**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, се използва ЕКАФП (Електронен Касов Апарат с Фискална Памет) или (не)фискален принтер, свързан с компютър. За целите на този документ ще обобщим понятието „търговски обект“, както и разликите между ЕКАФП и (не)фискален принтер.

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

Съществената разликата между (не)фискалния принтер и ЕКАФП се състои в това, че касовият апарат функционира самостоятелно, докато принтерът не функционира самостоятелно, а му е необходима връзка с компютър, както и специализиран търговски софтуер – за ресторанти или склад-магазин. Работата с подобен софтуер има предимства: поддръжката на складовите наличности, сигнализиране за изчерпващи се стоки, улесняване процеса на ревизия.

Нека разгледаме примерна постановка на търговска система предназначена за заведение (бар/ресторант), състояща се от: Компютър, монитор, периферни устройства, фискален принтер, каса.

Какво би се случило ако например, елемент от системата, спре да работи? Или процеса по инсталиране на системата е силно затруднен, а може би дори невъзможен, поради физическото местоположение на изграждащите го компоненти? Премахването на касовия апарат, за осъществяването на инсталацията е непрактично в реални условия – би възпрепятствало следенето на складови наличности, а в нашата дигитална ера връщането на клиентите във времето на хартиените носители не е опция. Още повече, ако конкретна система не доставя конкурентните си предимства в стабилна среда, какво би задържало клиентите й? Парадоксалните ситуации, предизвикани от не добре проектирана архитектура на системата, софтуера или компонентите са разнообразни. В ситуация със заведение – и клиенти, издирващи из менюто с ценоразпис поръчките си, или дори болнични заведения, не успяващи да проследят лекарствени наличности, подобни проблеми са не само с икономически последствия, а понякога и по-сериозни. Подобни ситуации са причината да се захвана с решението на проблема, защото освен за самите клиенти, подобно решение, ще 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 техническа литература и 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. Още повече, при наличието на уредбата, управлението през малко по размер устройство, като телефон би било много по-лесно, по-практично и не на последно място - по-ефективно от финансова гледна точка, в сравнение с използването на компютър.

# ВТОРА ГЛАВА

# ПРОЕКТИРАНЕ НА СТРУКТУРАТА

## **2.1. Функционални изисквания**

Smart Printer for Android няма да е изцяло автоматизиран продукт, който да работи самостоятелно. То ще очаква да получава заявка с текста, който да бъде принтиран от принтер.

Smart Printer е Android приложение, което е съставено от много на брой драйвери за различни принтери. В драйвери са дефинирани характеристиките на самите принтери (например 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.

Получавайки заявка от настроен търговски софтуер, пригоден за Smart Printer for Android, или Postman, или лентата в браузъра, софтуера започва да я раздробява(посредством т.нар. Parser – набор от код, който декодира информацията), обработва, а в последствие и я изпраща към принтера.

Приложението се дели на две основни части:

* Първи панел

Първият панел има етикет, показващ името на приложението, а непосредствено под него - две полета за въвеждане на текст. В едното поле, при наличие на връзка с локална мрежа се показва IP-то, а в другото поле се дефинира порта, през който се очаква заявката, съдържаща XML-а. Под двете полета е позициониран още един етикет, който индикира, съответно със зелен цвят “Connected”, когато приложението е свързано към локална мрежа и съответно с червен цвят - „Not connected“, ако приложението не е свързано към мрежа. Под гореизброените елементи, е дефиниран бутон, водещ към втория панел.

* Втори панел

Вторият панел е съставен от три падащи менюта, едно от които дава достъп на потребителя до избор за държавата, в която ще се използва принтера. Следващото падащо меню, позволява да бъде избран езикът, на който ще бъде визуализирано приложението.

Чрез третото падащо меню, ще се избира активния принтер, този с който си взаимодейства потребителя, същото меню ще се използва и за добавяне на принтери. Потребителя има възможност да работи с повече от един принтер едновременно. При добавянето на принтерите, би могъл да дефинира скриптове, форматиращи и оформящи принтирания от принтера текст:

* + - Headers – текст, който ще е принтиран в горната част на бележката;
    - Details – текста на бележката, ситуиран в средната й част;
    - Totals – частта, в която се визуализират общата сума на продажбата, както рестото и платената сума. Това поле се намира между полетата Details и Footers;
    - Footers – най-долната част на касовата бележка;

По подразбиране, приложението се използва с първоначален шаблон, който управлява данните, които са изпратени към принтера и разпределя в коя част от бележката и в какъв вид, попада информацията. Ако клиентът има изискване към подредбата на текста в бележката, то има опция за осъществяването му чрез гореспоменатите скриптове.

При добавяне на скрипт, шаблона по подразбиране, в дадената секция се изтрива и се заменя с нововъведения от потребителя скрипт. Предвиден е по един бутон за всеки вид скрипт, който добавя по още едно поле за въвеждане, т.е. всеки вид скрипт може да има по повече от един ред/команда – според нуждите на използващия го.

Възможност за изтриване и редактиране на добавените принтери дават частите от приложението, намиращи се непосредствено под самото падащо меню, визуализиращо активните принтери. Дефинирани са още два бутона – този за редактиране, обозначен с пиктограмен символ – молив, както и бутон за изтриване на избрания принтер, обозначен със символ минус.

При натискане на бутона за редактиране, странично се появява панел с плавна анимация, в него потребителя има възможност да редактира скриптовете, зададени при добавяне на устройството и да калибрира настройките на самото устройство.

Тези скриптове позволяват настройки на принтирането на бележката, като страна в която да се показва името на продукта, местоположение на общата дължима сума и др.

При натискане на бутона в горния десен ъгъл, с пиктограмен символ принтер, отново се появява страничен панел, в същия стил и анимация, съдържащ в себе си списък с активните принтери, както и актуална информация за състоянието им (дали са в изправност и готовност за печатане).

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

Smart Printer for Android трябва да е лесен за използване продукт, с иновативен и разбираем от всеки потребител интерфейс, без значение от възрастовата му група. При проектирането му трябва да се намери баланс между функционалност и простота на интерфейсната среда, без да се прави компромис със стабилността на приложението. Философията на дизайна е семпли форми и контрол в комбинация с модерен дизайн (актуалния в момента материален дизайн, препоръчван от Google). Поради тази причина избрах да използвам един много функционален език за създаване на Android приложения, а именно Basic4Android (B4A).

Сред основните предимствата на езика са:

* 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. Кривата на учене не е стръмна (основен проблем при обучение към Android Studio) – IDE-то и framework-а подпомагат пишещия с подсказки, довършване на изрази, предупреждения за логически или синтактични несъответствия.
* 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 не се различават от APK файловете, създадени (компилирани) с Java / Eclipse / Android Studio;
* Възможност за употреба на суров Java код, чрез обгръщане, осъществено с библиотека и монолитното му свързване към средата за разработка на B4A приложения;
* Представянето/производителността са много близки до тези на приложение, създадено с Android Studio Java, като практически няма систематизирани изследвания, доказващи разлика между приложенията написани на двата езика/среди;
* Бърз debugger, поддържащ светкавичен 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 (стъпка по стъпка, се преминава през всеки ред код, позволявайки на разработчика да проследи стойността в даден израз/променлива, както и връзките между модули и изрази);
* Изключително гъвкав език – единственото му ограничение е набора от библиотеки, които впрочем, поддържат 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.
* Възможност за паралелно използване на кода (back-end частта) в алтернативите на B4A - B4J (Basic for Java – алтернатива за компютър (уиндоус) или в B4I (Basic of IOS). Около 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 (характерно предизвикателство към дизайна на Android приложенията). Компилираните програми могат да бъдат тествани в емулатори на AVD Manager или на истински Android устройства, използващи Android Debug Bridge и B4A Bridge (приложение за Android позволяващо осъществяване на безжична връзка между ID и устройството).

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, приложение, 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 позволява да се използва 30 дни, но не с всички функционалности. Може да се закупи стандартна версия, която включва поддръжка 2 месеца или Enterprise версия включваща актуализиране на софтуера за 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 10 метра разстояние.

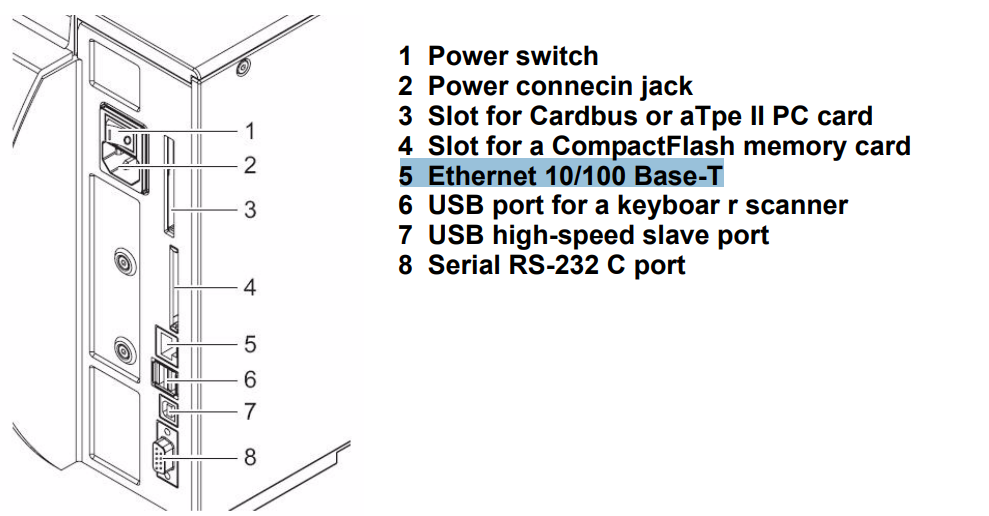
**Как да свържем принтера към устройство:**

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 връзката се осъществява чрез Ethernet cable към мрежата, а през нея информацията с инструкции се изпращат от Android Service-a, подадени преди това от софтуер или проста заявка, в XML структура. Стойностите по подразбиране(IP адрес и порт) се показват, зададени от драйвера и ако бъдат променени в принтера, трябва да бъдат изменени и в приложението.

* Rongta RP326 COM

Фигура 2.3.4



**Изисквания за Serial принтерите:**

Този принтер позволява връзка и чрез мрежа, и чрез сериен порт (Serial Port), но в случая ще послужи, като пример за свързване на принтера към мобилното устройство (Android Service-a) чрез сериен порт.

Свързва се с телефон (андроид устройство) чрез кабел (необходим е преходник от Serial Port към USB в допълнение с 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, например baud rate – скорост на информацията, която се предава (bit per second). Съществува възможност от приложението да се променят характеристики на принтера, но за тази цел принтера, трябва да е настроен на тази ‚скорост‘, 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 ‘комуникират’ помежду си. Например, при връзка между телефон четци на SD карти, камери, мишка или клавиатура и други 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 между устройството и мрежата. След натискане на бутона за настройки, приложението преминава в следващия панел, в който се избира език на приложението, както и страна, в която се използва то. При работа с принтерите, настройките позволяват управление на скриптовете им, редактирането им, изтриването им при нужда, както и пълен набор от функции за управлението на принтерите. Наличен е и екран показващ статуса на принтера – свързаност към мрежа и статус на заявката.

Фигура 2.4.1

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

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

* JavaObject

JavaObject библиотеката прилича на библиотеката за Reflection. И двете позволяват ‚комуникация‘ директно с 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, но 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. Декларира се обект JavaObject, който след това "обвива" всеки друг 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 позволява да последователно осъществяване на достъп до файловете и байтовите масиви. Може да се използва и за кодиране на номера до байтове (и обратно). Не трябва да се забравя, 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. Извършва това чрез т.нар. "Reflection", използващ 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 и обменят данни. Ефективния обмен на данни чрез HTTP, спомага за бързодействие на приложението и намаляване на трафика и използваните данни.

OkHttp е HTTP client, който е ефикасен по презумпция, защото:

* 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 обменените данни;
* Кеширането на отговорите избягва напълно мрежата за повторни заявки.

OkHttp свързаността, носи стабилност, тъй като при прекъсвания, породени от мрежови проблеми – тя по „невидим“ (Silent) за потребителя начин ще се самовъзстанови. Ако използвания service работи с множество IP адреси, OkHttp ще изпробва алтернативни адреси, в случай на неуспех в първото свързване.

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 ‘builders’ и ‘immutability’. Той поддържа както синхронни блокиращи повиквания, така и асинхронна комуникация с обратни повиквания.

HttpUtilis е базирана на Android версията на Apache HttpClient, Google спря поддръжката и актуализацията на този SDK преди много време и те го премахнаха от Android M. HttpUtils2 е базиран 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 (различията са в по-големия набор от поддържани 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 в сравнение с 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, 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 или че елемент от нея може да дефектира. Работи за трите вида свързване на принтерите, успешно заменя компютър, който макар качествата си изисква множество настройки, повече място, зависим е от кабели, и като цяло система с повече елементи, а от там повече елементи, които могат да дефектират. 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 се свързва с поддръжка от фирмата, от която е купена инсталацията и софтуера, както често се налага, иначе. Както и това, че ако реши да направи преаранжиране на магазина, няма да го има проблема с ограниченото място и достъп до кабели, необходими на компютър.

* 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, според желанията на клиента;
* Подобряване самочувствието на работещите в магазина поради иновативните технологии, използващи се от тях;
* Оставя впечатление у клиентите
* Минуси - засега не са установени, поради липса на системни изследвани.

# 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 в най-горната част на екрана, IP-адреса и порта, на които той „слуша“.

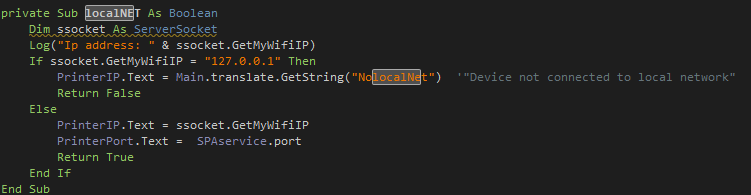
Под тях е ситуиран етикет, оказващ свързаността към локална мрежа. При натискането му, той опреснява информацията за връзката. До него е дефиниран бутон за преминаване към втория екран на приложението.

Фигура 3.1



Частта от кода, проверяваща свързаността на устройството:

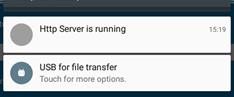
Фигура 3.2



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

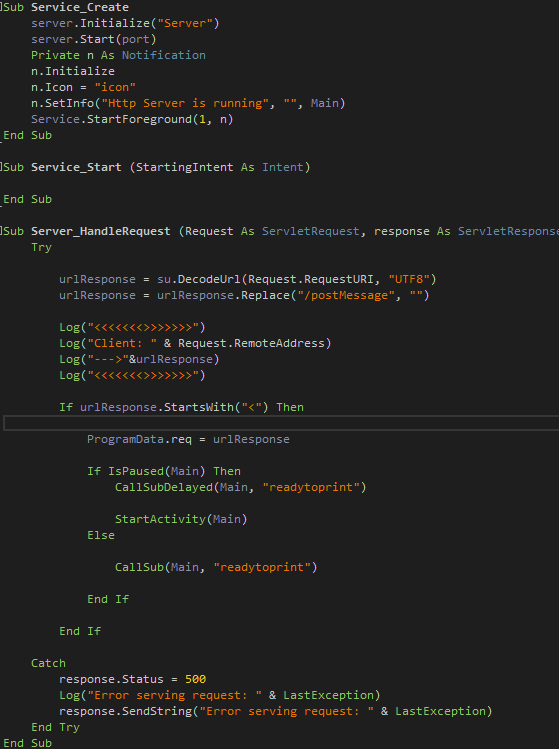
Взета е превантивна мярка – когато приложението се изчисти от полето с работещи апликации (търговецът случайно го изключи), то няма да спре Service-а, а напротив, ще е активно, докато не го изключим. Също така в центъра за уведомления ще се запази съобщение, че все още работи:

Фигура 3.3



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

Фигура 3.4



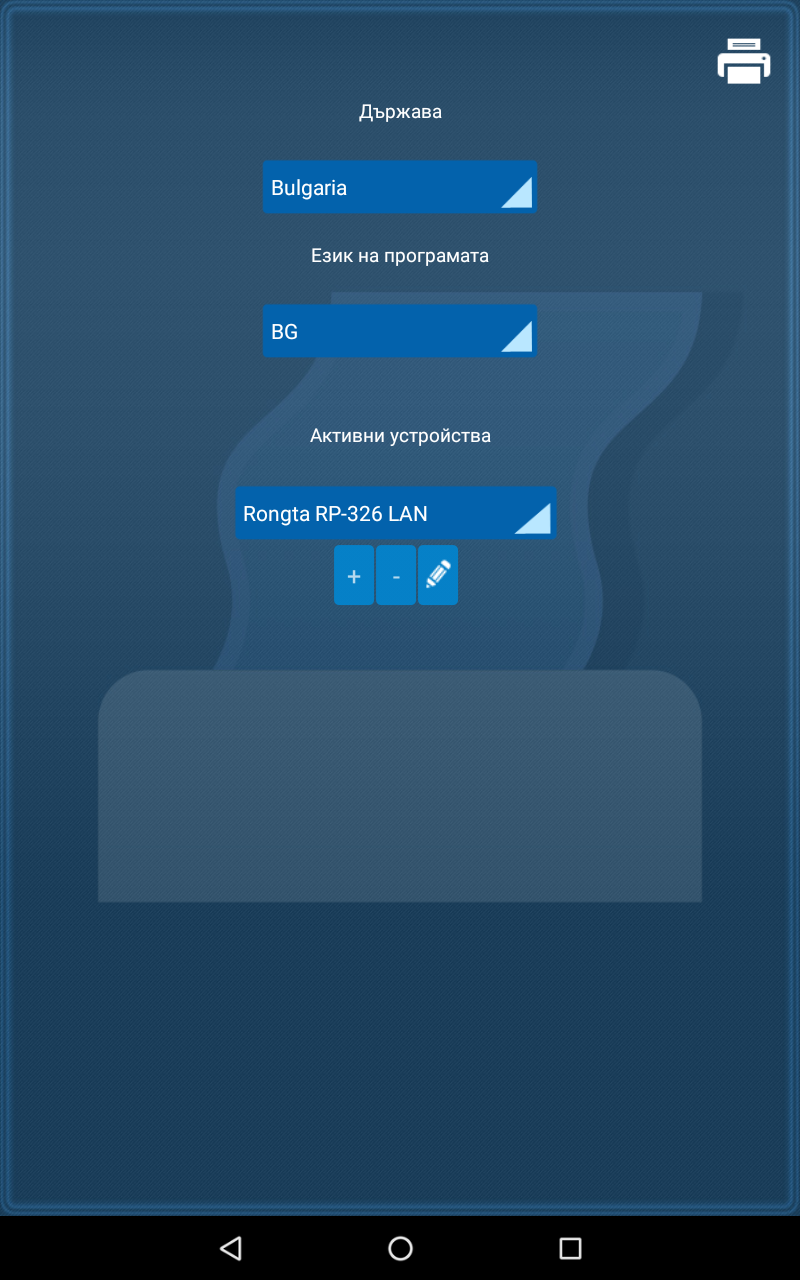
Проверката дали приетата, нужна част от заявката започва с „<“, е необходима, защото ако е активирано разширение (extension)(за блокиране на реклами например) в браузъра, то се изпраща заявка от типа “favicon.ico/” – бъг в браузъра.

След като се натисне бутонът, обозначен с общоприетия символ- пиктограма за настройки, се преминава 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, ще работи.

Фигура 3.5

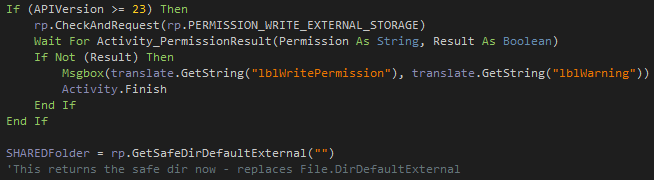


Когато се изберат тези настройки, биват запазени в RAF (random access file). Поведението на RAF е като на масив от байтове, държащи се в системата, те могат да бъдат и прочетени, и презаписани.

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

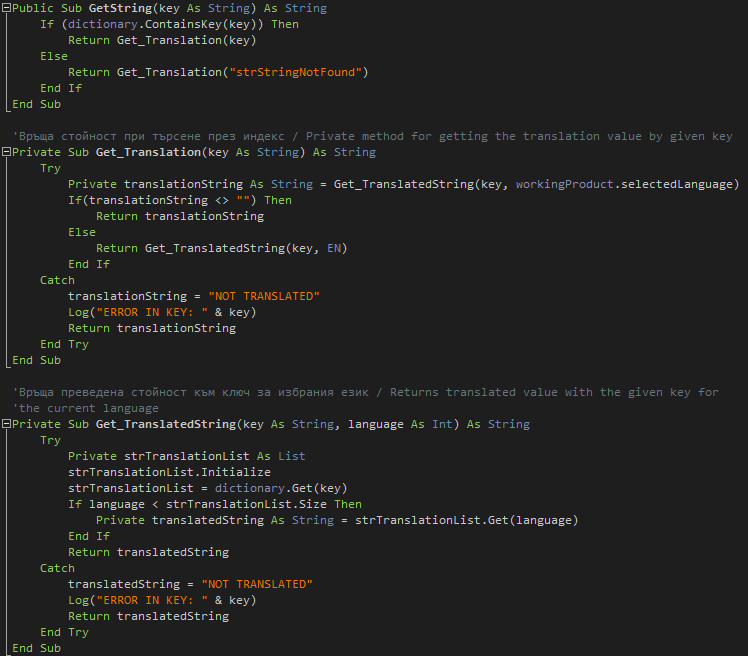
B4A v6.0 добавя поддръжка за разрешения за изпълнение. Удобството им се заключва в това, че потребителят няма да бъде попитан за него, когато инсталира приложението от Google Play, а вместо това те ще бъдат помолени да одобрят "опасни" разрешения по време на изпълнение. А записването в RAF е такова.

Фигура 3.6



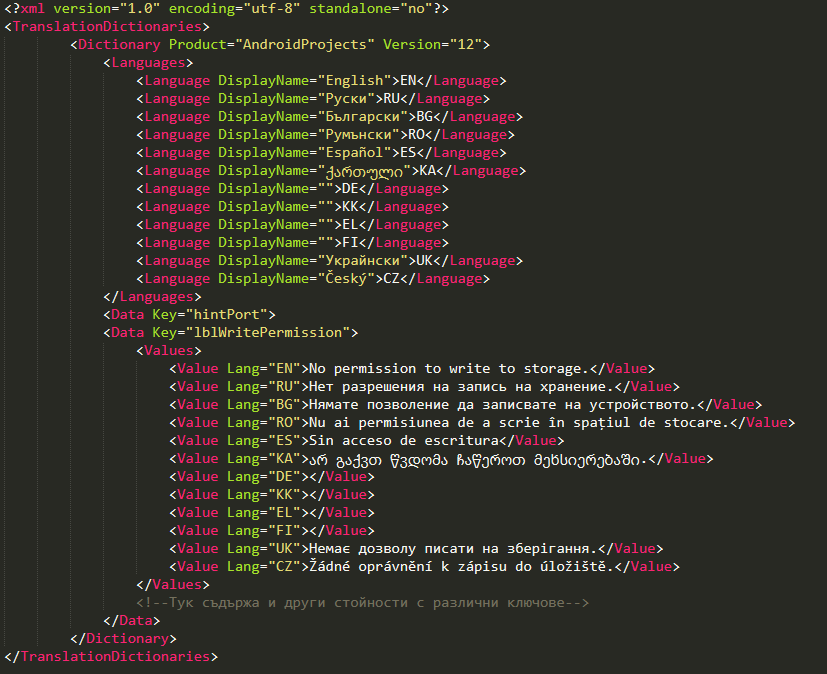
При избиране на език всеки текст от приложението се превежда, както и касовите бележки, които ще се издават, като дадения израз се извлича от map, съдържащ раздробен XML файл чрез Parser.

Фигура 3.7



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

Фигура 3.8



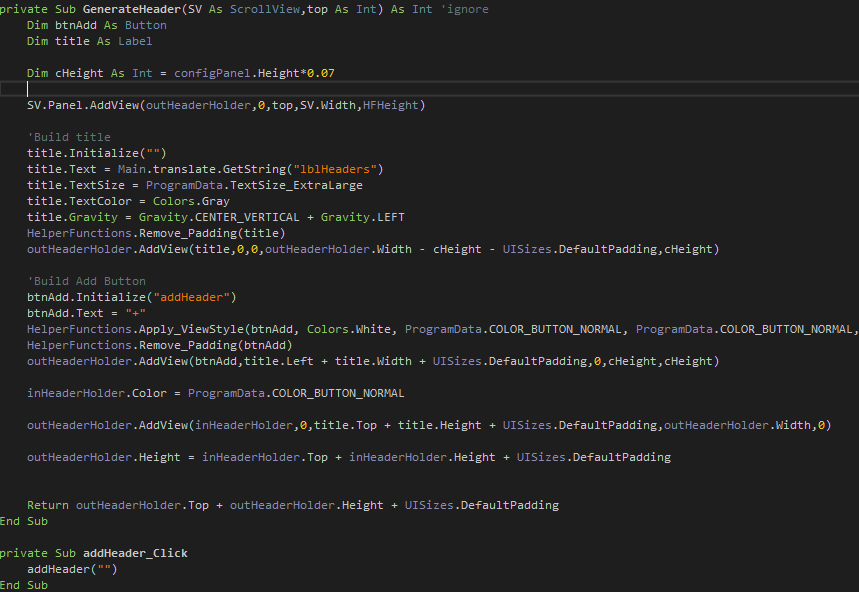
Налично е още едно падащо меню, което позволява възможността да се визуализират активните принтери и да бъдат избирани, тези, които потребителят желае да редактира/ или изтрие (чрез тези бутони под него – с моливчето за редактиране, с минуса – за изтриване на принтера от списъка с принтери. Бутонът със символ „плюс“ се използва за добавяне на принтер.

Когато се натисне бутона за добавяне на принтер, се появява страничен панел, визуализиран, чрез плъзгаща се анимация. Панелът съдържа информация за всички поддържани принтери от приложението. Потребителя добавя принтер, с който ще работи, избирайки го от списъка и извършвайки първоначална настройка. При нужда за промяна в стила на печатане, потребителя има възможност да измени шаблона по подразбиране от съответните полета от скриптовата част.

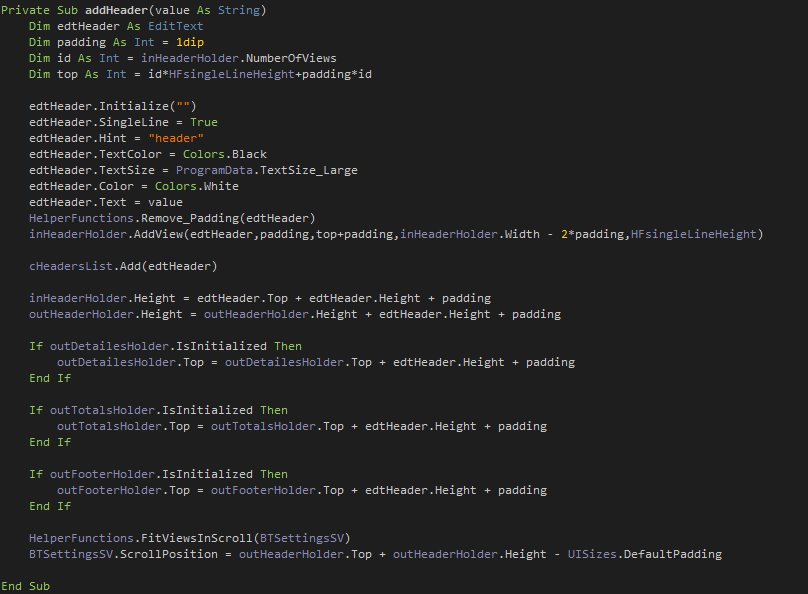
‚Скриптовете‘ са разделени на четири части – Headers, Details, Totals и Footers – точно както се разделя една касова бележка. Може да се добави по повече от един ‘script’ от всеки вид.

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

Фигура 3.9

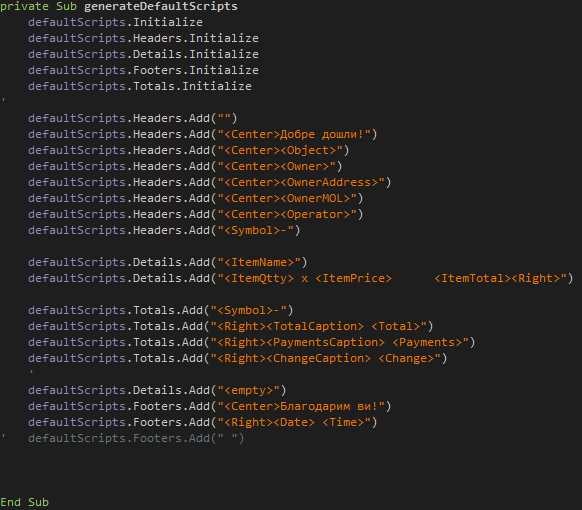


Фигура 3.10



Реализиран е шаблон на касовата бележка по подразбиране:

Фигура 3.11



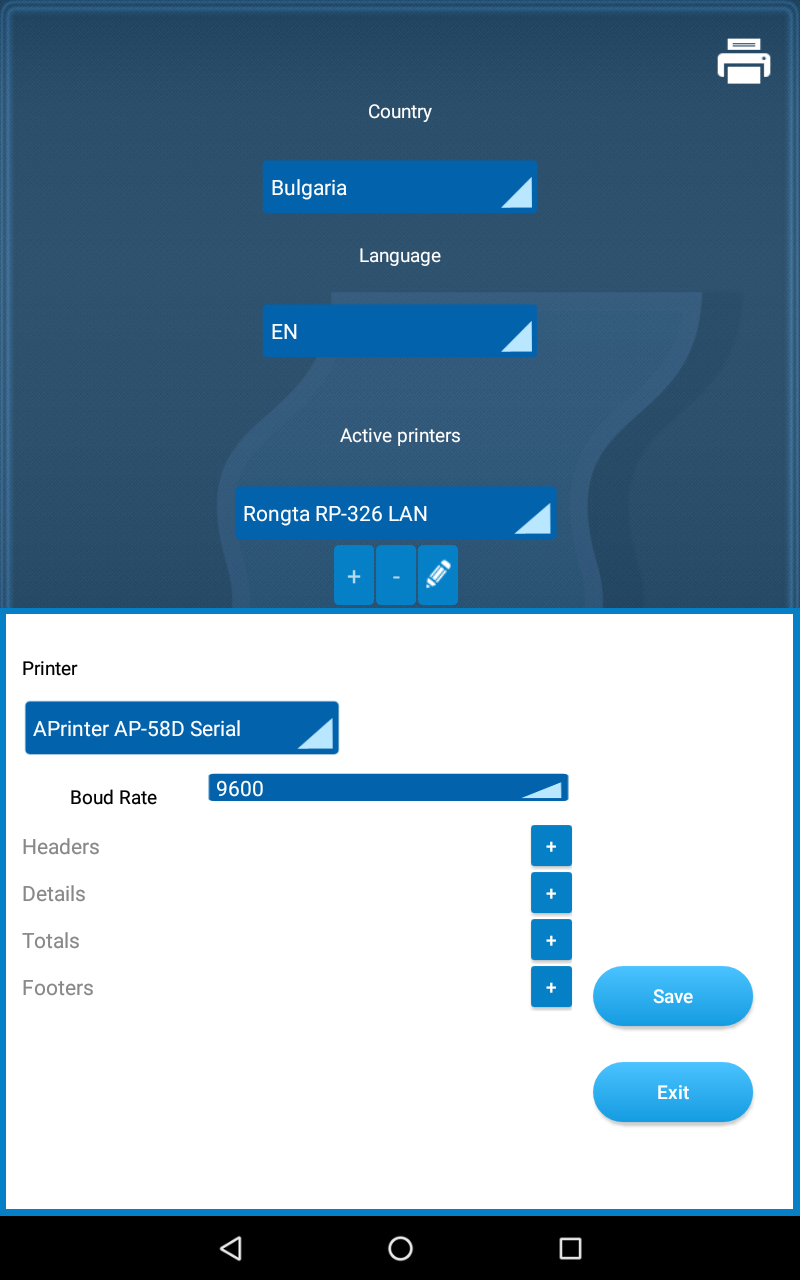
При промяната на даден участък от касовата бележка (Headers, Details, Totals, Footers) ще се изчисти и промени само той, останалите нямат да бъдат засегнати.

Визуализация на екрана за добавяне на принтер:

,

а

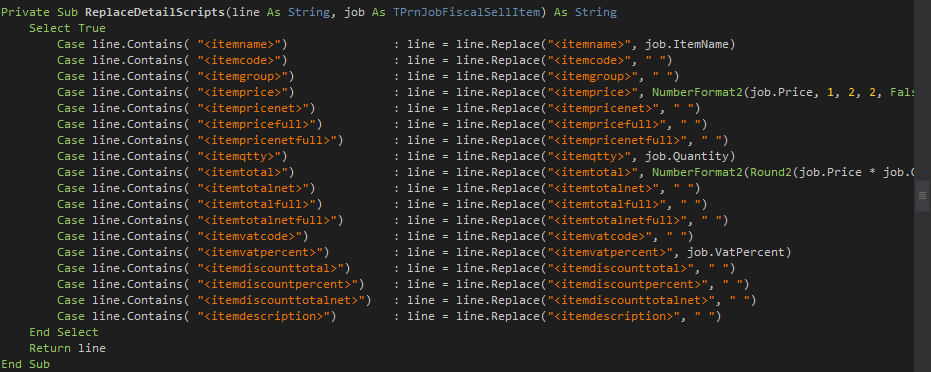
фигура 3.12



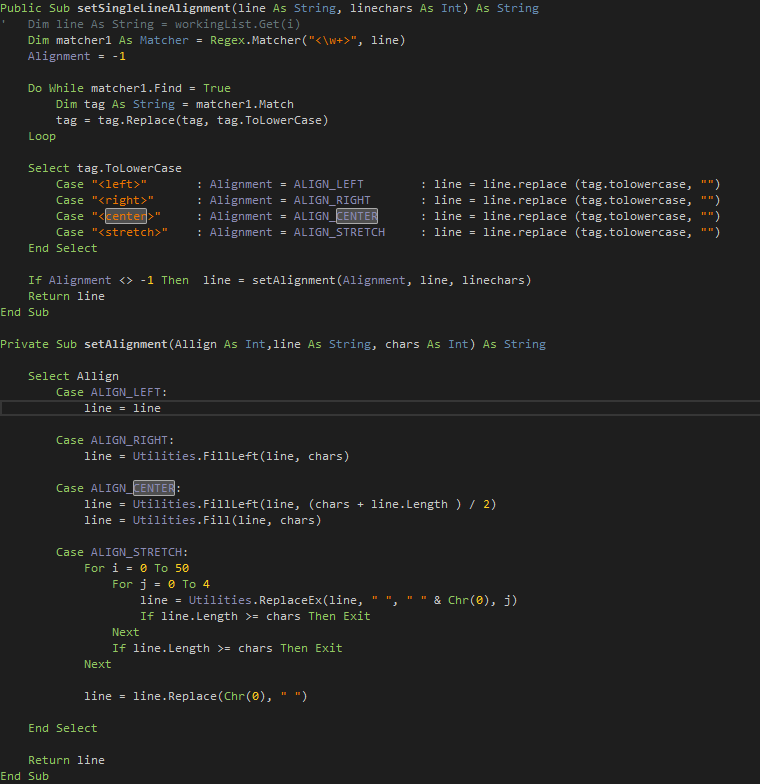
Всяко подразделение на касовата бележка ( Headers, details, totals, footers) има собствени команди, достъпващи елементите и имат и общи, които отговарят за това къде и как да бъдат разположени и изобразени те.

Пример за елементите в Details и за командите за форматиране:

Фигура 3.13

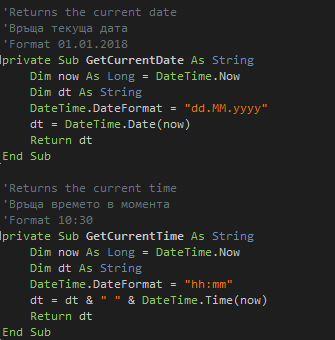


Фигура 3.14



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

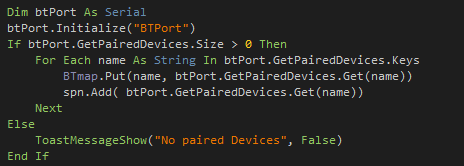
Фигура 3.15



Според модела, характеристиките и настройките на принтерите варират:

* При ‘Serial’ принтерите основна настройка е Baud rate. Baud rate показва скоростта на предаване в бит/секунда. Принтерът и комуникационната линия (COM port) трябва да са настроени с един и същи Baud rate. При смяна на настройката в принтера, има възможност и за смяна в самото приложение на обема информация, обменена за единица време.
* За ‘LAN’ принтерите настройките са техният IP address и IP port. Отново, при смяна в принтера, трябва да бъде променен IP адреса, на който устройството изпраща информацията, така че да съвпадат.
* За ‘Bluetooth’ принтерите характеристика е техният MAC (Media Access Control) адрес. Всяко Bluetooth устройство разполага със собствен MAC Address, който трябва да бъде извлечен посредством библиотеката Serial, с чиято помощ се извлича списък на всички устройства свързани към устройството с приложението, а те се визуализират с техните MAC адреси. Потребителят избира от този списък MAC адреса на принтера, с който ще работи, а той може да бъде открит в в документацията на самия принтер, или в случай, че такава не е налична – изтривайки всички устройства и свързвайки единствено търсения 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;
* Команда определяща, къде се отрязва 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 са създадени два бутона – „Запомни“, чрез който потребителя да запамети принтера с настройките, конфигурирани от него и „Изход“, за излизане от менюто без запаметяване на настройки. Може да бъде добавен повече от един принтер.

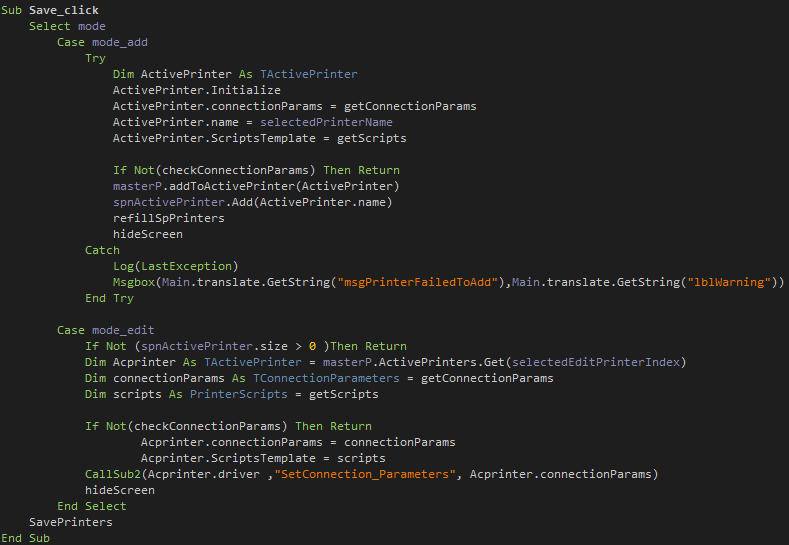
При натискане на бутона със символ молив се извиква екран за редактиране на избрания принтер от падащото меню с активните принтери, появяващ се отново чрез плъзгаща се анимация.

Показва се името на принтера, както и съответната характеристика спрямо вида принтер, полетата за скриптовете и добавените в тях настройки, както и два бутона за запаметяване на настройките и за изход.

Потребителя има възможност освен да редактира скриптовете при добавянето на принтера, да стори същото и на по-късен етап.

Когато бъде натиснат бутона “Save”, без значение дали се редактира или добавя принтер, информацията се запазва отново в RAF, информация за принтера и неговите особености:

Фигура 3.18

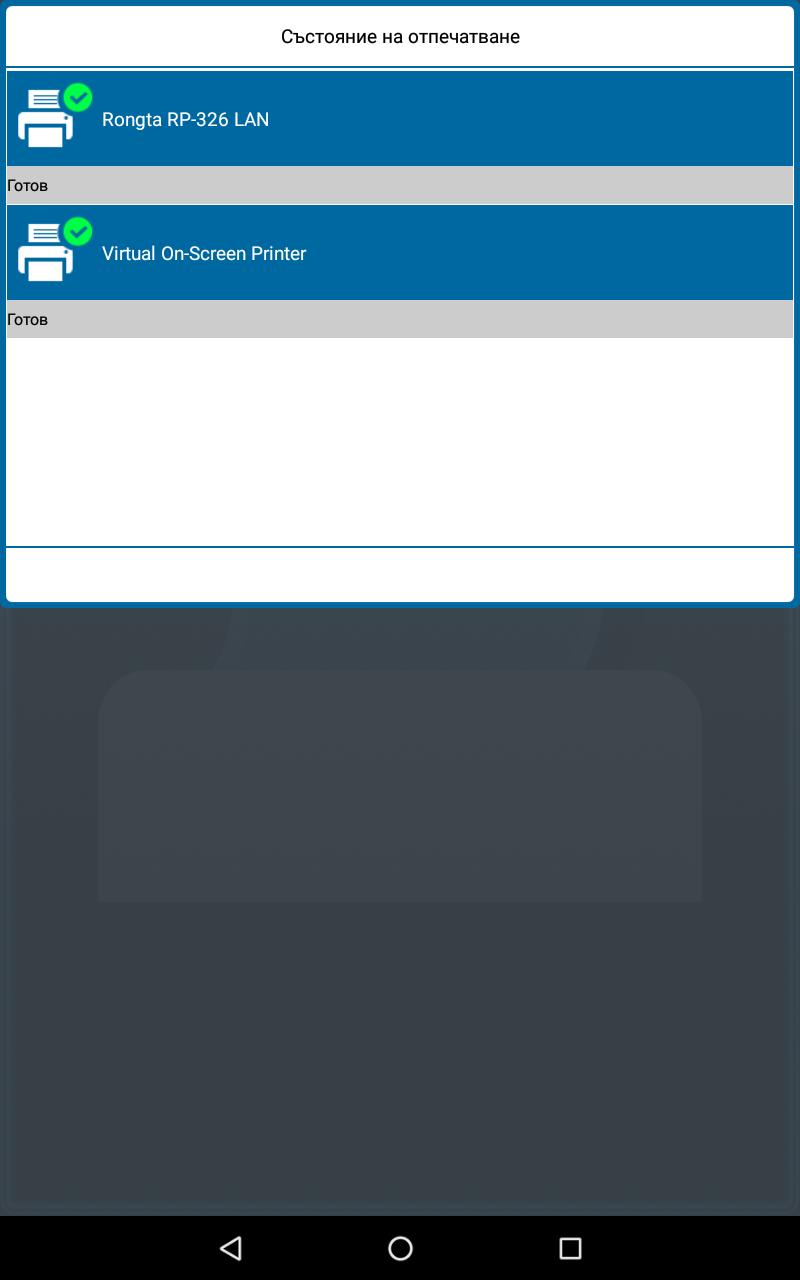


Иконката в горния десен ъгъл, по презумпция индикира, когато бъде добавен принтер, а при натискане, се появява плъзгащ се панел, показващ, че принтера е готов, независимо дали е свързан или не. При опит на потребителя да изпрати заявка към принтер, който не е свързан, или не работи поради някаква причина (липса на хартия, мастило и т.н.), панелът ще извести търговеца, че е възникнала грешка. При добавени принтери е налична индикация за готовността им, или възникнали грешки.

За изход от този панел, се натиска избелената под него част.

При натискане на някое известие, свързано със статуса на принтера, се показва обяснителен текст:

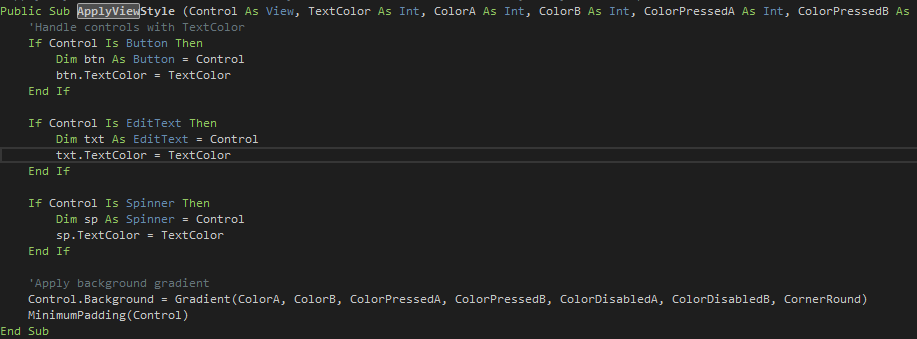
Фигура 3.19



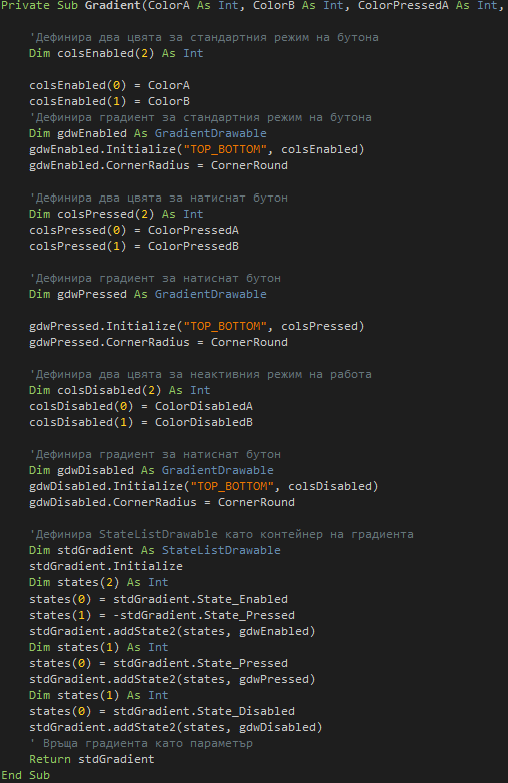
Взето е предвид, че при наличието на повече принтери, в случай, че не могат да бъдат показани всички, приложението позволява екрана да се движи нагоре и надолу (скроу), така че да се разгледат всички устройства. Същата превантивна мярка е взета и за скриптовете при добавянето и редактирането на принтерите.

В друга част на приложението, чрез модул, се задават стилове с цветове (могат да се прелеят два цвята/градиент), цвят когато е натиснат, независимо какъв вид е контролера(падащо меню, бутон, поле за текст):

Фигура 3.20



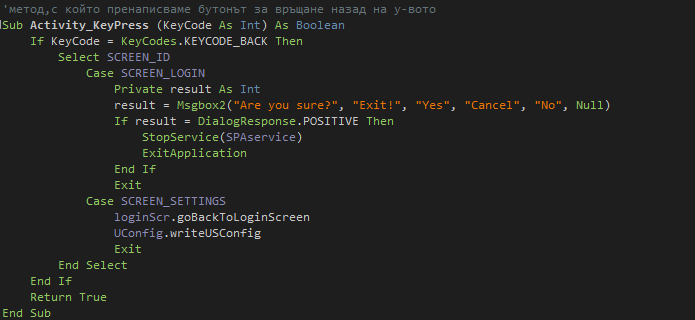
Фигура 3.21



Тези промени се осъществяват, чрез команда „ApplyViewStyle“ с цветовете, които се зададат, както и коефициент на закръгляне на ръбовете на дадения обект.

Имплементирана е функция, която определя какво да е поведението при натискането на бутона „назад“:

Фигура 3.22



Когато потребителя е във втория панел, натискането на бутона назад ще го отведе към първия, а ако се намира в първия – ще го попита дали желае да излезем от приложението, при позитивен отговор то ще спре.

# ЧЕТВЪРТА ГЛАВА

# РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ

Android service-a “Smart Printer for Android” може да се използва във всеки търговски обект, независимо какво предлага той, като 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). Потребителя (търговец) трябва да изтегли файла и да го инсталира на устройство си.

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

След инсталация трябва да бъде намерена тази икона:

Фигура 4.1



При стартиране на приложението се показва първият екран:

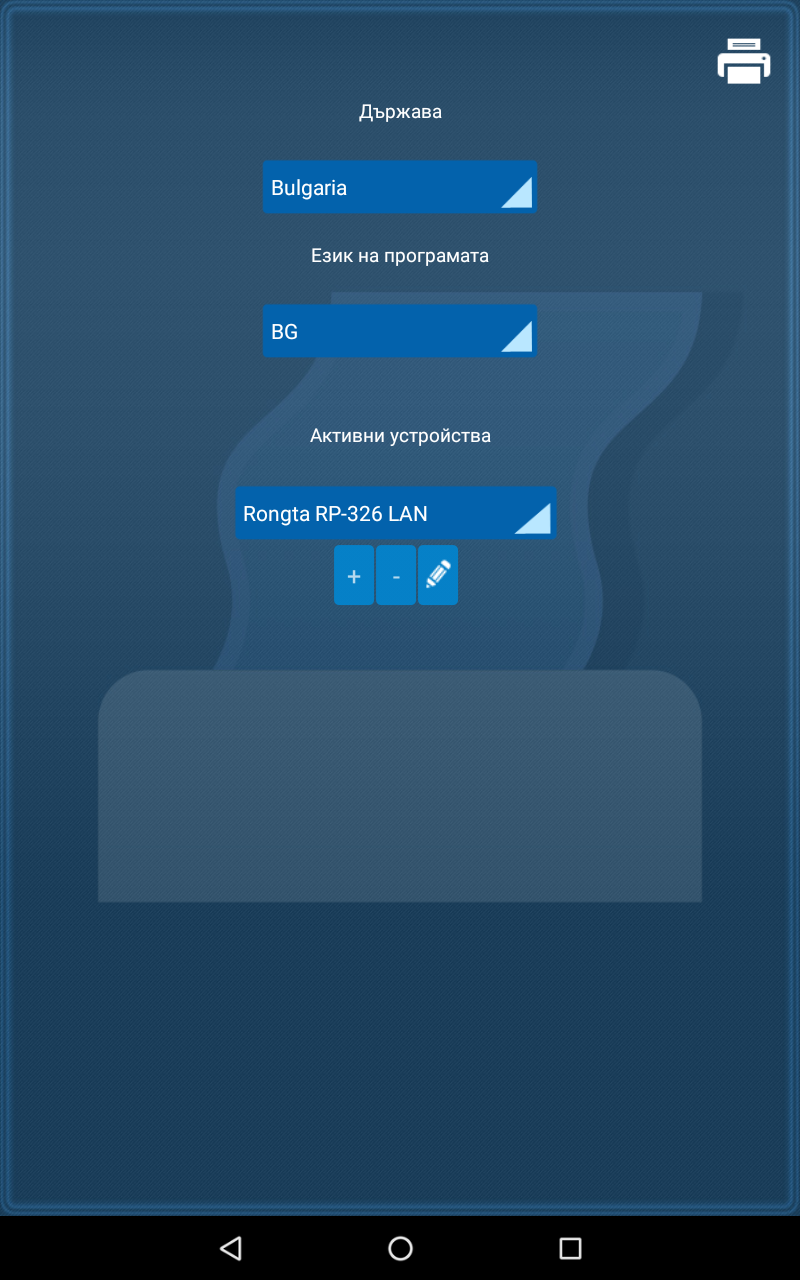
Фигура 4.2



В него се сигнализира дали потребителя е свързан към локална мрежа, показвайки IP адреса на устройството и порта. При натискане, в специален етикет, който се откроява 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, и различното законодателство на всяка държава. По-надолу се избира езикът, на който ще се използва приложението. В горната дясна част е дефиниран бутон, който при натискане визуализира плъзгащ се панел - показващ статуса на принтера, но без избран принтер, панела ще бъде празен.

За избиране на принтер, под етикета „Активни устройства“ трябва бъде натиснат бутона със символ „плюс“:

Фигура 4.4

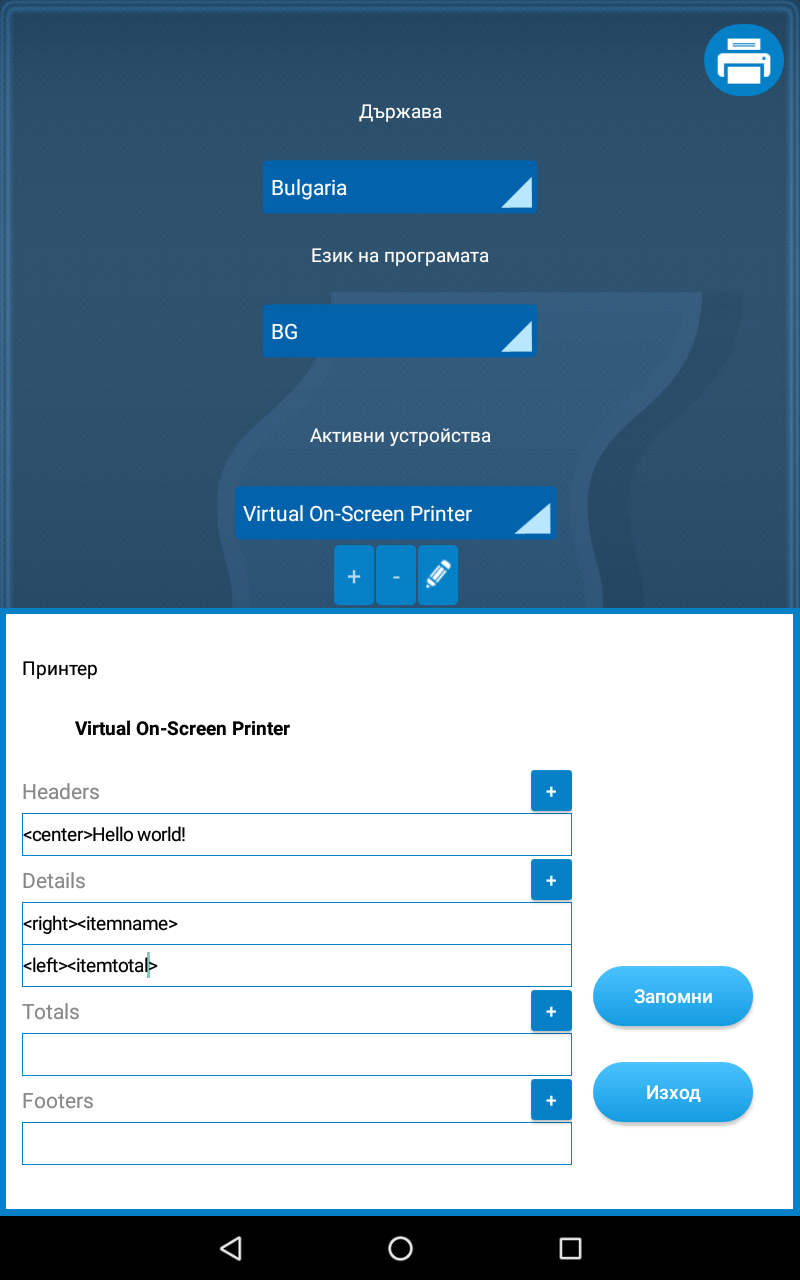


Появява се плъзгащ се панел, от който се избират принтерите, с които потребителя ще работи, показващи и прилежащите им характеристики “Baud Rate” за Serial printer, за LAN - порт и IP, за Bluetooth - MAC адрес, както и скриптове. Това позволява да се печата не само текст, който да бъде разполаган според желанията на потребителя, но и управление на атрибутите, които ще се извлекат от получената заявка, а в последствие разпределени. Например “<left><totalfull>”, ”<right><change>”,”<center><totaldiscount>”. Трябва да се има предвид, че тази част на приложението, трябва да бъде изменяна и манипулирана, само от потребители с достатъчни знания в сферата. За персонализиране на касова бележка, е възможна асистенция на квалифициран екип, който да конфигурира приложението спрямо нуждите на търговеца.

Когато потребителя, желае да редактира или изтрие конкретен принтер, трябва да го избере от падащото меню с активните принтери.

След това да натисне бутона със символ „молив“, за да редактира или съответно да изтрие – бутона със символ „минус“. При редактирането на принтер, характеристиките нужни за предаване на информацията(baud rate/ IP port, IP address) и неговите скриптове – могат да бъдат изменени частично или напълно или дори и изтрити.

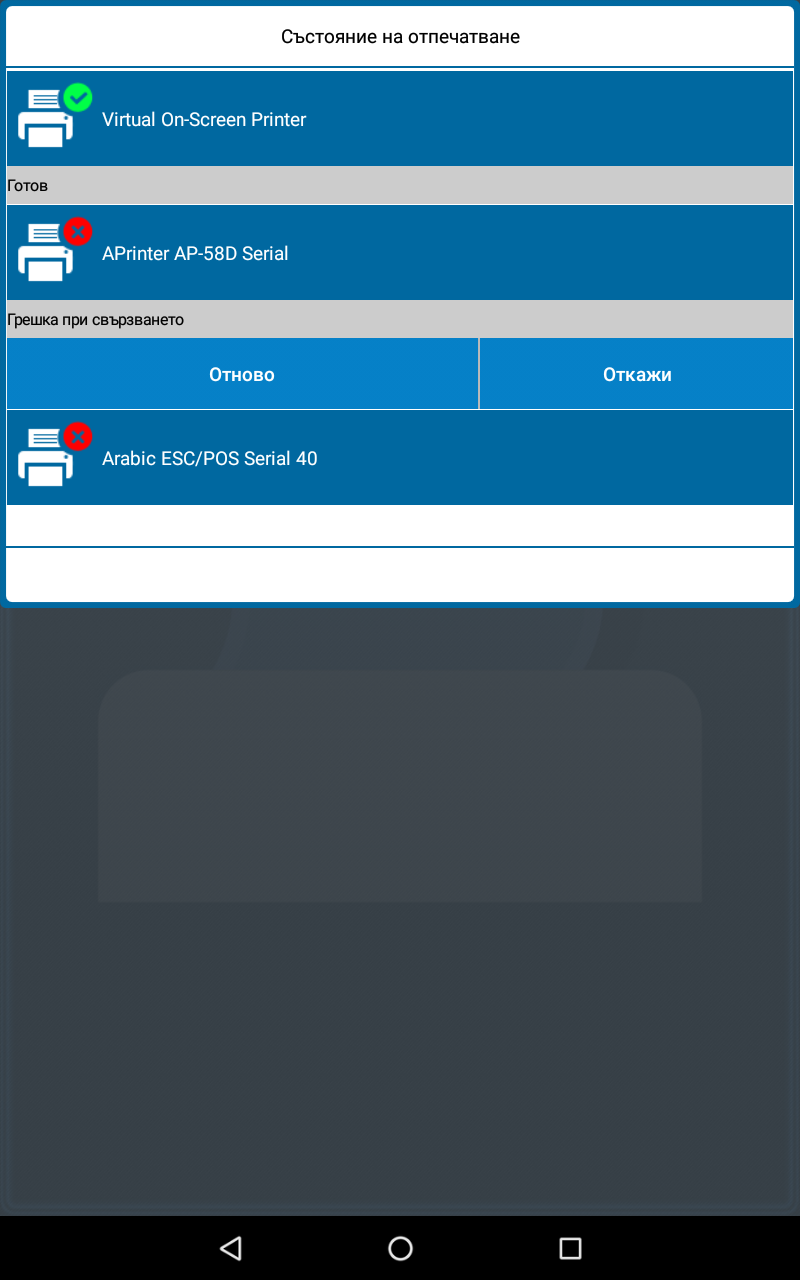
Фигура 4.5



В горната дясна част на екрана е построена известяващата част на приложението – показваща статуса на принтерите. По подразбиране, дори да не е свързан принтера, ще бъде индикиран, като готов.

При грешка, след като приложението опита, да изпрати заявка за принтиране, ще бъде показана информация за грешка и вида й.

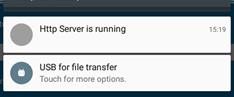
Фигура 4.6



При наличие на грешки ще се появят два бутона - за отказ на заявката, връщаща статуса в „готов“ и бутон за повторен опит за свързване и изпращане на получената заявка.

Когато приложението бъде изключено, service-a не спира работа, целта е да не се налага всеки път, когато приложението бъде случайно изключено да се активира отново. Има и уведомление:

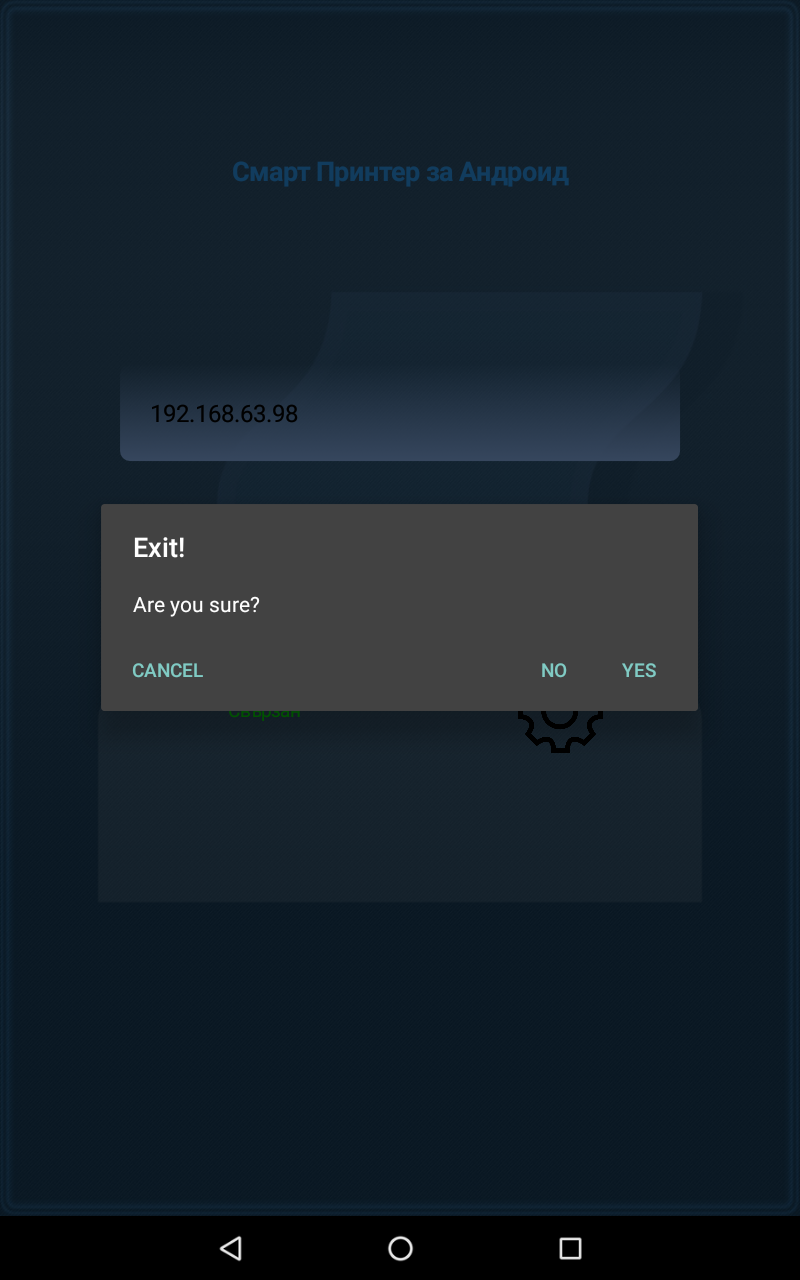
Фигура 4.7



Когато приложението е във фонов режим или е изключено от списъка със скорошните приложения, service-a ще пусне отново Activity-то при получаване на заявка и ще изпрати данните за принтиране.

За да бъде спряно приложението изцяло - или трябва да бъде спряно от настройките, принудително, или трябва да се влезе в приложението и от началния екран да бъде натиснат бутона за връщане назад. След потвърждение service-а спира.

Фигура 4.8



# Заключение

В заключение можем да кажем, че “Smart Printer for Android” задоволява нуждите на обекти от малки(квартални) магазинчета до големи вериги магазини, благодарение на удобството и мобилността си, както и разширените функционалности за използване на повече от един принтер и манипулиране на бележката от самите касиери. За момента приложението е в 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)