Laboratorijska vježba 7

Servisi

Zadatak 1

Kreirati servis u prvom planu koji je aktivan i kad aplikacija nije. Servis svaki sat vrši poziv ka web servisu koji vraća najnoviji dodani film u bazu. Nakon klika na notifikaciju koja se pojavi, otvara se nova simbolična aktivnost koja sadrži osnovne informacije o novom filmu.

U prvom koraku ćemo zahtijevati određene permisije u `Manifest` file-u.

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Ovim zahtijevamo permisije za pokretanje servisa u prvom planu i za `WAKE_LOCK` kako ne bi došlo do prekidanja servisa u `DOZE` ili `STANDBY` režimu rada.

Nakon zahtijevanja permisija, kreirat ćemo servis `New->Service->Service`, koji se zove `LatestMovieService`. Pošto se servis ne veže za komponentu, imamo sljedeću override-anu metodu:

```
override fun onBind(intent: Intent): IBinder? {
    return null
}
```

Definisat ćemo i sljedeće privatne varijable:

```
//za spriječavanje prekida u slučaja Daze
private var wakeLock: PowerManager.WakeLock? = null
//oznaka da je srevis pokrenut
private var isServiceStarted = false
//api ključ
private val tmdb_api_key : String = BuildConfig.TMDB_API_KEY
//primjer filma- novi filmovi ne moraju sadržavati sve podatke
private var movie = Movie(1,"test","test","test","test","test","test","test")
```

Override-ana `onCreate` metoda glasi ovako:

```
override fun onCreate() {
    super.onCreate()
    val notification = createNotification()
    startForeground(1, notification)
}
```

U prvom koraku odmah kreiramo notifikaciju da se traži novi film. Ova notifikacija će biti uvijek aktivna - ovo je po zahtjevima novijih verzija OS-a. Nakon toga pokrenemo servis u prvom planu.

Metoda `createNotification` je data u nastavku:

```
private fun createNotification(): Notification {
    val notificationChannelId = "LATEST MOVIE SERVICE CHANNEL"
```

```
    //Različite metode kreiranja notifikacije
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
        //Kreiramo notifikacijski kanal - notifikaicje se šalju na isti kanal
        val channel = NotificationChannel(
                notificationChannelId,
                "Latest Movie notifications channel",
                NotificationManager.IMPORTANCE_HIGH
        ).let {
        //Definišemo karakteristike notifikacije
            it.description = "Latest Movie Service channel"
            it.enableLights(true)
            it.lightColor = Color.RED
            it.enableVibration(true)
            it.vibrationPattern = longArrayOf(100, 200, 300, 400, 500, 400, 300, 200,
400)
            it
        }
        notificationManager.createNotificationChannel(channel)
    }
    //gradimo notifikaciju u ovisnosti od verzije
    val builder: Notification.Builder = if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) Notification.Builder(
            this,
            notificationChannelId
    ) else Notification.Builder(this)
    //Kreira se notifikacija koja će se prikazati kada se servis pokrene
    return builder
        .setContentTitle("Finding latest film")
        .setSmallIcon(android.R.drawable.ic_popup_sync)
        .setTicker("Film")
        .setPriority(Notification.PRIORITY_HIGH) // ako je ispod api 26
        .build()
}
```

Kada se servis pokrene poziva se `onStartCommand`. Istu metodu ćemo override-ati.

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
    startService()
    // servis će se restartovati ako ovo vratimo
    return START_STICKY
}
```

Unutar ove metode ćemo pozvati `startService`. U nastavku je data ista funkcija.

```
private fun startService() {
//U slučaju da se ponovo pokrene ne mora se pozivati ova metoda
    if (isServiceStarted) return
//Postavimo da je servis pokrenut
    isServiceStarted = true
//Koristit ćemo wakeLock da spriječimo gašenje servisa
    wakeLock =
```

```
            (getSystemService(Context.POWER_SERVICE) as PowerManager).run {
                newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
"LatestMovieService::lock").apply {
                    acquire()
                }
            }
    //Beskonačna petlja koja dohvati podatke svakih sat vremena
    GlobalScope.launch(Dispatchers.IO) {
        while (isServiceStarted) {
            launch(Dispatchers.IO) {
                getData()
            }
            //Sačekaj sat vremena prije nego se ponovo pokreneš
            delay(3600000)
        }
    }
}
```

U ovoj metodi poziv ka servisu je odvojen u coroutine. Već smo ranije spomenuli da se servisi ne izvršavaju na odvojenoj niti. Pošto poziv ka web servisu blokira rad aplikacije, moramo ga odvojiti u coroutine koja se odvija na pozadinskoj niti. Nakon što prikupimo film iz poziva servisa, kreirat ćemo novu notifikaciju koja sa sobom sadrži i intent za otvaranje nove aktivnosti. Da bismo otvorili aktivnost na ovaj način, potrebno je da koristimo `PendingIntent`. `PendingIntent` je referenca na token koji sistem čuva i koji opisuje odgovarajuće podatke. U slučaju da se aplikacija ugasi i njeni procesi unište, `PendingIntent` će ostati i moći će se iskoristiti kasnije zajedno sa svim svojim podacima. On ostaje validan sve dok se ne ukloni. U nastavku je dat prikaz poziva web servisa i definisanja `PendingIntent`.

```
private fun getData() {
    try {
        val url1 =
                "https://api.themoviedb.org/3/movie/latest?api_key=${tmdb_api_key}"
        val url = URL(url1)
        (url.openConnection() as? HttpURLConnection)?.run {
            val result = this.inputStream.bufferedReader().use { it.readText() }
            val jsonObject = JSONObject(result)
            movie.title = jsonObject.getString("title")
            movie.id = jsonObject.getLong("id")
            movie.releaseDate = jsonObject.getString("release_date")
            movie.homepage = jsonObject.getString("homepage")
            movie.overview = jsonObject.getString("overview")
            //Radi očuvanja pristojnosti
            if (!jsonObject.getBoolean("adult")) {
                movie.backdropPath = jsonObject.getString("backdrop_path")
                movie.posterPath = jsonObject.getString("poster_path")
            }
        }
        //Kreiramo notify event - kreirat ćemo posebnu klasu za prikaz prikupljenih
podataka
        val notifyIntent = Intent(this, MovieDetailResultActivity::class.java).apply {
        //Naglašavamo da je riječ o posebnoj aktivnosti koja samo služi za prikaz
```

```
podataka notifikacije
            flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
        //Kako smo ovdje poslali film?
             putExtra("movie",movie)
        }
        val notifyPendingIntent = PendingIntent.getActivity(
                this, 0, notifyIntent, PendingIntent.FLAG_UPDATE_CURRENT
        )
        //Kreiramo notifikaciju na istom kanalu samo s drugačijim idom notifikacije -
ovu notifikaciju uklanjamo
        val notification = NotificationCompat.Builder(baseContext, "LATEST MOVIE
SERVICE CHANNEL").apply{
            setSmallIcon(android.R.drawable.stat_notify_sync)
            setContentTitle("New movie found")
            setContentText(movie.title)
            setContentIntent(notifyPendingIntent)
            setOngoing(false)
            build()
        }
        with(NotificationManagerCompat.from(applicationContext)) {
            notify(123, notification.build())
        }
    } catch (e: MalformedURLException) {
        Log.v("MalformedURLException", "Cannot open HttpURLConnection")
    } catch (e: IOException) {
        Log.v("IOException", "Cannot read stream")
    } catch (e: JSONException) {
        Log.v("IOException", "Cannot parse JSON")
    }
}
```

Cjelokupni servis je dat u nastavku:

```
class LatestMovieService : Service() {
    private var wakeLock: PowerManager.WakeLock? = null
    private var isServiceStarted = false
    private val tmdb_api_key : String = BuildConfig.TMDB_API_KEY
    private var movie = Movie(1,"test","test","test","test","test","test","test")
    override fun onBind(intent: Intent): IBinder? {
        return null
    }
    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        startService()
        // servis će se restartovati ako ovo vratimo
        return START_STICKY
    }
    override fun onCreate() {
        super.onCreate()
        val notification = createNotification()
        startForeground(1, notification)
    }
    private fun startService() {
```

```
        if (isServiceStarted) return
        isServiceStarted = true
        // we need this lock so our service gets not affected by Doze Mode
        wakeLock =
                (getSystemService(Context.POWER_SERVICE) as PowerManager).run {
                    newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
"LatestMovieService::lock").apply {
                        acquire()
                    }
                }
        //Beskonačna petlja koja dohvati podatke svakih sat vremena
        GlobalScope.launch(Dispatchers.IO) {
            while (isServiceStarted) {
                launch(Dispatchers.IO) {
                    getData()
                }
                delay(3600000)
            }
        }
    }
    private fun getData() {
        try {
            val url1 = "https://api.themoviedb.org/3/movie/latest?
api_key=${tmdb_api_key}"
            val url = URL(url1)
            (url.openConnection() as? HttpURLConnection)?.run {
                val result = this.inputStream.bufferedReader().use { it.readText() }
                val jsonObject = JSONObject(result)
                movie.title = jsonObject.getString("title")
                movie.id = jsonObject.getLong("id")
                movie.releaseDate = jsonObject.getString("release_date")
                movie.homepage = jsonObject.getString("homepage")
                movie.overview = jsonObject.getString("overview")
                if (!jsonObject.getBoolean("adult")) {
                    movie.backdropPath = jsonObject.getString("backdrop_path")
                    movie.posterPath = jsonObject.getString("poster_path")
                }
            }
            val notifyIntent = Intent(this,
MovieDetailResultActivity::class.java).apply {
                flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
                putExtra("movie",movie)
            }
            val notifyPendingIntent = PendingIntent.getActivity(
                    this, 0, notifyIntent, PendingIntent.FLAG_UPDATE_CURRENT
            )
            val notification = NotificationCompat.Builder(baseContext, "LATEST MOVIE
SERVICE CHANNEL").apply{
                setSmallIcon(android.R.drawable.stat_notify_sync)
                setContentTitle("New movie found")
                setContentText(movie.title)
```

```
                setContentIntent(notifyPendingIntent)
                setOngoing(false)
                build()
            }
            with(NotificationManagerCompat.from(applicationContext)) {
                notify(123, notification.build())
            }
        } catch (e: MalformedURLException) {
            Log.v("MalformedURLException", "Cannot open HttpURLConnection")
        } catch (e: IOException) {
            Log.v("IOException", "Cannot read stream")
        } catch (e: JSONException) {
            Log.v("IOException", "Cannot parse JSON")
        }
    }
    private fun createNotification(): Notification {
        val notificationChannelId = "LATEST MOVIE SERVICE CHANNEL"
        //Različite metode kreiranja notifikacije
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE)
as NotificationManager
            val channel = NotificationChannel(
                    notificationChannelId,
                    "Latest Movie notifications channel",
                    NotificationManager.IMPORTANCE_HIGH
            ).let {
                it.description = "Latest Movie Service channel"
                it.enableLights(true)
                it.lightColor = Color.RED
                it.enableVibration(true)
                it.vibrationPattern = longArrayOf(100, 200, 300, 400, 500, 400, 300,
200, 400)
                it
            }
            notificationManager.createNotificationChannel(channel)
        }
        val builder: Notification.Builder = if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.O) Notification.Builder(
                this,
                notificationChannelId
        ) else Notification.Builder(this)
        return builder
                .setContentTitle("Finding latest film")
                .setSmallIcon(android.R.drawable.ic_popup_sync)
                .setTicker("Film")
                .setPriority(Notification.PRIORITY_HIGH) // ako je ispod api 26
                .build()
    }
}
```

Kreirajmo sada našu novu aktivnost koja treba prikazati podatke o filmu. Aktivnost ćemo kreirati kao do sada. Za layout ćemo iskoristiti onaj definisan za detalje o

filmu bez liste sličnih filmova i glumaca. Nova aktivnost će drugačije biti definisana u `Manifest` file-u. Potrebno je da postaivmo roditeljsku aktivnost, zatim način pokretanja ove aktivnosti, odnosno ova aktivnost je `Task` koji se pokrene jednom i koji se ne stavlja u `backstack` i čiji je zadatak samo da pokaže podatke o filmu.

```
<activity
    android:name=".MovieDetailResultActivity"
    android:parentActivityName=".MainActivity"
    android:launchMode="singleTask"
    android:taskAffinity=""
    android:excludeFromRecents="true"/>
```

Podatke iz `PendingIntent-a` kupimo na isti način kao iz običnog. U nastavku slijedi kôd nove aktivnosti.

```
class MovieDetailResultActivity : AppCompatActivity() {
    private  var movie= Movie(0, "Test", "Test", "Test", "Test", "Test", "Test",
"Test")
    private lateinit var title : TextView
    private lateinit var overview : TextView
    private lateinit var releaseDate : TextView
    private lateinit var genre : TextView
    private lateinit var website : TextView
    private lateinit var poster : ImageView
    private lateinit var backdrop : ImageView
    private val posterPath = "https://image.tmdb.org/t/p/w780"
    private val backdropPath = "https://image.tmdb.org/t/p/w500"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_movie_detail_result)
        title = findViewById(R.id.movie_title)
        overview = findViewById(R.id.movie_overview)
        releaseDate = findViewById(R.id.movie_release_date)
        genre = findViewById(R.id.movie_genre)
        poster = findViewById(R.id.movie_poster)
        website = findViewById(R.id.movie_website)
        backdrop = findViewById(R.id.movie_backdrop)
        val notificationManager = getSystemService(NOTIFICATION_SERVICE) as
NotificationManager
        notificationManager.cancel(123)
        if(intent?.getParcelableExtra<Movie>("movie")!==null) {
            movie=intent?.getParcelableExtra<Movie>("movie")!!
            populateDetails()
        }
    }
    private fun populateDetails() {
        title.text=movie.title
        releaseDate.text=movie.releaseDate
        genre.text=movie.genre
        website.text=movie.homepage
        overview.text=movie.overview
        val context: Context = poster.getContext()
```

```
        var id = 0;
        if (movie.genre!==null)
            id = context.getResources()
                .getIdentifier(movie.genre, "drawable", context.getPackageName())
        if (id===0) id=context.getResources()
            .getIdentifier("picture1", "drawable", context.getPackageName())
        Glide.with(context)
            .load(posterPath + movie.posterPath)
            .placeholder(R.drawable.picture1)
            .error(id)
            .fallback(id)
            .into(poster);
        var backdropContext: Context = backdrop.getContext()
        Glide.with(backdropContext)
            .load(backdropPath + movie.backdropPath)
            .centerCrop()
            .placeholder(R.drawable.backdrop)
            .error(R.drawable.backdrop)
            .fallback(R.drawable.backdrop)
            .into(backdrop);
    }
}
```

Ono što možemo primijetiti u ovom kôdu da mi film šaljemo kroz `Intent`. Ovo ranije nikada nismo radili. Kako bismo poslali instancu naše klase pomoću intenta, naša klasa mora da implementira interfejs `Parcelable`. Samim tim naša klasa `Movie` će sada izgledati ovako:

```
data class Movie (
        var id: Long,
        var title: String,
        var overview: String,
        var releaseDate: String,
        var homepage: String?,
        var genre: String?,
        var posterPath: String,
        var backdropPath: String
):Parcelable {
    constructor(parcel: Parcel) : this(
            parcel.readLong(),
            parcel.readString()!!,
            parcel.readString()!!,
            parcel.readString()!!,
            parcel.readString(),
            parcel.readString(),
            parcel.readString()!!,
            parcel.readString()!!) {
    }
    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeLong(id)
        parcel.writeString(title)
        parcel.writeString(overview)
```

```
        parcel.writeString(releaseDate)
        parcel.writeString(homepage)
        parcel.writeString(genre)
        parcel.writeString(posterPath)
        parcel.writeString(backdropPath)
    }
    override fun describeContents(): Int {
        return 0
    }
    companion object CREATOR : Parcelable.Creator<Movie> {
        override fun createFromParcel(parcel: Parcel): Movie {
            return Movie(parcel)
        }
        override fun newArray(size: Int): Array<Movie?> {
            return arrayOfNulls(size)
        }
    }
}
```

`Parcelable` predstavlja Android implementaciju Java `Serializible` klase. Da bismo implementirali ovaj intefejs potrebno je da implementiramo metode `writeToParcel`, `describeContents`, te konstruktor koji prima `Parcel`. Potrebno je implementirati objekat `Parcelable.Creator`.

Za kraj ostane još da pokrenemo servis u glavnoj aktivnosti u `onCreate` metodi:

```
Intent(this, LatestMovieService::class.java).also {
//Različito pokretanje u ovisnosti od verzije
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        startForegroundService(it)
        return
    }
    startService(it)
}
```

Konačni izgled aplikacije je dat u nastavku: