Пловдивски университет „Паисий Хилендарски”

Факултет по математика и информатика

Курсова работа

По дисциплина “Мобилни приложения”

На тема: „WiFi мениджър“

|  |  |
| --- | --- |
| **Изготвил**: Данаил Костадинов  Атанасов  **Специалност**: Софтуерно инженерство, 3-ти курс  **Факултетен номер**: 2001321029 | Проверил:  / доц. д‐р С. Хаджиколева/ |

**СЪДЪРЖАНИЕ**

**[Увод](#_gjdgxs) 3**

[Описание на софтуерния продукт](#_30j0zll) 3

[Мотиви за избора ми на темата на проекта](#_1fob9te) 3

[Използвани средства, технологии и други](#_3znysh7) 4

[Изисквания към приложението по зададената тема](#_2et92p0) 4

**[Реализация на софтуерния продукт](#_tyjcwt) 5**

[Общ поглед над архитектурата на софтуерния продукт](#_3dy6vkm) 5

[Архитектура на backend частта на продукта](#_1t3h5sf) 5

[Entities](#_4d34og8) 6

[Models](#_17dp8vu) 6

[Activities](#_3rdcrjn) 8

[Архитектура на дизайна на продукта](#_26in1rg) 9

**[Заключение](#_lnxbz9) 15**

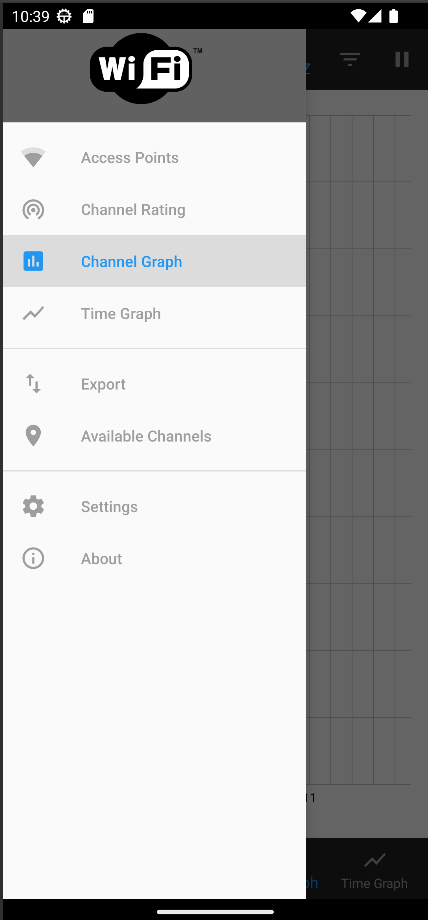
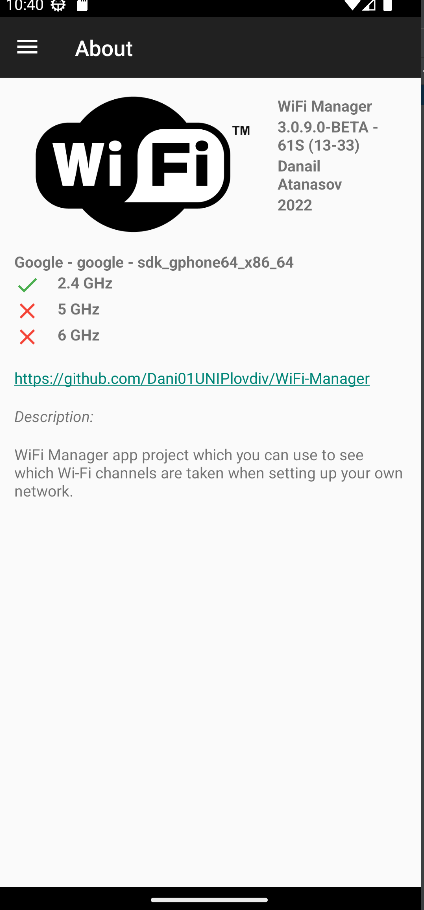
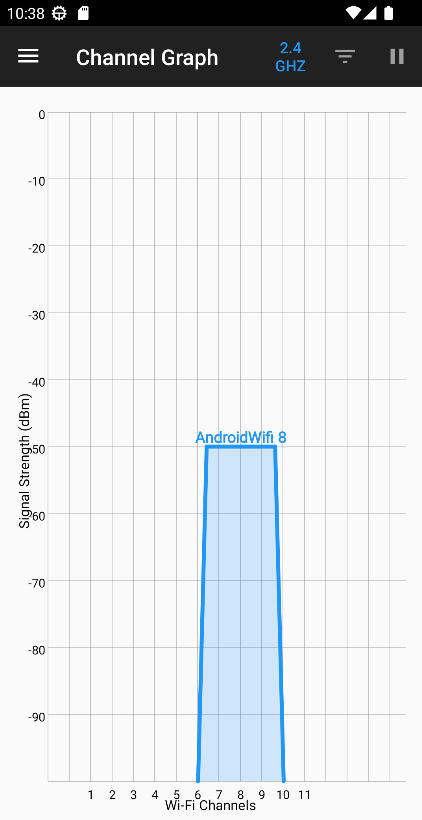
**[Използвана литература](#_35nkun2) 15**

# Увод

## Описание на софтуерния продукт

Wi-Fi мениджърът е мобилно приложение, чрез което потребителят или менъджърът на мрежата може да разглежда WiFi мрежите, техните канали и промени в реално време.

Потребителите разполагат със списък на всички мрежи до които тяхното устройство има обхват.



## Мотиви за избора ми на темата на проекта

Основната причина да избера тази тема беше поради нуждата от приложения с което може да се подпомага при настройка на безжична мрежа когато се премести мрежово устройство.

## Използвани средства, технологии и други

Приложението е изградено от android 5.0 (API level: 21) и неговия XML-синтаксис- за дизайн на самата програма, и Kotlin като език за backend-а, включващ преминаването от един frame на друг, или от едно activity към друго, както и зареждането на списъка с мрежи и за тяхната филтрация по категории.

Използван е достъп до мрежовия адаптер с който могат да се сканират Wi-Fi мрежите наблизо и да се преценят каналите им.

## Изисквания към приложението по зададената тема

Дизайнът на приложението трябва да съдържа списък с безжични мрежи, графа за канали, канален рейтинг за скорост/достъпност и графа за анализиране на мрежата в даден диапазон от време.

При избор на една от мрежите се зарежда нейната пълна информация (широчина на канала, канал, разстояние до изсточника, честота на мрежата, мрежови протокол и тн.).

При избор на меню канална графа, да се очертават всички канали на всички достъпни мрежи, като да може да се филтрира между 2.4GHz мрежи и 5Ghz.

# Реализация на софтуерния продукт

## Общ поглед над архитектурата на софтуерния продукт

Приложението е създадено посредством Kotlin и Android. Навигирането между различните изгледи и дейности е възможно на странично (side) меню, което зарежда повечето от различните екрани като отделни frame-ове, които изглеждат на потребителя като част от цялото. Използвани са разнообразни видове изгледи и оформления за в различните екрани и ситуации - LinearLayout, RelativeLayout, TableLayout, GridLayout, ScrollView, CardView, RecyclerView и други. Статичните стрингови променливи са изнесени в ‘strings.xml’ файла, за да могат да се преизползват и за да се улесни редактирането им, като по този начин ще се налага тяхната редакция само на едно място, а именно в горепосочения файл:

<string name="app\_name" translatable="false">"WiFi Manager"</string>  
<string name="app\_full\_name" translatable="false">"WiFi Manager"</string>

<string name="app\_url" translatable="false">"https://github.com/Dani01UNIPlovdiv/WiFi\_Manager"</string>

Така изглежда по най-общия начин архитектурата на софтуерното приложение. В следващите редове ще се запознаем по-подробно със структурата както на Kotlin (backend) частта, така и с дизайна на всеки един от изгледите и екраните.

## Архитектура на backend частта на продукта

Архитектурата на Kotlin частта в приложението е систематизирана в няколко основни направления - **activities**, които включват основните дейности (Main, MainContext, MainReload и DrawerNavigation), **models**, съдържащи моделите за всеки един от фреймовете, и не на последно място - **entities -** съдържащ основния модел за цялото приложение - Wi-Fi Manager. Ще преминем през всеки един от компонентите, започвайки с последния.

## Entities

В тази директория е разположен основния модел, с който се работи в backend частта на целия софтуерен продукт. Това е най-обикновен клас, който ще държи характеристиките на всяка една отделна мрежа - SSID, честота, канал, протокол за защита и . Списъкът с мрежи се показва автоматично при стартиране на приложението и се обновява всеки 5 секунди, а при разглеждане на канали се обновяват през 1 секунда.

## 

## Models

Това е директорията, в която са поместени **фреймовете** за всеки един от екраните. Всеки един от тези Kotlin класове наследява родителския клас **Fragment** и задължително override-ва неговия метод **onCreateView**, чрез който се избира кой изглед да бъде зареден във **FrameLayout** контейнера.

* WiFiConnection е първият клас - той зарежда екрана с мрежите които са достъпни за потребителя. Освен че override-ва задължително, както всеки друг клас от тази папка, **onCreateView** метода, той override-ва и **onViewCreated**, за да може да закачи click събитията и техните обработчици към всяка една от категориите, които в дизайна са представени като отделни бутони.

първо откриваме всеки един от бутоните:

val drawer = mainActivity.findViewById<DrawerLayout>(R.id.*drawer\_layout*)

след това закачаме към всеки един от тях съответния **setOnClickListener**. Най-важното при всеки един от тези обработчици на събитието е коя категория подават необходима за филтрацията,

След като един от бутоните за менюта е натиснат се зарежда нов фрагмент, който зарежда този фрагмент отговарящ.

## Activities

Това са последните компоненти на backend частта на този софтуерен продукт.

MainActivity-то е главното активити на проекта. В него се зарежда toolbar менюто

actionBarDrawerToggle = createDrawerToggle(drawer)

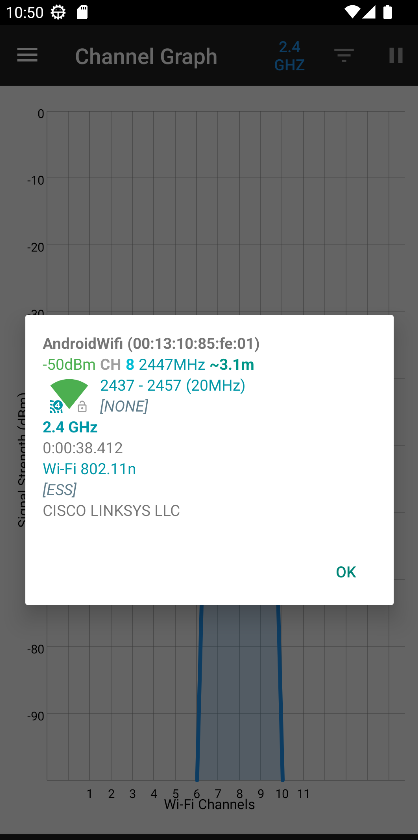
както и **NavigationView**, в което ще се заредят опциите на страничното меню. Когато приложението се отваря за първи път, **savedInstanceState** няма да има стойност и в този случай е указано да се зарежда основният фрагмент.

Към заглавието на приложението, което е разположено в toolbar-менюто, е закачено click събитие и когато бъде кликнато, да се отваря или затваря страничното меню.

В това activity е override-нато и събитието (**onNavigationItemSelected**), което се предизвиква при кликване на някоя от опциите на страничното меню - това довежда до зареждането на съответния фрагмент.

В главния фрагмент на приложението се показват всички мрежи като те се извличат директно от мрежовия контролер:

fun makeView(convertView: View?,  
 parent: ViewGroup?,  
 wiFiDetail: WiFiDetail,  
 child: Boolean = false,  
 @LayoutRes layout: Int = MainContext.INSTANCE.settings.accessPointView().layout)  
 : View {  
 val view = convertView ?: MainContext.INSTANCE.layoutInflater.inflate(layout, parent, false)  
 setViewCompact(view, wiFiDetail, child)  
 setViewExtra(view, wiFiDetail)  
 setViewVendorShort(view, wiFiDetail.wiFiAdditional)  
 return view  
}

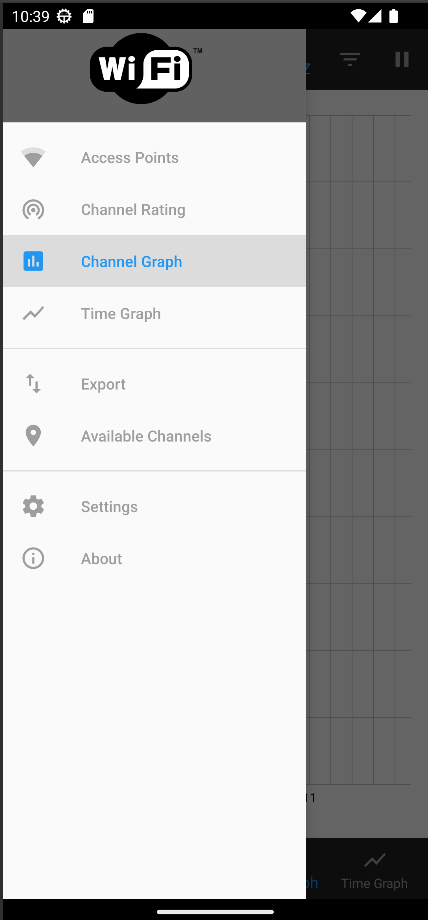
Те се извличат използвайки правата на Android систамата за контрол над хардуерните устройства.

При натискане на сегмент показващ мрежа се отваря Pop-up прозорец, който показва повече информация за тази мрежа.

fun makeViewDetailed(wiFiDetail: WiFiDetail): View {  
 val view = MainContext.INSTANCE.layoutInflater.inflate(R.layout.access\_point\_view\_popup, null)  
 setViewCompact(view, wiFiDetail, false)  
 setViewExtra(view, wiFiDetail)  
 setViewCapabilitiesLong(view, wiFiDetail)  
 setViewVendorLong(view, wiFiDetail.wiFiAdditional)  
 setViewWiFiBand(view, wiFiDetail.wiFiSignal)  
 setView80211mc(view, wiFiDetail.wiFiSignal)  
 setViewWiFiStandard(view, wiFiDetail.wiFiSignal)  
 setTimestamp(view, wiFiDetail.wiFiSignal)  
 enableTextSelection(view)  
 return view  
}

## Архитектура на дизайна на продукта

Ще започнем с разглеждането на структурата на дизайн частта на приложението от главния файл - **activity\_main.xml**.

 В него е поместен **DrawerLayout**, благодарение на който можем да изградим плъзгащото се странично меню. Веднага след него е разположен линеарен layout с вертикална ориентация, за да може фрагментите, които се зареждат в frame-контейнерът, да застават под toolbar. Именно в този **LinearLayout** е разположен **toolbar**-ът (**androidx.appcompat.widget.Toolbar**), в който се поместват заглавието на приложението и бутонът, чрез който достъпваме настройките.

Toolbar-ът има зададени теми, които се прилагат при отворено и затворен меню.

*android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"*

*app:popupTheme="@style/ThemeOverlay.AppCompat.Light"*

В рамките на **LinearLayout** е и поместен **frame-контейнерът**, в който се зарежда всеки един от фрагментите.

*<****FrameLayout***

*android:id="@+id/fragment\_container"*

*android:layout\_width="match\_parent"*

*android:layout\_height="match\_parent" />*

Веднага след **LinearLayout** се намира **NavigationView-**то, в което е указано откъде да извлече layout-а на header-а на side менюто и самото меню:

*app:headerLayout="@layout/nav\_header"*

*app:menu="@layout/drawer\_menu”*

Заглавната част на страничното меню е описана в отделен файл - **nav\_header.xml**. Тук отново е използва **LinearLayout**, в който са поместени **ImageView** и **TextView**. Снимката, използвана в header-a, е добавена допълнително в drawables.

Менюто с неговите опции също са декларирани в отделен файл - **drawer\_menu.xml**. Важна характеристика на менюто е - *tools:showIn="navigation\_view",* който оказва, че това меню се отваря в **NavigationView**, каквото имаме декларирано в activity\_main.xml.

При натискане на един от бутоните се зарежда нов фрагмент спрямо името на бутона:

navigationMenuController = NavigationMenuController(this)  
navigationMenuController.currentNavigationMenu(settings.selectedMenu())  
onNavigationItemSelected(currentMenuItem())  
  
connectionView = ConnectionView(this)

# Заключение

Това според мен е основата за едно приложение предназначено за менижиране на безжични мрежи, то трябва да е максимално опростено, с малък брой фрагменти, като могат да се разширят функционалностите му, например да се направи начин за геолокиране на мрежите из града в който живее потребителя или да могат да се споделят настройките за дадена мрежа само чрез QR код, но за сега този дизайн и функционалности са достатъчни за почти всеки начинаещ се мрежови ентусиаст