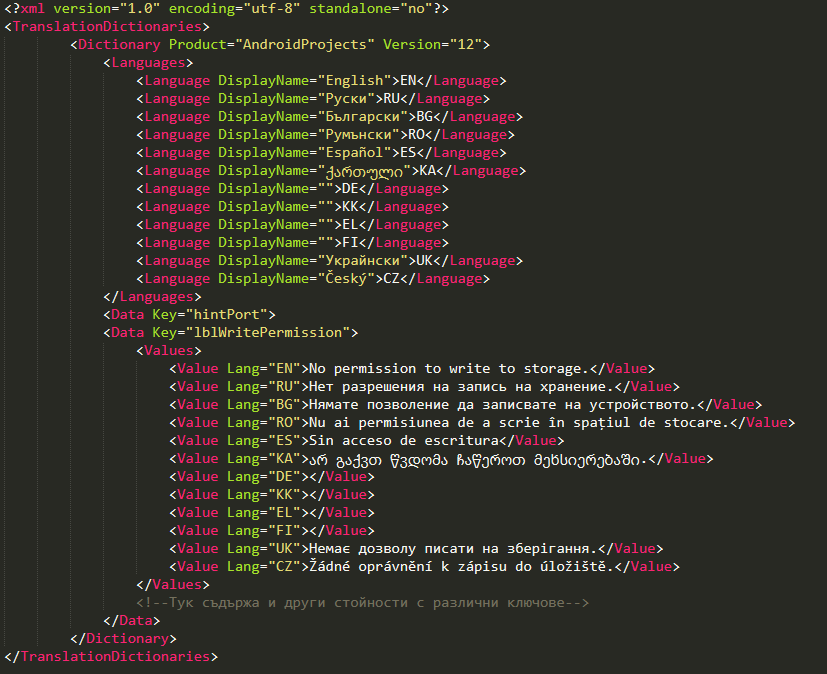
При избиране на език всеки текст от приложението се превежда, както и касовите бележки, които ще се издават, като се взима дадения израз от map, съдържащ раздробен XML файл чрез Parser.

Фигура 3.7



Ето и шаблон на XML-a, използващ се от преводача:

Фигура 3.8



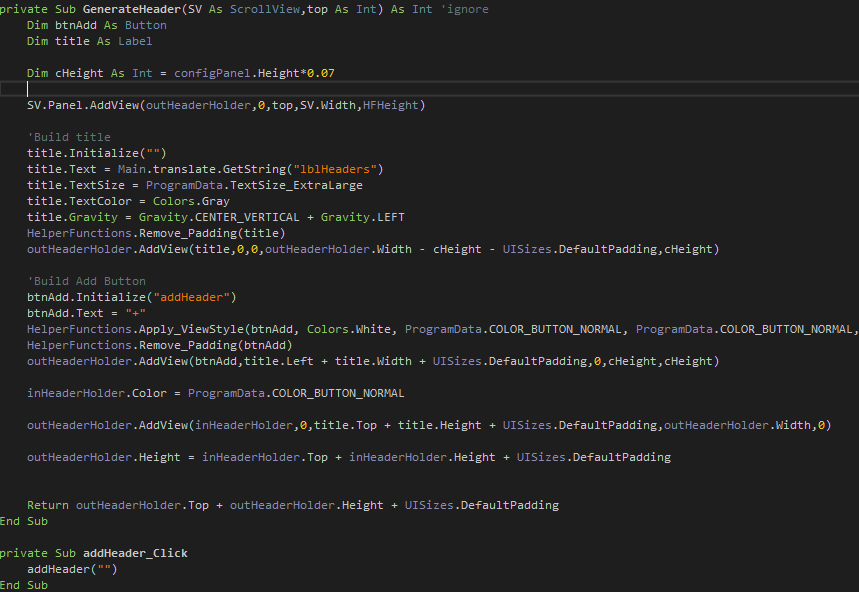
Има още едно падащо меню, което ни дава възможността да виждаме активните принтери и да изберем принтер, който искаме да редактираме или да го изтрием през тези бутони под него – с моливчето е за редактиране, с минуса се изтрива принтера от списъка, а плюса е за добавяне на принтер.

Когато се цъкне добавяне на принтер, отстрани излиза плъзгащ се панел, в който има падащо меню, в което са вкарани всички поддържани принтери от приложението. Можем да добавим принтер, с който искаме да работим, избирайки един от тях, и да ги настроим, също така и при желание за някаква промяна в печатането на касовата бележка- да променим първоначално зададения шаблон чрез ‚скриптове‘.

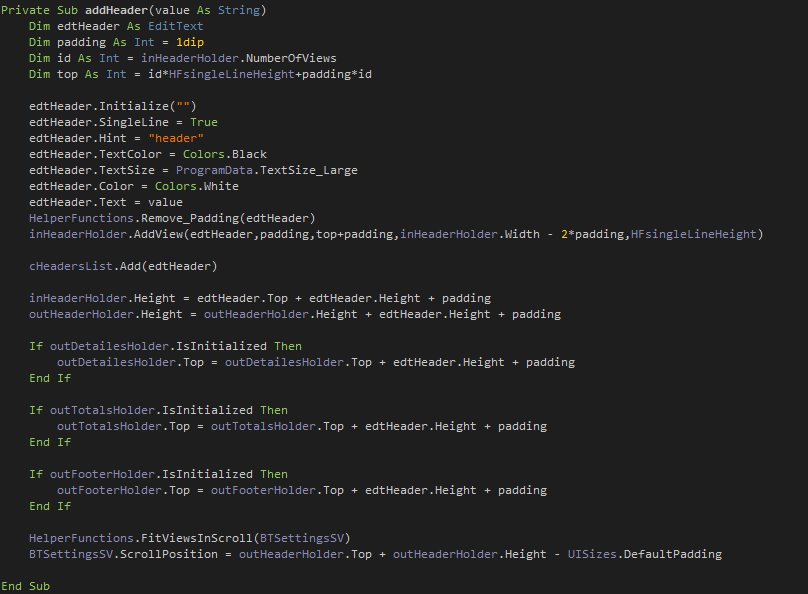
‚Скриптовете‘ са разделени на четири части – Headers, Details, Totals и Footers – точно както се разделя една касова бележка. Може да се добави по повече от един ‘script’ от всеки вид.

Визуализацията се осъществява по този начин, аналогичен за всеки от тях:

Фигура 3.9

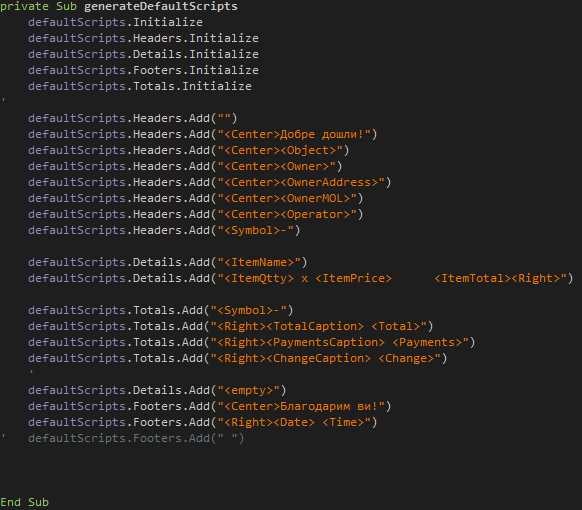


Фигура 3.10



Реализиран е шаблон на касовата бележка по подразбиране:

Фигура 3.11



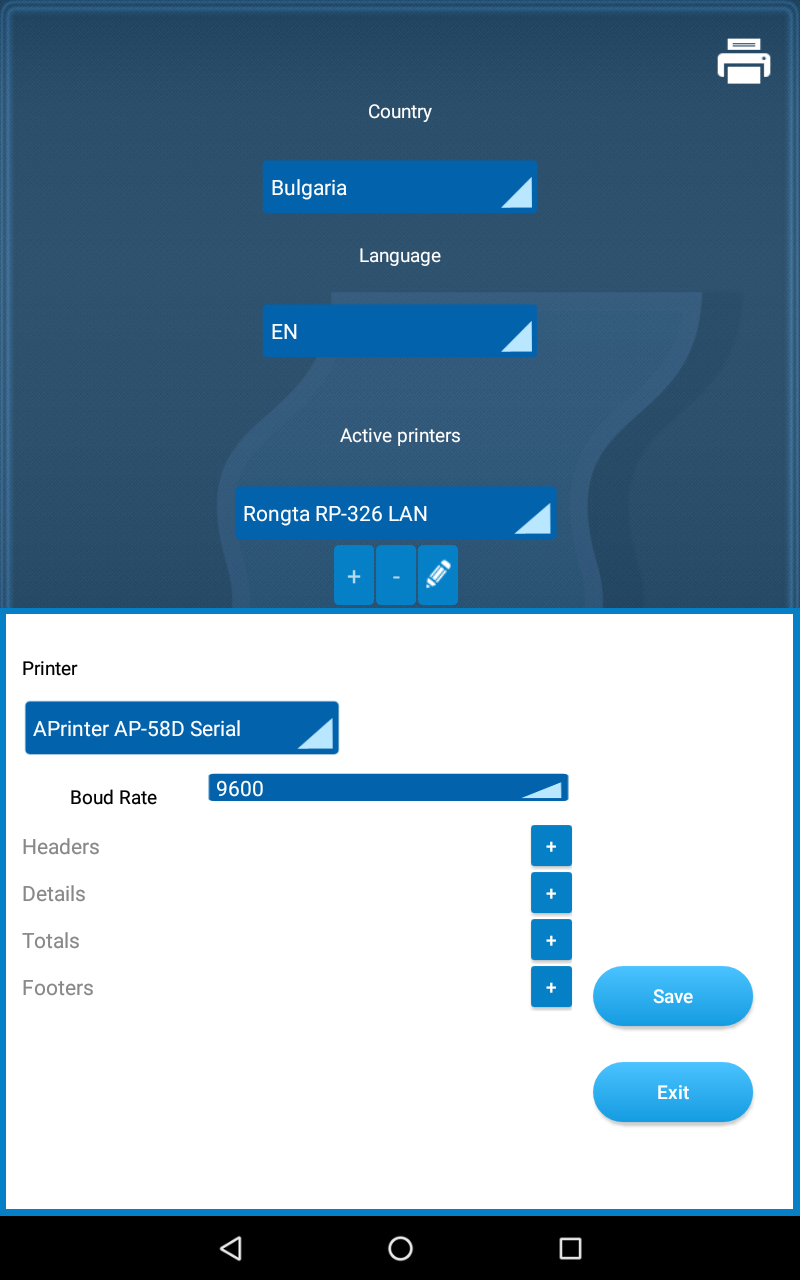
При промяната на даден участък от касовата бележка (Headers, Details, Totals, Footers) ще се изчисти и промени само той, останалите нямат да бъдат засегнати.

Визуализация на екрана за добавяне на принтер:

,

а

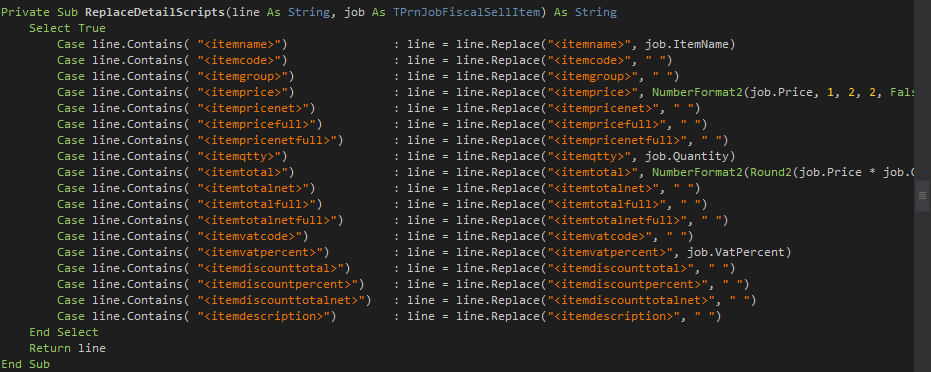
фигура 3.12



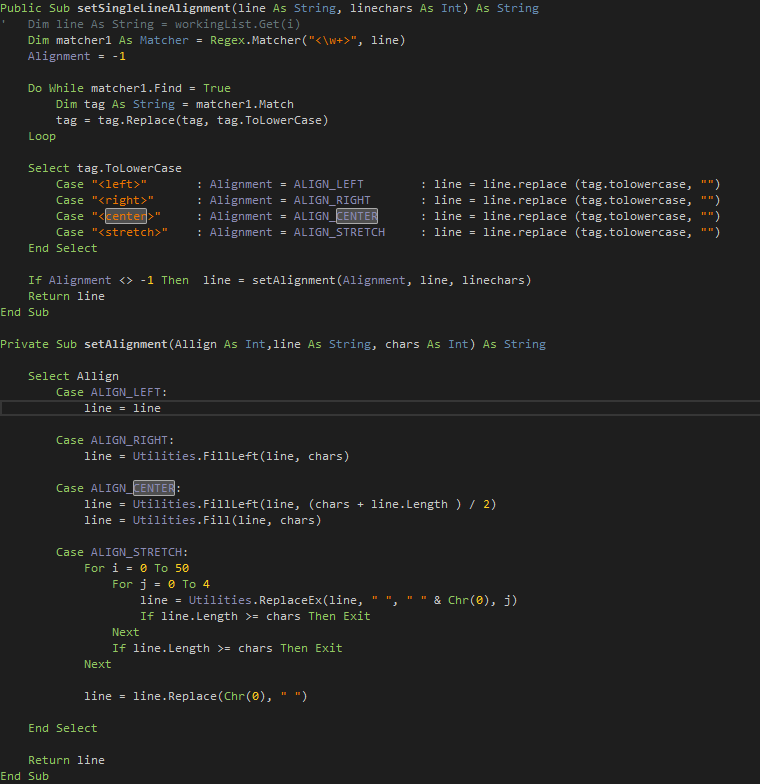
Всяко подразделение на касовата бележка ( Headers, details, totals, footers) има собствени команди, достъпващи елементите и имат и общи, които отговарят за това къде и как да бъдат разположени и изобразени те.

Пример за елементите в Details и за командите за форматиране:

Фигура 3.13

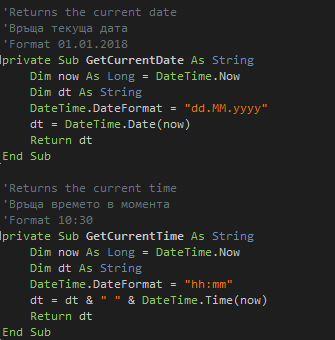


Фигура 3.14



В скриптовете по подразбиране най-отдолу се изписват датата и часа, взети по този начин:

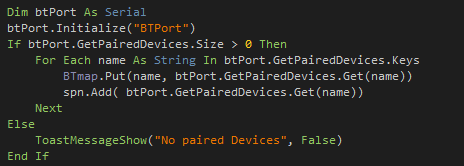
Фигура 3.15



При различните видове принтери има и различно показващи се характеристики:

* За ‘Serial’ принтерите е Baud rate. Baud rate показва скоростта на предаване в бит/секунда. Принтерът и комуникационната линия (COM port) трябва да са с един и същи Baud rate. При смяна на настройката в принтера, има възможност и за смяна в самото приложение за това на единица време колко информация да изпраща.
* За ‘LAN’ принтерите е техният IP address и IP port. Отново, при смяна в принтера, можем да сменим IP адреса, на който устройството изпраща информацията, така че да съвпадат с това на принтера.
* За ‘Bluetooth’ принтерите е техният MAC (Media Access Control) адрес. Всяко Bluetooth устройство има собствен MAC Address. Затова ние трябва да го вземем. Правим го като благодарение на библиотеката Serial прочитаме всички свързани устройства с нашето и вкарваме в падащото меню техните MAC адреси. След това сами трябва да изберем кой MAC адрес ни трябва и да го запазим. Разбираме като прочетем в документацията на самия принтер, или ако сме я изгубили - изтрием всички устройства и скачим само нашия принтер.

Фигура 3.16



Останалите характеристики се задават в драйвера на принтера:

* дали е фискален или не
* кодовата страница (Code page), която е специфичен набор от печатащи се и контролни символи с уникални номера и това къде да я търси
* Задаване кога принтера да отреже касовата бележка и колко празни реда да има преди това
* Задаване каква е връзката, осъществяваща се с устройството и др.

В драйвера се задават и “Job”-ове, за да се изпратят и изпълнят командите в принтера.

Фигура 3.17



Вдясно от скриптовете има два бутона – „Запомни“, ако искаме да запаметим принтера с настройките, конфигурирани от нас и „Изход“, ако искаме да излезем без да запомняме никакви настройки.

Разбира се, можем да добавим повече от един принтер.

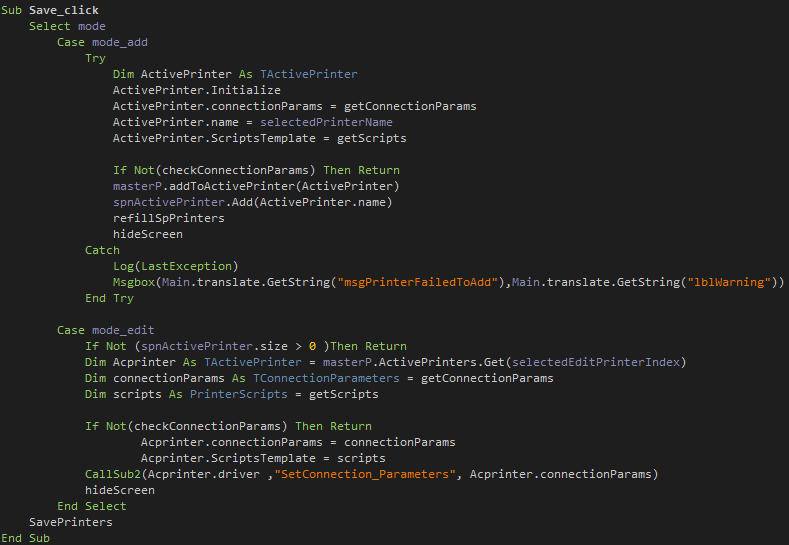
При натискане на моливчето можем да редактираме избрания принтер от падащото меню с активните принтери, като пак излиза плъзгащ се панел.

Изписва се името на принтера, съответната характеристика спрямо вида принтер, полетата за скриптовете и какво е добавено в тях съответно при добавянето на принтер и отново два бутона – един за запомняне на настройките и един за изход.

Освен да редактираме скриптовете, зададени при добавянето на принтера, отново можем да добавим още редове за тях.

Когато цъкнем “Save” бутона, дали когато редактираме или добавяме принтер, се запазват отново в RAF принтера и неговите особености:

Фигура 3.18

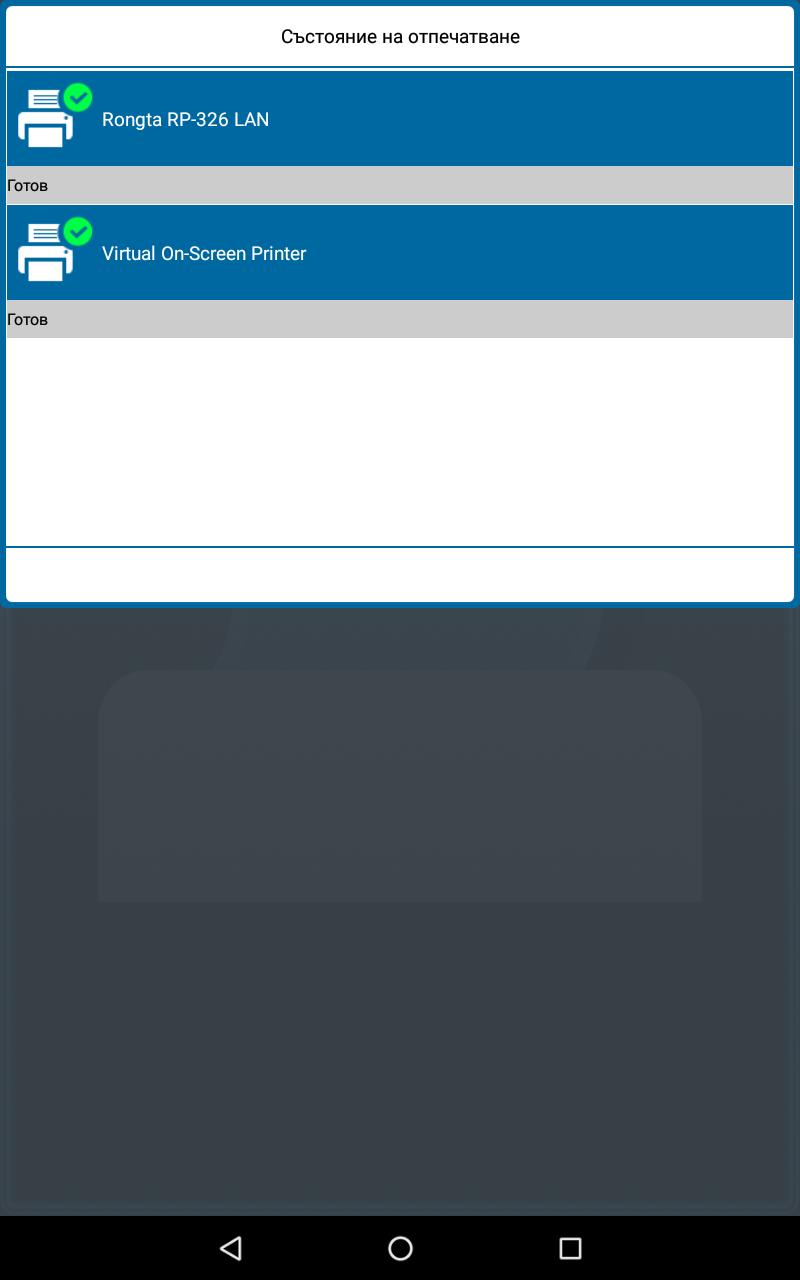


Иконката горе вдясно, когато се добави принтер, по презумпция показва, при натискане, излизайки плъзгащ се панел, че принтера е готов, независимо дали е свързан или не свързан, но когато се опитаме да изпратим заявка и той не е свързан, или не работи поради някаква причина(няма хартия, мастило и т.н.), то ще извести търговеца, че има някаква грешка. При добавени принтери, ще се покажат всички, че са готови, но отново, ако при някой възникне проблем , то там ще ни извести.

За да излезем от този панел, натискаме избелената част под него.

При натискане на някоя нотификация, свързана със статуса на принтера, се показва текст:

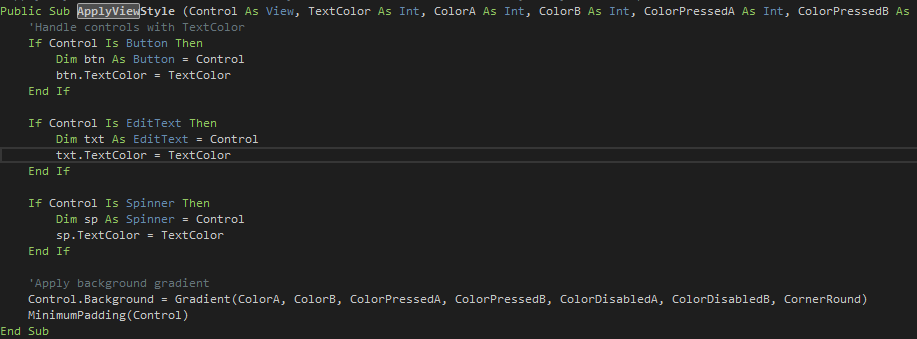
Фигура 3.19



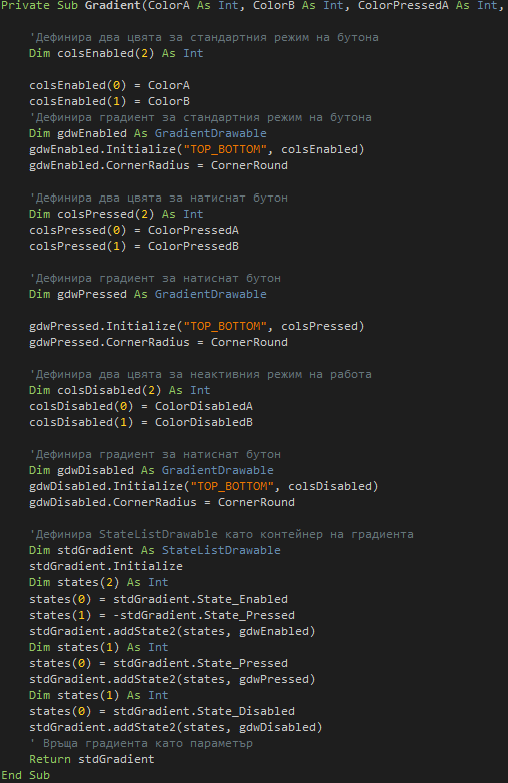
Взето е предвид, че ако стане така че са повече принтери, с обща височина по-голяма от тази на виждащия се панел, в който се намират, всъщност да са в плъзгащ се панел – позволява да се движи нагоре и надолу екрана, така че да се разгледат всички устройства. Същата превантивна мярка е взета и за скриптовете при добавянето и редактирането на принтерите.

Има модул, в който независимо какъв вид е контролера(падащо меню, бутон, поле за текст), на който искаме да зададем даден стил – цвят по подразбиране(могат да се прелеят два цвята/градиент), цвят когато е натиснат и др., го прави:

Фигура 3.20



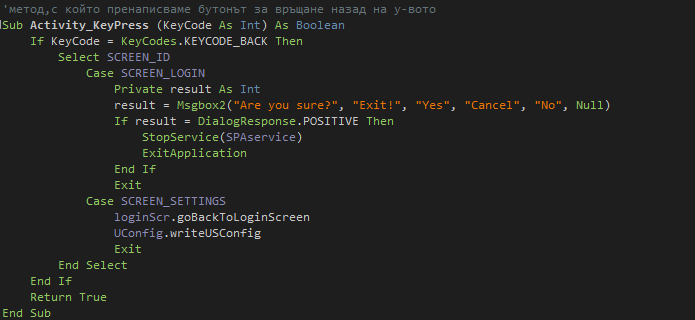
Фигура 3.21



Просто трябва да извикаме „ApplyViewStyle“ с цветовете, които изберем и коефициента на закръгляне на ръбовете на дадения обект.

Имплементирана е функция, която казва това какво да е поведението при натискането на бутона за назад:

Фигура 3.22



Ако сме във втория панел, то натискането на бутона назад ще ни върне към първия, а ако сме в първия – ще ни попита дали искаме да излезем от апликацията и при позитивен отговор ще я спрем.

# ЧЕТВЪРТА ГЛАВА

# РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ

Android service-a “Smart Printer for Android” може да се използва във всеки търговски обект, независимо какво предлага той – услуга или продукт. Приложението не е публично достъпно през интернет, а трябва да се закупи. То се изпраща по e-mail, който ще съдържа инсталационния файл (.apk). Съответният потребител(търговец) е нужно да изтегли файла и да го инсталира на своето устройство.

Има възможност и за извикване на квалифициран екип за инсталиране на уредбата и обясняване поведението на програмата.

След инсталация трябва да намерите тази икона:

Фигура 4.1



При стартиране на приложението се показва първият екран:

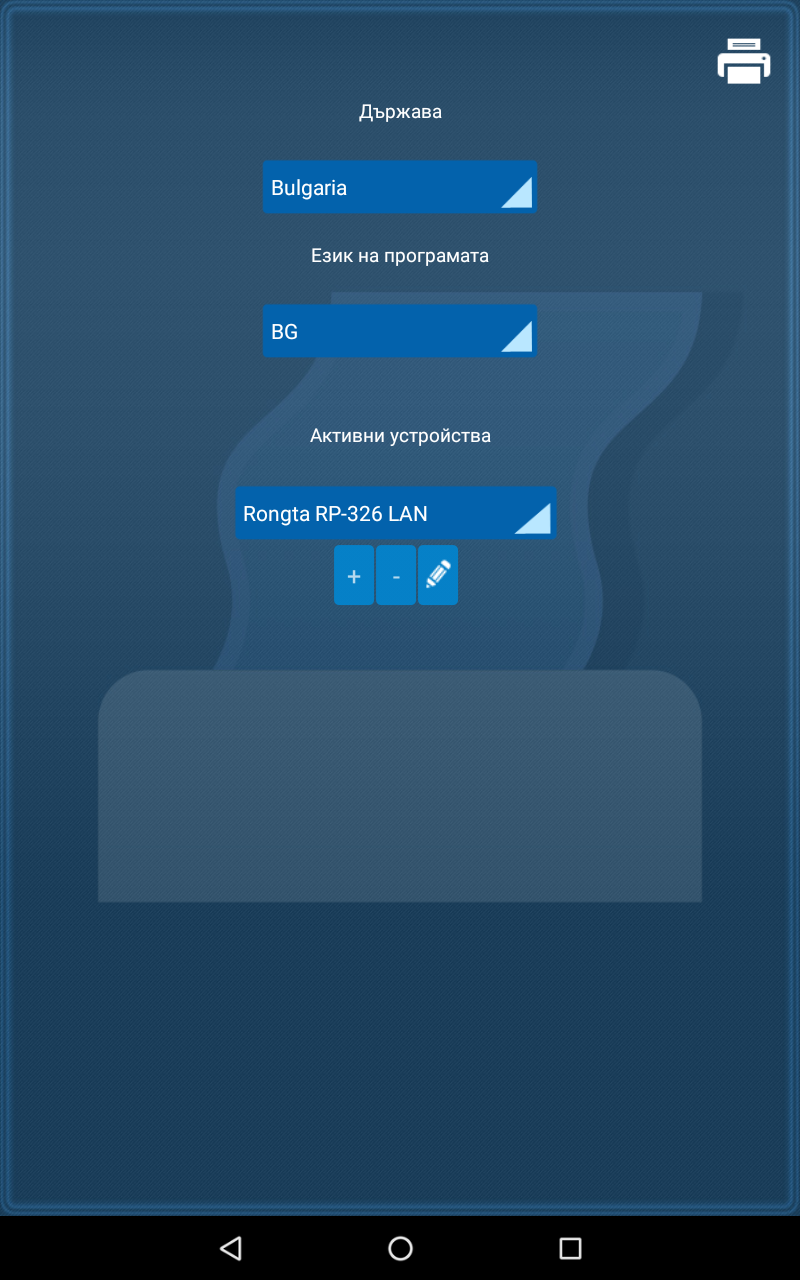
Фигура 4.2



В него се сигнализира дали сте свързани към локална мрежа, показвайки IP адреса на устройството и порта. При натискане отдолу на открояващия се етикет, се актуализира информацията за свързаността.

При преминаване към настройките се появява този екран:

Фигура 4.3



В него се избира държавата, в която работи касовия апарат, поради изисквания. Отдолу се избира езикът, на който искаме да се преведе приложението. Горе вдясно има бутон, който като се натисне излиза плъзгащ се панел, в който пише статуса на принтера, но все още този панел ще е празен, защото не сме избрали принтер.

За да изберем принтер, под етикета „Активни устройства“ трябва да цъкнем плюса:

Фигура 4.4

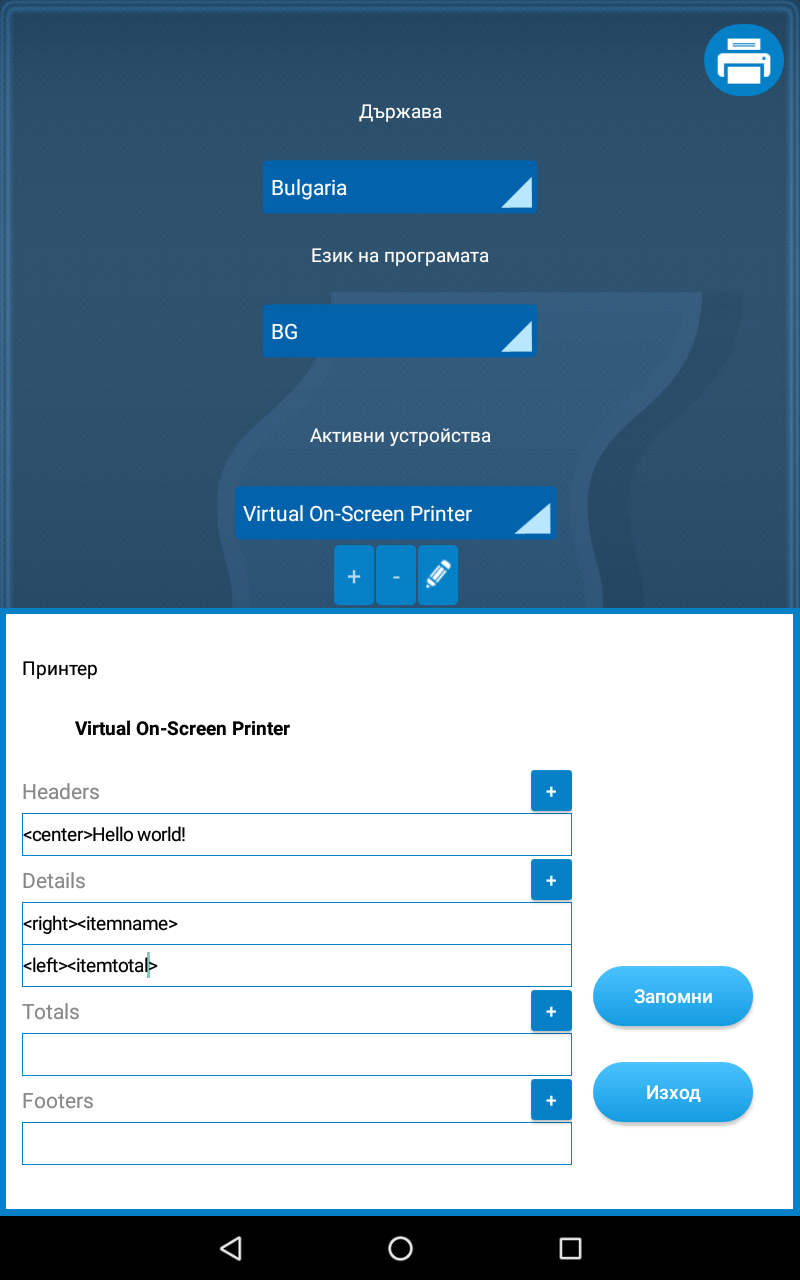


Появява се плъзгащ се панел и оттук можем да изберем принтер, с който искаме да работим, показва се неговия “Baud Rate” за Serial printer, за LAN са порт и IP, за Bluetooth е MAC address, и можем да зададем собственоръчно какво и къде да се напечата на бележката. Освен че можем да напишем просто текст и да го разположим някъде, можем да управляваме и атрибутите, които ще се вземат от получената заявка и да ги разпределим, например “<left><totalfull>”, ”<right><change>”,”<center><totaldiscount>”. Трябва да се има предвид, че тази част на приложението, не е препоръчително да се пипа от потребителя, всичко направено там е на негова отговорност! За персонализиране на касова бележка, може да бъде извикан квалифициран екип, който да конфигурира приложението спрямо нуждите на търговеца.

Ако искаме да редактираме или изтрием даден принтер, трябва да го изберем от падащото меню с активните принтери.

След това да натиснем моливчето, за да редактираме или съответно минуса – да го изтрием. При редактирането на принтер, можем да променим характеристиките, нужни за предаване на информацията(baud rate/ IP port, IP address) и неговите скриптове –да ги редактираме, изтрием или да добавим още редове.

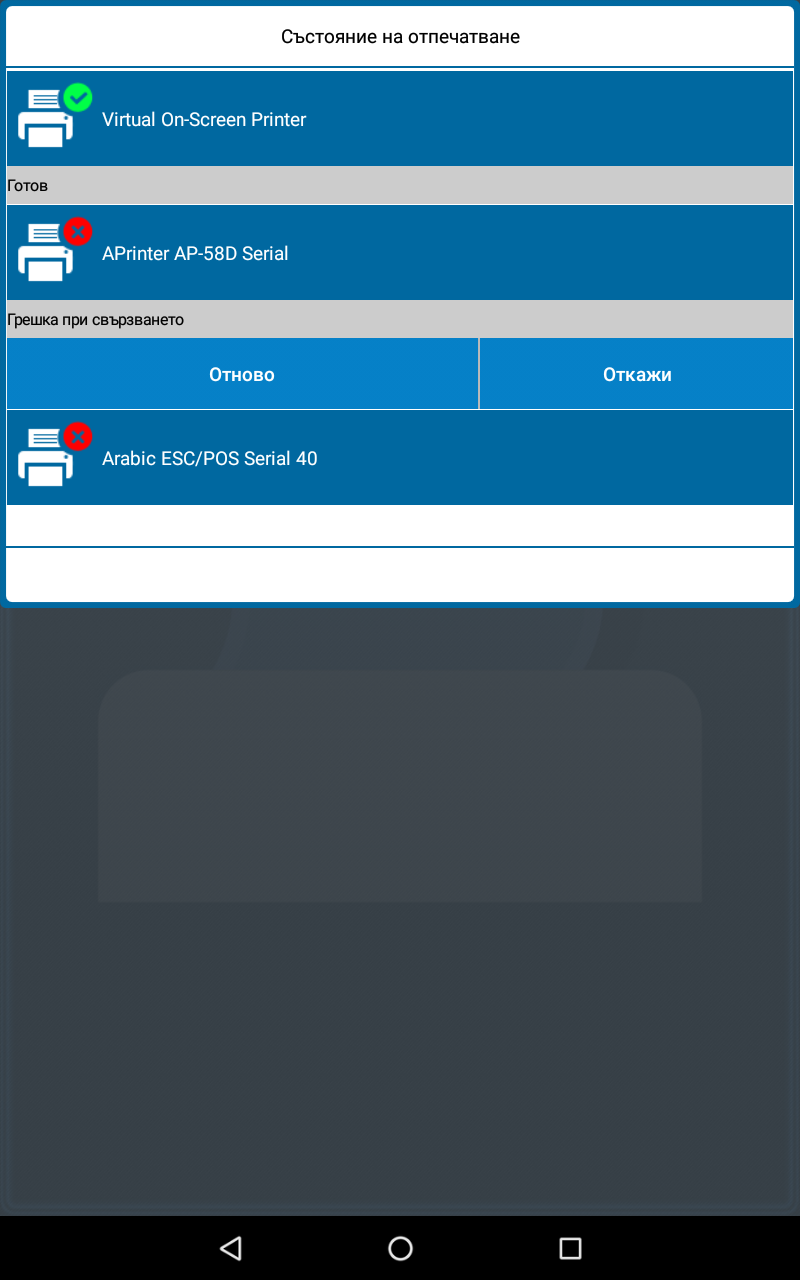
Фигура 4.5



Има и бутон(горе вдясно) известяващ за статуса на принтера. По подразбиране принтера, дори да не е свързан, винаги ще изписва, че е готов.

Но когато се случи някаква грешка след опит за изпращане на заявка за принтиране, от вида на това, че няма връзка или принтера не е успял да усвои и обработи информацията, то ще излезе грешка.

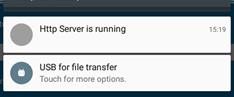
Фигура 4.6



Съответно ако има грешка ни известява и излизат два бутона - бутон за отказване на заявката, връщаща статуса в ‘готов‘ и един бутон за повторен опит за свързване и изпращане на получената заявка.

Когато приложението бъде изключено, service-a няма да спре да работи. Това е направено за да не се налага всеки път, когато случайно го изключите да трябва да го пускате отново, има и уведомление:

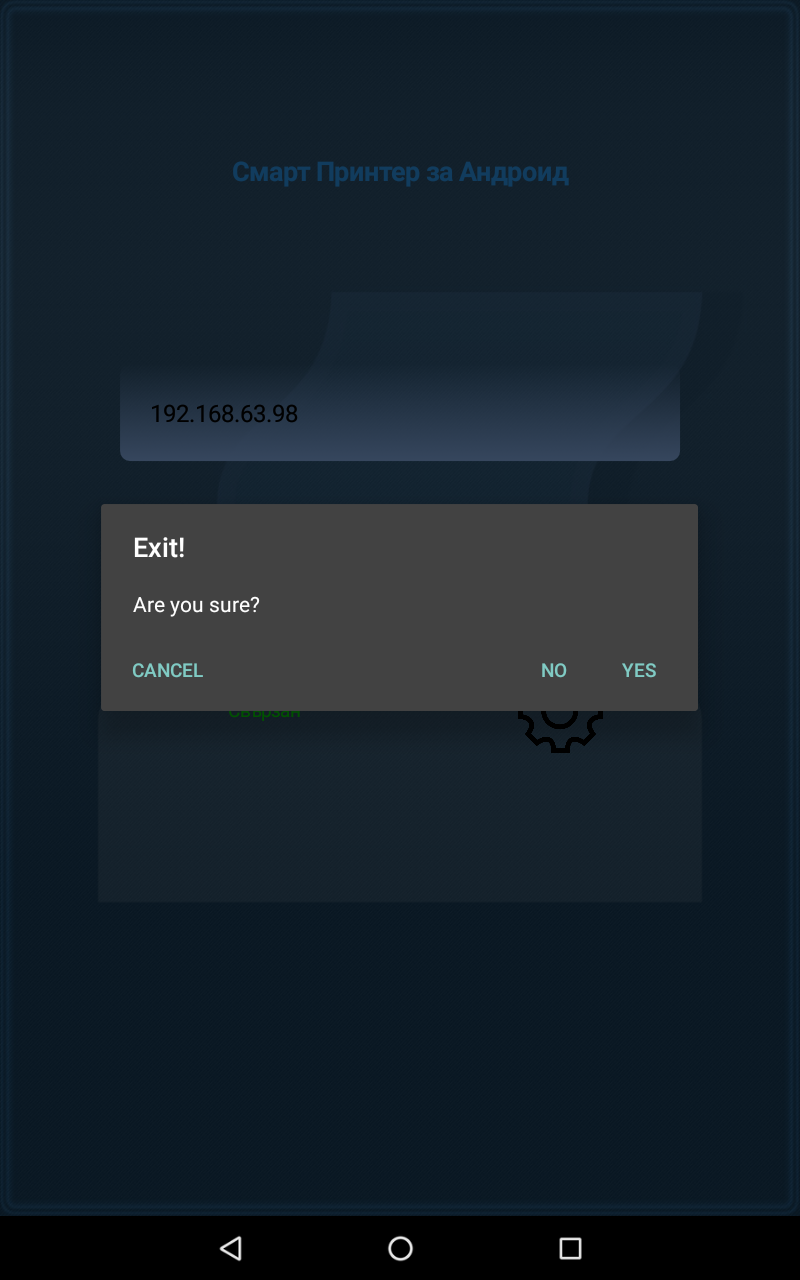
Фигура 4.7



Когато сме оставили апликацията във фонов режим или я изключим от скорошните приложения, service-a ще пусне отново Activity-то при получаване на заявка и ще изпрати данните за принтиране.

За да спрем приложението изцяло или трябва да спрем го от настройките принудително, или трябва да се влезе в приложението и от началния екран да цъкнем бутона за връщане назад и да потвърдим, че искаме да излезем- така спираме service-а, работещ на фонов режим.

Фигура 4.8



# Заключение

В заключение можем да кажем, че “Smart Printer for Android” задоволява нуждите на обекти от квартални магазинчета до големи вериги магазини с удобството и мобилността си, както и разширените функционалности за използване на повече от един принтер и манипулиране на бележката от самите касиери. За момента е само тест версия. Но управителите на различни магазини проявяват все повече интерес към този продукт.

Все още има какво да се изчисти и подобри в това приложение, така че разработката му продължава. Тестовете се правят с помощта на най-голямата българска софтуерна фирма за счетоводен софтуер – „Микроинвест“. Те с удоволствие се съгласиха да помогнат с бъдещето развитие на проекта, защото според тях тази система ще подобри начина на работа в търговските обекти, и то не само в България. Имат партньори от приблизително 16 страни, като повече от половината от тях вече проявиха интерес към приложението.

# Източници

<https://www.b4x.com/android/forum/>

<http://square.github.io/okhttp/>

<https://cdn.testequity.com/documents/pdf/toolsandsupplies/BRA-BP-PR600-Installation.pdf>

# Съдържание

[Увод 3](#_Toc2029810)

[ПЪРВА ГЛАВА 5](#_Toc2029811)

[Обзор за съществуващите решения 5](#_Toc2029812)

[**1.1 Подобни съществуващи решения** 5](#_Toc2029813)

[ВТОРА ГЛАВА 6](#_Toc2029814)

[ПРОЕКТИРАНЕ НА СТРУКТУРАТА 6](#_Toc2029815)

[**2.1. Функционални изисквания** 6](#_Toc2029816)

[**2.2. Съображения за избор на програмни средства и развойната среда** 10](#_Toc2029817)

[**2.3. Преглед на част от принтерите, свързващи се с приложението и работата с тях** 16](#_Toc2029818)

[**2.4. Принцип на софтуерния модел** 22](#_Toc2029819)

[**2.5. Преглед на библиотеките използвани за Android приложението** 23](#_Toc2029820)

[**2.6. Получени резултати** 30](#_Toc2029821)

[ТРЕТА ГЛАВА 32](#_Toc2029822)

[ПРОГРАМНА РЕАЛИЗАЦИЯ 32](#_Toc2029823)

[ЧЕТВЪРТА ГЛАВА 56](#_Toc2029824)

[РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ 56](#_Toc2029825)

[Заключение 65](#_Toc2029826)

[Източници 66](#_Toc2029827)

[Съдържание 66](#_Toc2029828)