# Mobil Programozási
# Alapok

## Kotlin nyelven megírt
## Shopping list kibővítése 5 új elemmel

**Készítette:** Kocsis Katalin

**Neptun kód:** WGOWUG

A KotlinShoppingList nevű mobil alkalmazás grafikus felületét XML-fájlban van elkészítve, és az eseménykezelők Kotlin nyelven kerültek megvalósításra. Most új elemekkel bővítjük ki az alkalmazást, hogy még teljesebben kielégítse a felhasználói igényeket. Új elemek bevezetésével lehetővé tesszük, hogy a felhasználó részletesebben és személyre szabottabban kezelje a vásárlási listáját. A következő új elemek kerülnek bevezetésre:

- Bolt neve
- termék márkája,
- termék darabszáma
- termék kategóriája
- termék allergén tartalma

Az alkalmazáshoz tartozik egy adatbázis. A ShoppingItem osztályunk egy adatbázis táblát készít el, aminek a neve shoppingitem. Minden item egy tábla lesz. Ennek a táblának az egyes oszlop nevei jelenleg a termék neve, és a termék ára valamint egy check box ami alapján be tudjuk pipálni ha az adott terméket megvettük. Ezt Bővítjük a fent említett elemekkel.

```
@Entity(tableName = "shoppingitem")
data class ShoppingItem(@PrimaryKey(autoGenerate = true) var itemId: Long?,
                        @ColumnInfo(name = "name") var name: String,
                        @ColumnInfo(name = "price") var price: Int,
                        @ColumnInfo(name = "bought") var bought: Boolean,
                        @ColumnInfo(name = "shop") var shop: String,
                        @ColumnInfo(name = "brand") var brand: String,
                        @ColumnInfo(name = "piece") var piece: Int,
                        @ColumnInfo(name = "category") var category: String,
                        @ColumnInfo(name = "allergen") var allergen: String
) : Serializable
```

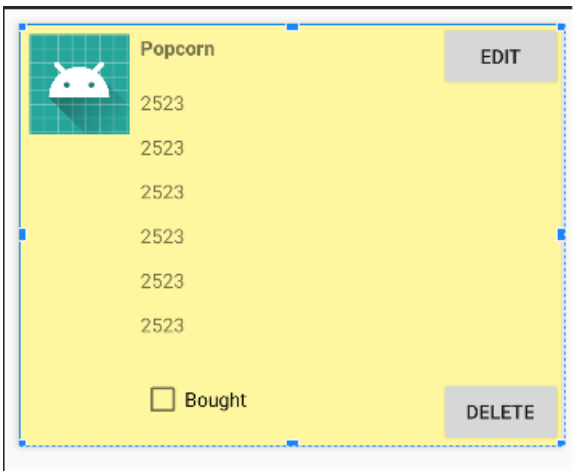
A termék darabszámát Int-tel adtuk meg úgy, mint a termék árát. Ide csak számértékek kerülhetnek. A többi elemet pedig szövegesen lehet megadni, azt hogy megvettük-e a terméket az pedig boolean értéket kapott. A táblához tartozik egy automatikusan generált kulcs. Ha megvannak az új elemek, akkor majd a ShoppingitemDAO fogja a műveleteket ellátni. A listázást, új item beszúrását, item törlését, és az egyes mezők frissítését. Az AppDatabase készíti el az adatbázist a Shoppingitem táblája alapján.

A táblába felvittük az új mezőket, és szeretnénk ezt a mobil képernyőjén is szemléltetni. A továbbiakban a row_item nevű xml fájlban az alábbi módon bővítjük az alkalmazásunkat az új elemekkel.

row_item.xml

```
<TextView
    android:id="@+id/tvShop"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="52dp"
    android:layout_marginRight="8dp"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="@+id/tvName"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvName"
    tools:text="2523" />

<TextView
    android:id="@+id/tvBrand"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="84dp"
    android:layout_marginRight="8dp"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintLeft_toLeftOf="@+id/tvName"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvName"
    tools:text="2523" />

<TextView
    android:id="@+id/tvPiece"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="116dp"
    android:layout_marginRight="8dp"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="@+id/tvName"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvName"
    tools:text="2523" />

<TextView
    android:id="@+id/tvCategory"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="148dp"
    android:layout_marginRight="8dp"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="@+id/tvName"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvName"
    tools:text="2523" />

<TextView
    android:id="@+id/tvAllergen"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="180dp"
    android:layout_marginRight="8dp"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="@+id/tvName"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvName"
    tools:text="2523" />
```

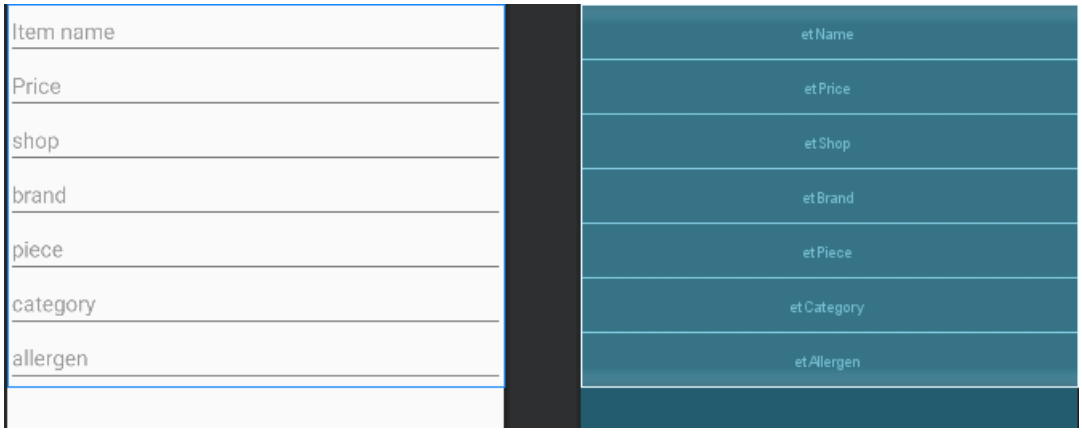
Láthatjuk, hogy annyi eleme lesz amennyi az adott listában lesz. Minden elemnek saját id neve lesz, és minden egyes elem az a Lista nevétől fogja nézni az egyes elemek elhelyezkedését a képernyőn. A design részen belül így fog kinézni az adott Listaegy item-je.



A dialog_create_item xml fájlban amikor új elemet akarunk létrehozni, vagy módosítani akkor ugyanazokat a mezőket lehet módosítani amiket létrehoztunk. Tehát itt is van a termék neve, ára, és az új elemeket is meg kell adni a következő módon:

```
<EditText
    android:id="@+id/etShop"
    android:hint="shop"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/etBrand"
    android:hint="brand"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/etPiece"
    android:hint="piece"
    android:inputType="numberDecimal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/etCategory"
    android:hint="category"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/etAllergen"
    android:hint="allergen"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Mindegyik újonnan felvitt mezőnek van egyedi nevű id-ja. Meghatároztuk, hogy a darabszám mezőnél a beviteli érték csak szám lehet. A design részben látható hogy itt mindegyik egy-egy sor lesz egymás alatt. Itt nem kell egymáshoz igazítani a sorokat.




A grafikus felület elkészítése után az eseménykezelőkkel foglalkozunk. A MainActivity kotlin nyelven megírt osztály kezeli az új elem felvétel gombját, új elem felvitelének a vizsgálatát teszi lehetővé, lekéri a shopping elemeket, és az egyes elem szerkesztésekor a teljes shopping listát teszi be, ami átadja a dialógus ablaknak a paramétereket az edit gomb megnyomásakor. Az új elem felvételekor is egy teljes shopping itemet hoz létre.

A ShoppingItemDialog osztályban egy dialógus ablakot készít, ami az új itemek felvitelére, és meglévő elem módosítására alkalmas. Ebben az osztályban szükséges az új elemeket is felvenni amikkel bővítettük a táblánkat, mivel ezeket is kell majd használnunk az adott item szerkesztésénél vagy elkészítésekor.

```
private lateinit var etItem: EditText
private lateinit var etPrice: EditText
private lateinit var etShop: EditText
private lateinit var etBrand: EditText
private lateinit var etPiece: EditText
private lateinit var etCategory: EditText
private lateinit var etAllergen: EditText
```

A dialógus egyes elemeit inicializáljuk. A dialog_create xml-t fogja ez használni.

A dialógus egyes elemeit inicializáljuk a következő lépésben. Itt minden egyes elemet el fogunk érni a create_dialog xml-ből. Egy-egy beviteli mezőt az előbb felvett változókba tesszük.

```
private fun initDialogContent(builder: AlertDialog.Builder) {
    /*etItem = EditText(activity)
    builder.setView(etItem)*/

    val rootView = requireActivity().layoutInflater.inflate(R.layout.dialog_create_item,  root: null)
    etItem = rootView.etName
    etPrice = rootView.etPrice
    etShop=rootView.etShop
    etBrand=rootView.etBrand
    etPiece=rootView.etPiece
    etCategory=rootView.etCategory
    etAllergen=rootView.etAllergen
    builder.setView(rootView)

    val arguments = this.arguments
    if (arguments != null &&
            arguments.containsKey(MainActivity.KEY_ITEM_TO_EDIT)) {
        val item = arguments.getSerializable(
                MainActivity.KEY_ITEM_TO_EDIT) as ShoppingItem
        etItem.setText(item.name)
        etPrice.setText(item.price.toString())
        etShop.setText(item.shop)
        etBrand.setText(item.brand)
        etPiece.setText(item.piece.toString())
        etCategory.setText(item.category)
        etAllergen.setText(item.allergen)

        builder.setTitle("Edit todo")
    }
}
```

Ha kapott argumentumot az edit rész, azaz létezik az adott item amit módosítani akarunk, akkor az egyes paramétereit átírhatjuk. Ha nem tudjuk szerkeszteni, nem kap argumentumot az edit kódrész, akkor az új elem felvitelét (create) fogja jelenteni. Tehát az initDialogContent függvényünk a texteket (elemeket) kéri le ( ár, termék név, márka, darabszám, kategória, allergén, bolt neve) Cardok szerint, megnézzük hogy kapott-e argumentumot, ha kapott argumentumot nem az új elem fog feljönni hanem az edit, azaz lehet módosítani. A módosításkor ellenőrizve van, hogy nem lehet egy adott mező üresen. Ha üresen hagytunk egy mezőt és az ok gombra kattintunk, akkor mindenképp figyelmeztetést ír ki. Ezt a módosított elemeket majd a DAO kapja meg.

Az edit résznél szintén lemásolja a shopping itemet, amit kapott argumentumként, és itt történik a tényleges módosítás.

```
    private fun handleItemEdit() {
        val itemToEdit = arguments?.getSerializable(
                MainActivity.KEY_ITEM_TO_EDIT) as ShoppingItem
        itemToEdit.name = etItem.text.toString()
        itemToEdit.price = etPrice.text.toString().toInt()
        itemToEdit.shop=etShop.text.toString()
        itemToEdit.brand=etBrand.text.toString()
        itemToEdit.piece=etPiece.text.toString().toInt()
        itemToEdit.category=etCategory.text.toString()
        itemToEdit.allergen=etAllergen.text.toString()

        shoppingItemHandler.shoppingItemUpdated(itemToEdit)
    }
}
```

Ezek a módosított adatok átadódnak a MainActivitybe. A create részt a main fogja meghívni, ami egy shopping itemet vár. Dialogba pedig példányosítottunk egy shopping itemet.

Az új elem létrehozásánál is ugyanúgy átadódik a MainActivitybe az új adatok. Az új adatokat pedig a Dialogban kezeljük, itt tudjuk felvinni új elemet, a handleItemCreate függvény meghívódik majd egy új shopping itemet hoz létre az új adatokkal.

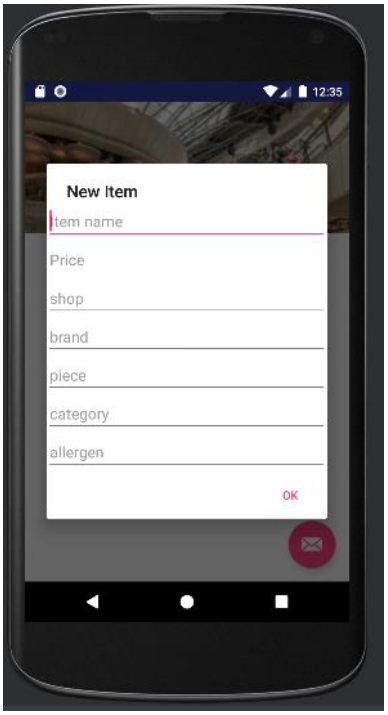
```
private fun handleItemCreate() {
    shoppingItemHandler.shoppingItemCreated(ShoppingItem(
            itemId: null,
            etItem.text.toString(),
            etPrice.text.toString().toInt(),
            bought: false,
        etShop.text.toString(),
        etBrand.text.toString(),
        etPiece.text.toString().toInt(),
        etCategory.text.toString(),
        etAllergen.text.toString()

    ))
}
```

A ShoppingAdapter osztályban még beállítjuk, hogy az új elemek látszódjanak. Lekérjük a shopping item elemek adattagjait.

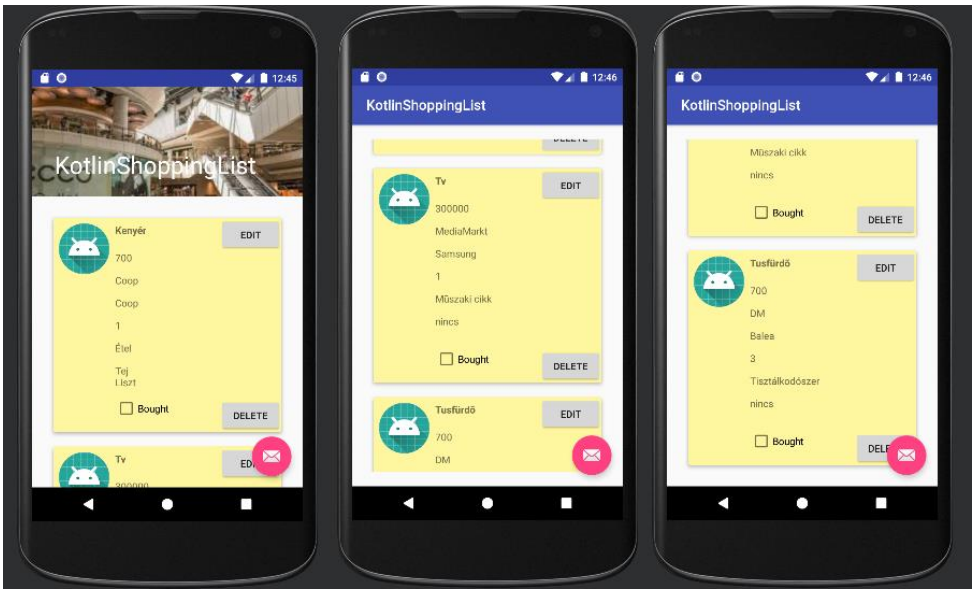
```
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val tvName: TextView = itemView.tvName
        val tvPrice: TextView = itemView.tvPrice
        val cbBought: CheckBox = itemView.cbBought
        val btnDelete: Button = itemView.btnDelete
        val btnEdit: Button = itemView.btnEdit
        val tvShop: TextView=itemView.tvShop
        val tvBrand: TextView=itemView.tvBrand
        val tvPiece: TextView=itemView.tvPiece
        val tvCategory: TextView=itemView.tvCategory
        val tvAllergen: TextView=itemView.tvAllergen
    }
}
```

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    holder.tvName.text = items[position].name
    holder.tvPrice.text = items[position].price.toString()
    holder.cbBought.isChecked = items[position].bought
    holder.tvShop.text=items[position].shop
    holder.tvBrand.text=items[position].brand
    holder.tvPiece.text=items[position].piece.toString()
    holder.tvCategory.text=items[position].category
    holder.tvAllergen.text=items[position].allergen

    holder.btnDelete.setOnClickListener { it: View!
        deleteItem(holder.adapterPosition)
    }

    holder.btnEdit.setOnClickListener { it: View!
        (holder.itemView.context as MainActivity).showEditItemDialog(
                items[holder.adapterPosition])
    }

    holder.cbBought.setOnClickListener { it: View!
        items[position].bought = holder.cbBought.isChecked
        val dbThread = Thread {
            AppDatabase.getInstance(context).shoppingItemDao().updateItem(items[position])
        }
        dbThread.start()
    }
}
```

A lefuttatás után új Card-okat tudtam felvenni a kibővített mezőkkel.




Most már nem csak a termék nevét és árát tudjuk megadni, hanem a bolt nevét, a termék márkáját, azt hogy hány darab terméket szeretnénk, milyen kategóriába tartozik az adott termék, és hogy tartalmaz e valamilyen allergén anyagot amit a vásárláskor majd figyelembe kell venni.




Próbaképpen felvettünk 3 terméket. Kenyeret, tv-t, és tusfürdőt. Mindegyiknél működött a szerkesztés, a törlés is, és a check box-nál is ki tudtam pipálni ha megvettem az adott terméket.