PERSISTENCE IN ANDROID

OSC-ADA

11.05.2019.

Agenda

- Uvod
- Shared Preferences

</> primjer

- O bazama i SQL-u kratko

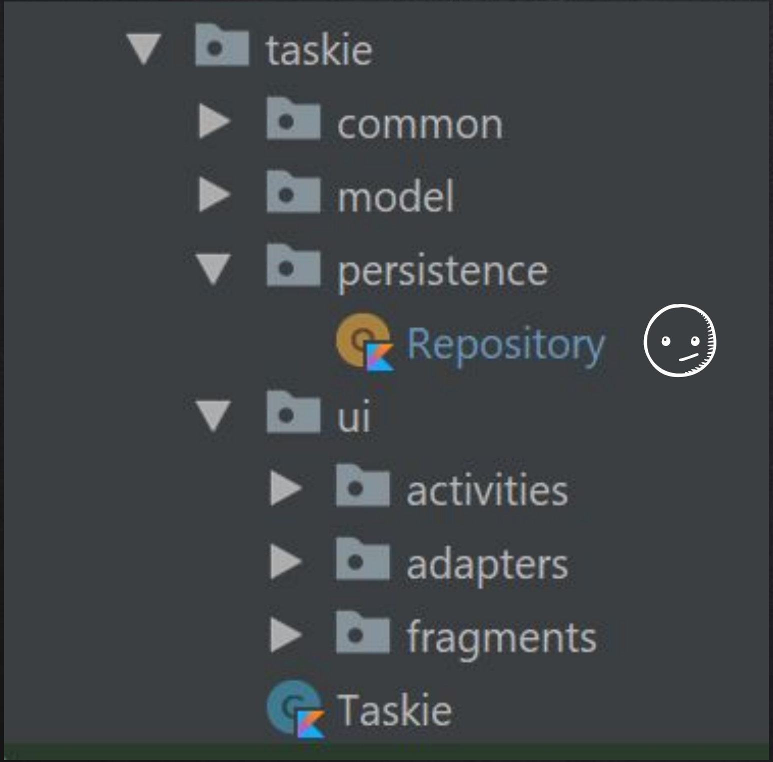
- Room library

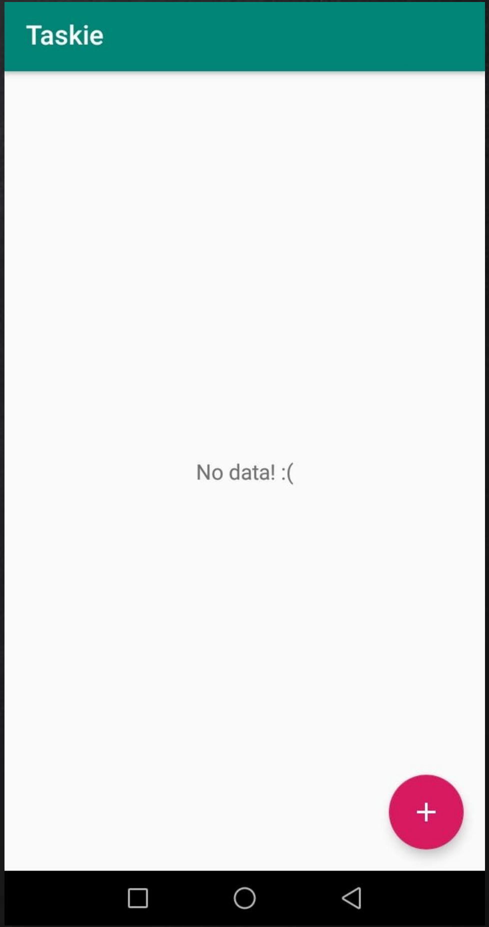
</> Read & write

</> delete & update

- Homework (upute) i još neki dodaci

PAUZA – negdje na pola puta :D

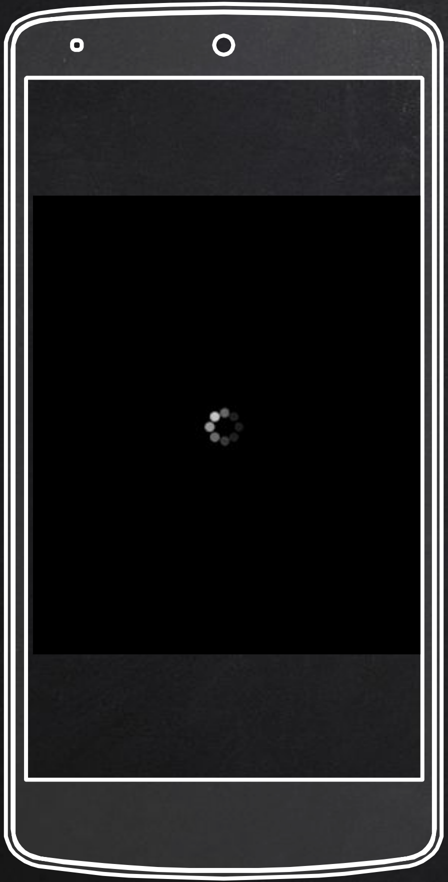

taskie
common
model
persistence
Repository
ui
activities
adapters
fragments
Taskie


Taskie
No data! :(

Android Data Persistence

ON-DEVICE / OFFLINE / LOCAL

OFF-DEVICE / ONLINE



“

Ovo mi je najdraži screen

Niko Nikad

Internal file storage

Shared Preferences

STORAGE OPTIONS

External file storage

Database

Internal file storage

- Sustav pruža privatni direktorij svakoj aplikaciji za njezine potrebe
- Prilikom deinstaliranja aplikacije datoteke iz internal storage-a se brišu
- Ako baš želimo dijetlit datoteke to radimo preko FileProvider-a

```
val filename = "myfile"
val fileContents = "Hello world!"
context.openFileOutput(filename, Context.MODE_PRIVATE).use {
        it.write(fileContents.toByteArray())
}
```

External file storage

- Dijeljeni prostor za sve aplikacije
- Može biti da se fizički može odvojiti od uređaja
- Prilikom pristupanja datotekama potrebno provjeriti dostupnost
- Koristi se za podatke koji se dijele među aplikacijama i koji trebaju ostati sačuvani i nakon deinstalacije aplikacije

npr. SD kartica

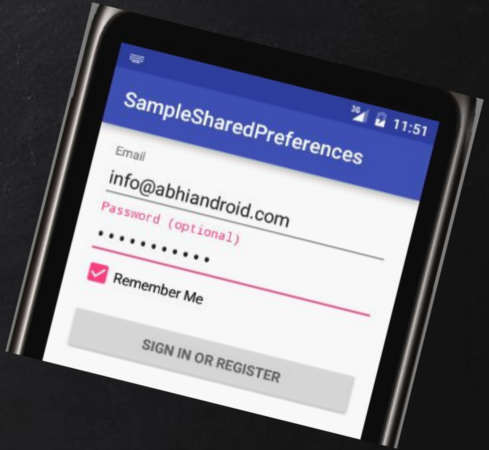
```
<manifest ...>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

Standardne javne datoteke

Više o internal i external storage-u Možeš saznati ovdje:

https://developer.android.com/training/data-storage/files

1.


SampleSharedPreferences

11:51

Email

info@abhiandroid.com

Password (optional)

••••••••••

✓ Remember Me

SIGN IN OR REGISTER

Shared preferences

Let's save key-value data!

Kreiranje ili pristupanje

.getSharedPreferences()

- Za kreiranje datoteke prema imenu
- Prvi parametar ime
- Može se pozvati na bilo kojem contextu

.getPreferences()

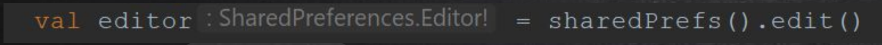
- Za kreiranje datoteke za određeni Activity
- Ne treba ime
- Poziva se iz Activity-a

.getDefaultSharedPreferences()

- Za kreiranje datoteke na razini cijele aplikacije
- Najčešće se koristi za spremanje nekakvih postavki aplikacije

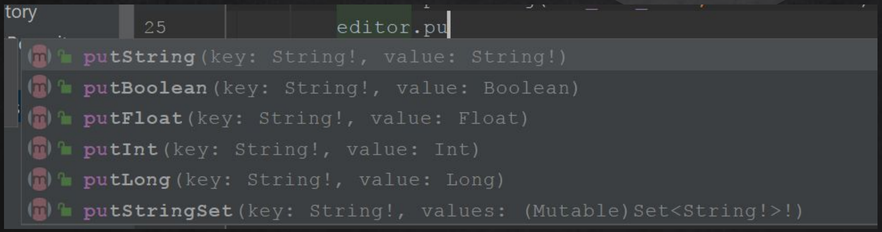
```
private fun sharedPrefs() =
PreferenceManager.getDefaultSharedPreferences(BestPizzasApplication.getAppContext())
```

Write

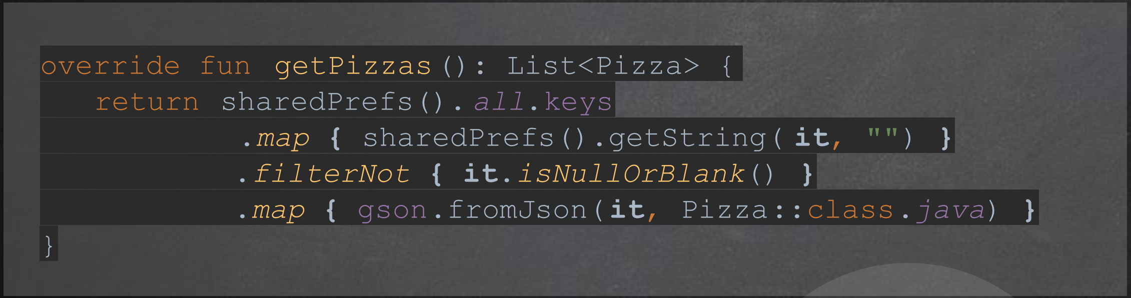


```
val editor : SharedPreferences.Editor! = sharedPrefs().edit()
```

$$\begin{tikzcd}\text{editor} \arrow[r] & \text{put()} \arrow[r] & \text{apply()}\end{tikzcd}$$



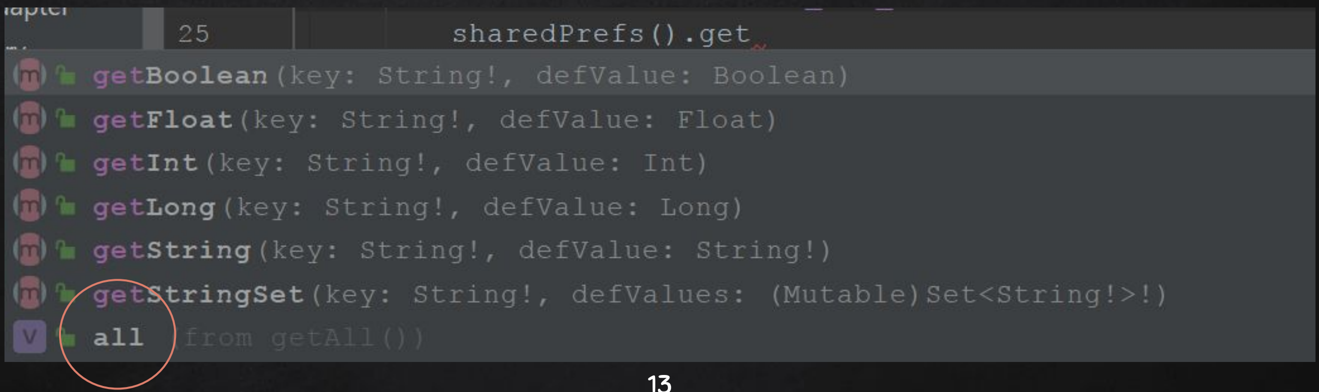
```
tory    25        editor.pu
        putString(key: String!, value: String!)
        putBoolean(key: String!, value: Boolean)
        putFloat(key: String!, value: Float)
        putInt(key: String!, value: Int)
        putLong(key: String!, value: Long)
        putStringSet(key: String!, values: (Mutable)Set<String!>!)
```



```kotlin
override fun getPizzas(): List<Pizza> {
    return sharedPrefs().all.keys
            .map { sharedPrefs().getString(it, "") }
            .filterNot { it.isNullOrBlank() }
            .map { gson.fromJson(it, Pizza::class.java) }
}
```

READ

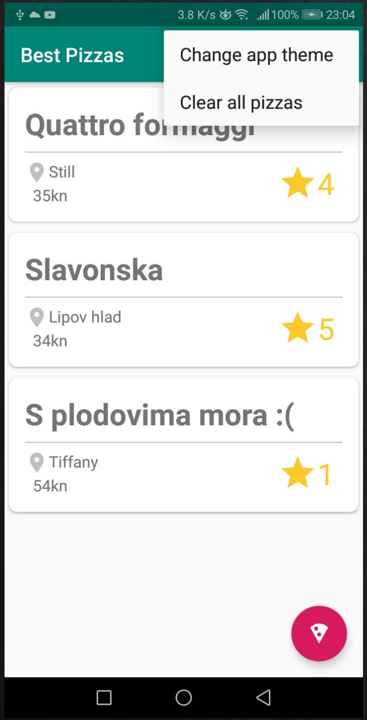
get()



```
        sharedPrefs().get
getBoolean(key: String!, defValue: Boolean)
getFloat(key: String!, defValue: Float)
getInt(key: String!, defValue: Int)
getLong(key: String!, defValue: Long)
getString(key: String!, defValue: String!)
getStringSet(key: String!, defValues: (Mutable)Set<String!>!)
all (from getAll())
```

ZADATAK

-> U projektu Best Pizzas omogućiti korisniku promjenu teme aplikacije koristeći sharedPrefs



3.8 K/s 100% 23:04

Best Pizzas

Change app theme

Clear all pizzas

Quattro formaggi

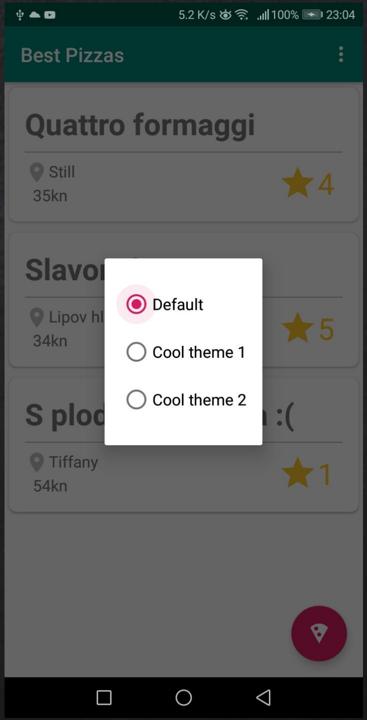
Still 35kn ★4

Slavonska

Lipov hlad 34kn ★5

S plodovima mora :(

Tiffany 54kn ★1



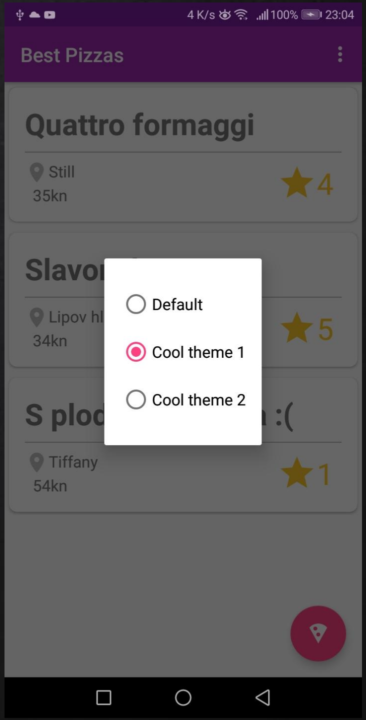
5.2 K/s 100% 23:04

Best Pizzas

[✓ Default]

Cool theme 1

Cool theme 2



4 K/s 100% 23:04

Best Pizzas

Default

[✓ Cool theme 1]

Cool theme 2

Rješenje </>

```
object PizzaPrefs {

   const val KEY_THEME_NAME = "KEY_THEME_NAME"

   private fun sharedPrefs() =
PreferenceManager.getDefaultSharedPreferences(BestPizzasApplication.getAppContext())

   fun store(key: String, value: String) {
       val editor = sharedPrefs().edit()
       editor.putString(key , value).apply()
   }

   fun getString(key: String,  defaultValue: String ): String? =
sharedPrefs().getString(key ,defaultValue )

}
```

</>

```
//Cool theme 1
<color name="themeOneColorPrimary" >#9C27B0</color>
<color name="themeOneColorPrimaryDark" >#7B1FA2</color>
<color name="themeOneColorAccent" >#FF4081</color>

//Cool theme 2
<color name="themeTwoColorPrimary" >#9E9E9E</color>
<color name="themeTwoColorPrimaryDark" >#616161</color>
<color name="themeTwoColorAccent" >#009688</color>
```

```xml
<!-- Base application theme. -->
<style name="Base.Theme.App" parent="Theme.AppCompat.Light.DarkActionBar" >
</style>

<style name="Theme.App.Default" parent="Base.Theme.App" >
   <item name="colorPrimary" >@color/colorPrimary </item>
   <item name="colorPrimaryDark" >@color/colorPrimaryDark </item>
   <item name="colorAccent" >@color/colorAccent </item>
</style>

<style name="Theme.App.ThemeOne" parent="Base.Theme.App" >
   <item name="colorPrimary" >@color/themeOneColorPrimary </item>
   <item name="colorPrimaryDark" >@color/themeOneColorPrimaryDark </item>
   <item name="colorAccent" >@color/themeOneColorAccent </item>
</style>

<style name="Theme.App.ThemeTwo" parent="Base.Theme.App" >
   <item name="colorPrimary" >@color/themeTwoColorPrimary </item>
   <item name="colorPrimaryDark" >@color/themeTwoColorPrimaryDark </item>
   <item name="colorAccent" >@color/themeTwoColorAccent </item>
</style>
```

```kotlin
abstract class BaseActivity : AppCompatActivity() {

    private val currentTheme: String? = PizzaPrefs.getString(PizzaPrefs.KEY_THEME_NAME, "Default")

    override fun onCreate(savedInstanceState: Bundle?) {
        setAppTheme()
        super.onCreate(savedInstanceState)

        setContentView(getLayoutResourceId())
        setUpUi()

    }

    private fun setAppTheme() {
        when (currentTheme) {
            DEFAULT_THEME -> setTheme(R.style.Theme_App_Default)
            COOL_THEME_ONE -> setTheme(R.style.Theme_App_ThemeOne)
            COOL_THEME_TWO -> setTheme(R.style.Theme_App_ThemeTwo)
            else -> setTheme(R.style.Theme_App_Default)
        }
    }


    protected fun showFragment(fragment: Fragment) {
        showFragment(R.id.fragmentContainer, fragment)
    }

    abstract fun getLayoutResourceId(): Int
    abstract fun setUpUi()

    companion object {
        private const val DEFAULT_THEME = "Default"
        private const val COOL_THEME_ONE = "Cool theme 1"
        private const val COOL_THEME_TWO = "Cool theme 2"
    }
}
```

```
class MainActivity : BaseActivity() {

...

private fun saveTheme(themeName: String) {
    PizzaPrefs.store(PizzaPrefs.KEY_THEME_NAME, themeName)
    recreate()
}

private fun getCurrentThemeName(): String? {
    return PizzaPrefs.getString(PizzaPrefs.KEY_THEME_NAME, "")
}



}
```

2.

DATABASE

Let’s save real data!

Primary key

id	type	place	price
123	bla	bla	bla
456	bla	bla	bla
789	lab	bla	bla

SQLite databases

SQL

Jezik koji omogućuje :

- pristup bazi podataka
- dohvaćanje podataka iz baze
- dodavanje novih podataka u bazu
- brisanje postojećih podataka iz baze
- izmjenu postojećih podataka u bazi

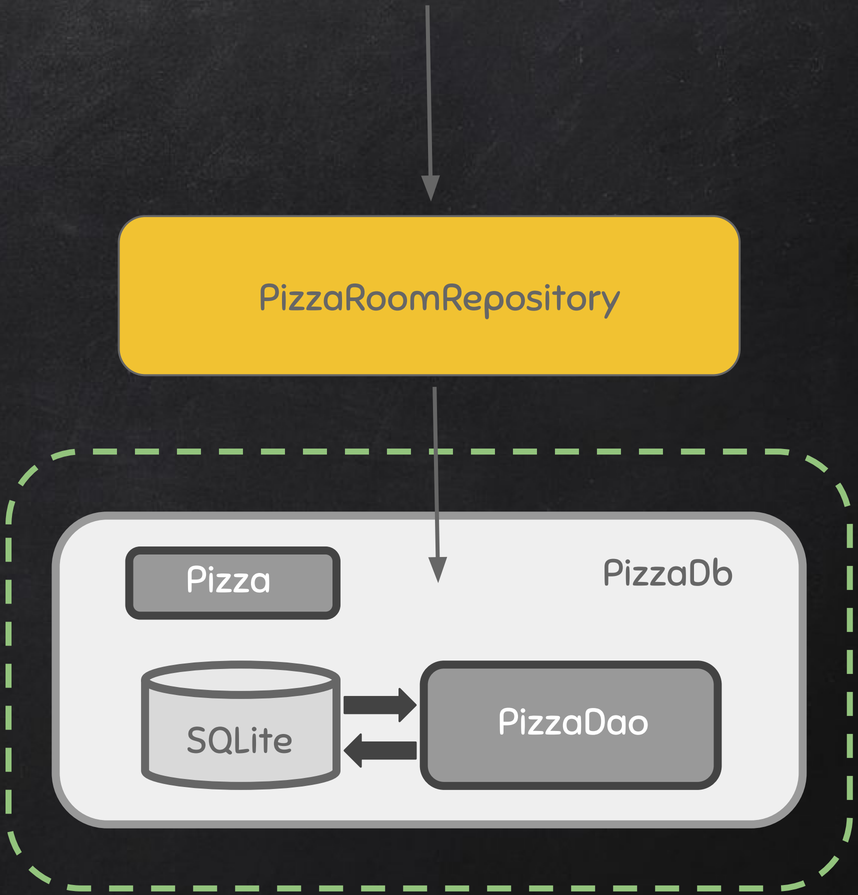
```
SELECT * FROM Customers
WHERE Country='Mexico';
```

ROOM
lib

@Database

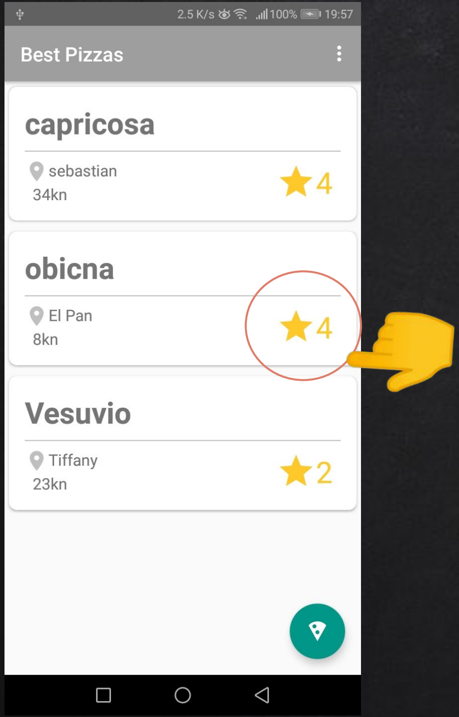
@Entity

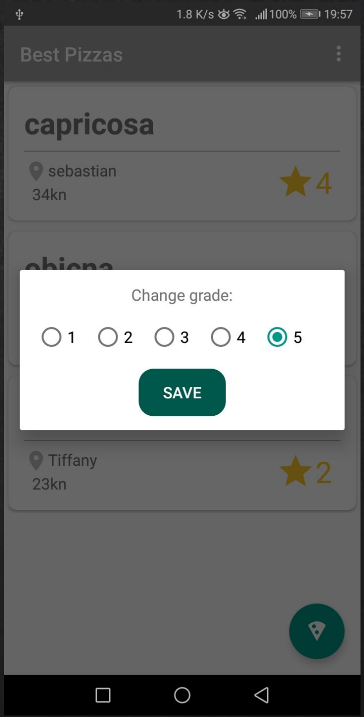
@Dao

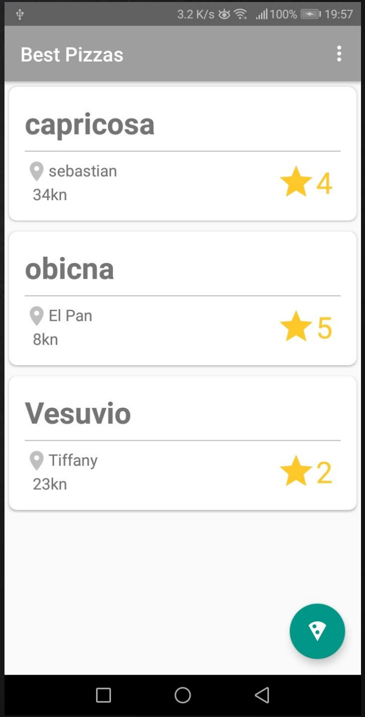


ZADATAK

-> koristeći Room omogućiti spremanje pizza, prikazivanje istih u listi, brisanje svih zajedno i mijenjanje ocjene







Rješenje </>

build.gradle
dependencies {}

```
def room_version = "1.1.1"

implementation "android.arch.persistence.room:runtime: $room_version"
kapt "android.arch.persistence.room:compiler: $room_version"
```

```kotlin
@Entity
data class Pizza(
    @PrimaryKey(autoGenerate = true)
    var pizzaDbId: Long? = null,
    var id: Int = 0,
    val type: String,
    val place: String,
    val price: Int,
    var grade: Int
)
```

```
@Database(entities = [Pizza::class], version = 1)
abstract class DaoProvider : RoomDatabase() {

   abstract fun pizzaDao(): PizzaDao

   companion object {
       private var instance: DaoProvider? = null

       fun getInstance(context: Context): DaoProvider {

           if (instance == null) {
               instance = Room.databaseBuilder(
                   context. applicationContext,
                   DaoProvider::class.java,
                   "PizzaDb"
               ).allowMainThreadQueries().build()
           }
           return instance as DaoProvider
       }
   }

}
```

```
@Dao
interface PizzaDao {

    @Query("SELECT * FROM Pizza")
    fun loadAll(): List<Pizza>

    @Query("SELECT * FROM PIZZA WHERE pizzaDbId = :pizzaId" )
    fun getPizza(pizzaId: Long): Pizza

    @Insert(onConflict = IGNORE)
    fun insertPizza(pizza: Pizza): Long

    @Update(onConflict = REPLACE)
    fun updatePizza(pizza: Pizza)

    @Delete
    fun deletePizza(pizza: Pizza)

    @Query("DELETE FROM Pizza")
    fun deleteAllPizzas()

    @Query("UPDATE pizza SET grade = :pizzaGrade WHERE pizzaDbId = :pizzaId " )
    fun changePizzaGrade(pizzaId: Long, pizzaGrade: Int)
}
```

```
class PizzaRoomRepository : BestPizzasRepository {

    private var db: DaoProvider = DaoProvider.getInstance(BestPizzasApplication.getAppContext())

    private var pizzaDao: PizzaDao = db.pizzaDao()

    override fun addPizza(pizza: Pizza) {
        pizzaDao.insertPizza(pizza)
    }

    override fun getPizzas(): List<Pizza> {
        return pizzaDao.loadAll()
    }

...
    override fun clearAllPizzas() {
        pizzaDao.deleteAllPizzas()
    }

    fun changePizzaGrade(pizza: Pizza, grade: Int) {
        pizzaDao.changePizzaGrade(pizza.pizzaDbId, grade)
    }
}
```

```
private val repository: BestPizzasRepository = FakeRepository()
```

```
private val repository: BestPizzasRepository = PizzaRoomRepository()
```

Homework

Upgrade your Taskie project

1. Koristeći SharedPreferences spremiti zadnji izabrani prioritet i njega staviti kao defaultni prilikom kreiranja novog taska

2. Implementirati Room library tako da se :

- Task doda u bazu na save click
- Taskovi dohvate iz baze i prikažu u listi
- Task briše iz baze na swipe u lijevo
- Task može izmijeniti u TaskDetailsFragmentu i omogućiti spremanje izmjene

3. Dodati menu u kojem se taskovi mogu:

- Poredati po prioritetu
- Obrisati svi kompletno

BONUS

4. Dodati AlertDialog prije brisanja jednog taska i svih taskova

DODATAK

Ovdje imate objašnjeno kako spremati ne primitivne tipove podataka (kao što je kod vas slučaj s prioritetom koji je enum) u room pomoću TypeConverter-a

https://medium.com/@FizzyInTheHall/converting-types-with-room-and-kotlin-9ee45da5e3ac

Hvala na pažnji! ☺

terezija.umiljanovic@gmail.com

Credits

Special thanks to all the people who made and released these awesome resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ Photographs by Unsplash