

Aplikacja przypominająca o zażyciu leków “MedReminder”

Przedmiot: Aplikacje mobilne w medycynie

Autorzy:

Sophia Krupnik

Angelika Kielbasa

Michał Komala

Cel projektu

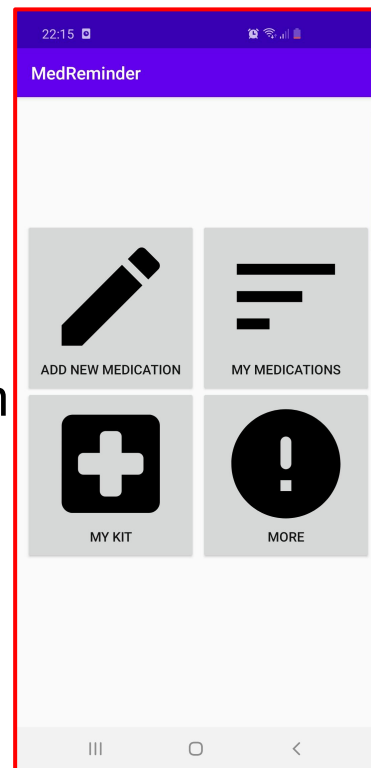
Stworzenie aplikacji, która będzie wysyłać użytkownikowi powiadomienia przypominające o zażyciu dawki leku.

Użytkownik może wprowadzić godzinę o której ma otrzymywać codziennie powiadomienia. Możliwe jest wprowadzenie więcej niż jednego leku, a prowadzone kuracje są przechowywane i wyświetlane w postaci listy.

Działanie aplikacji

Przyciski:

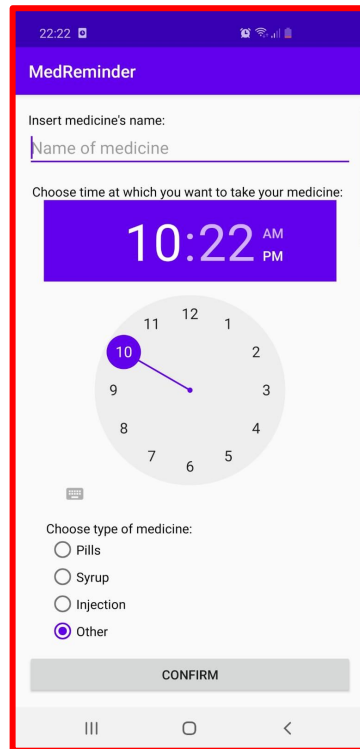
- » “Add new medication” - przechodzi do ekranu dodawania nowego leku
- » “My medication” - przechodzi do ekranu, w którym wyświetlana jest lista obecnych kuracji
- » “My kit” - przechodzi do ekranu, w którym wyświetlana jest lista obecnie zażywanych leków
- » “More” - informacje o aplikacji oraz poradnik



Działanie aplikacji

Ekran “Add new medication”:

- » wprowadzenie nazwy leku
- » wybranie godziny powiadomienia
- » wybór rodzaju leku
- » “Confirm” dodaje nowy alarm



22:22

MedReminder

Insert medicine's name:

Name of medicine

Choose time at which you want to take your medicine:

10:22 AM PM

10

Choose type of medicine:

☐ Pills

☐ Syrup

☐ Injection

☒ Other

CONFIRM

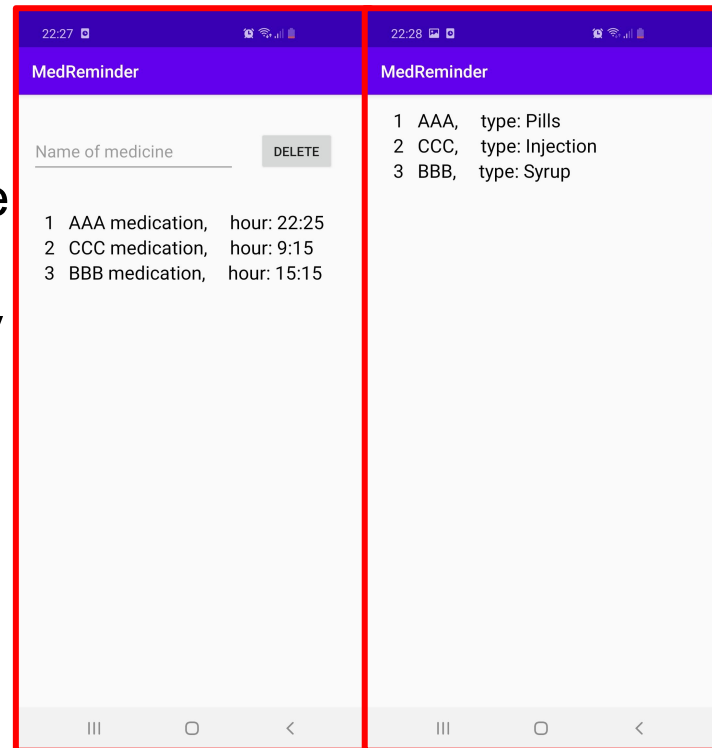
Działanie aplikacji

Ekran “My Medications”:

- » wyświetla obecne kuracje oraz wybrane godziny alarmu
- » usuwanie kuracji (wprowadzenie nazwy leku + “Delete”)

Ekran “My kit”:

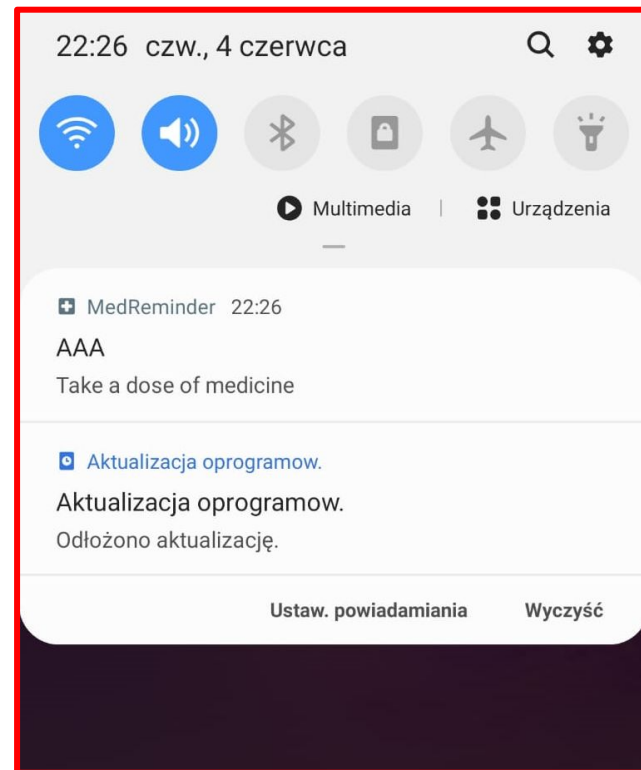
- » wyświetla zażywane leki oraz informacje jakiego są rodzaju



Działanie aplikacji

Powiadomienie:

- » pojawia się mniej więcej o określonej godzinie (+/- 2 minuty)
- » zawiera nazwę leku, którego dawkę należy zażyć
- » pojawia się codziennie do momentu usunięcia kuracji
- » towarzyszą mu sygnał dźwiękowy SMS'a i wibracje



Architektura

- » zamiana ekranów (kliknięcie dowolnego przycisku w Menu)
 - wyświetlany ekran to Fragment
 - zamiana widocznego Fragmentu przy użyciu `supportFragmentManager` i dostępnych w nim funkcji `beginTransaction()` i `replace()`

```
private fun changeFragment(fragment: Fragment, addToBackStack : Boolean) {  
    val transaction : FragmentTransaction = supportFragmentManager.beginTransaction()  
    transaction.replace(R.id.main_container, fragment)  
    if(addToBackStack){  
        transaction.addToBackStack( name: null)  
    }  
    transaction.commit()  
}
```

Architektura

- » elementy uzupełniane przez użytkownika
 - EditText - pole do wpisywania nazw
 - TimePicker - zegar do wyboru godziny
 - RadioButton - pojedynczy przycisk do wyboru typu leku
 - RadioGroup - zbiór kilku przycisków RadioButton

Architektura

- » przechowywanie i wyświetlanie informacji o kuracjach
 - przechowywane w `sharedPreferences`
 - wyświetlane z użyciem metody z wykładów (`RecyclerView`)
- » generowanie powiadomienia
 - utworzenie kanału dla powiadomień z aplikacji przy użyciu `NotificationManager` i dostępnej w nim funkcji `createNotificationChannel()` (jeden kanał może być używany przez wiele powiadomień)
 - przypisanie każdej kuracji unikalnego ID powiadomienia (zajęte numery przechowywane w `sharedPreferences`)

Architektura

- ustawienie powiadomienia o określonej godzinie przy użyciu alarmManager (można wybrać jak często ma być powtarzany alarm np. codziennie)
- utworzenie klasy AlarmReceiver, w której definiujemy formę powiadomienia (przy użyciu NotificationCompat.Builder podajemy treść powiadomienia, obrazek itp., dodatkowo podajemy parametry wibracji oraz odgłos powiadomienia)
- jednym z parametrów alarmManagera jest Intent, który wymaga definicji klasy AlarmReceiver
- ostatecznie trzeba dodać receiver AlarmReceiver oraz wyrazić aplikacji zgodę na wibracje w AndroidManifest

Architektura

```
private fun setAlarm(medicineName : String, notificationID: Int, medicineHourInt : Int, medicineMinuteInt : Int) {  
    val calendar: Calendar = Calendar.getInstance().apply { this: Calendar  
        timeInMillis = System.currentTimeMillis()  
        set(Calendar.HOUR_OF_DAY, medicineHourInt)  
        set(Calendar.MINUTE, medicineMinuteInt)  
        set(Calendar.SECOND, 0)  
    }  
  
    alarmMgr = context?.getSystemService(Context.ALARM_SERVICE) as AlarmManager  
    alarmIntent = Intent(context, AlarmReceiver::class.java).let { intent ->  
        var extras = bundleOf( ...pairs:  
            "notificationID" to notificationID,  
            "medicineName" to medicineName  
        )  
        intent.putExtras(extras)  
        PendingIntent.getBroadcast(context, notificationID, intent, flags: 0) ^let  
    }  
  
    alarmMgr?.setInexactRepeating(  
        AlarmManager.RTC_WAKEUP,  
        calendar.timeInMillis,  
        AlarmManager.INTERVAL_DAY,  
        alarmIntent  
    )  
}
```

Sposób wykorzystania AlarmManagera w projekcie

Architektura

```
class AlarmReceiver : BroadcastReceiver() {  
  
    lateinit var notificationManager : NotificationManager  
  
    @RequiresApi(Build.VERSION_CODES.O)  
    override fun onReceive(context: Context?, intent: Intent?) {  
        //get passed data  
        val notificationID = intent!!.getIntExtra( name: "notificationID", defaultValue: 0)  
        val medicineName = intent.getStringExtra( name: "medicineName")  
  
        notificationManager = context?.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager  
        val builder = NotificationCompat.Builder(context, channelId: "medReminder_channel")  
            .setSmallIcon(R.drawable.local_hospital_black)  
            .setContentTitle(medicineName)  
            .setContentText("Take a dose of medicine")  
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)  
  
        //set vibrations  
        val v = context.getSystemService(Context.VIBRATOR_SERVICE) as Vibrator  
        v.vibrate( milliseconds: 500L)  
  
        notificationManager.notify(notificationID,builder.build())  
    }  
}
```

Zdefiniowana klasa AlarmReceiver w projekcie

Architektura

- » wyświetlanie komunikatów
 - Toast - do wyświetlania poszczególnych błędów lub potwierdzenia wykonanego działania

- » wygląd przycisków
 - wykorzystano ikony dostępne na stronie *material.io*, znajduje się tam duża baza ikon, które można pobierać i umieszczać w folderze *drawable* w projekcie

Problemy

1. Jak wywołać z poziomu obiektu A metodę w obiekcie B?
 - zdefiniować nazwę oraz parametry metody w interfejsie, a następnie interfejs zaimplementować w obu obiektach
2. Tworzenie nowego AlarmManagera
 - dość problematyczne, dlatego polecamy dokumentację na oficjalnej stronie Androida
 - na stronie zademonstrowane są przykładowe linijki kodu, które można skopiować i zmodyfikować pod swój projekt

Problemy

3. Przerywanie kuracji

- anulowanie alarmu powiadomienia również przy pomocy AlarmManagera, rada: zapobiec możliwości wpisania dwa razy tej samej nazwy leku, ponieważ przy zakończeniu kuracji z listy znikną oba pola o tej samej nazwie, ale usunie się jedynie jeden alarm (będziemy otrzymywać powiadomienia o nieistniejącej kuracji)

Dziękujemy za uwagę

Dla chętnych link do Githuba z gotową aplikacją :
<https://github.com/mrkomal/MedReminder.git>



pcworld.pl