ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Pizza Pro

semestrálna práca

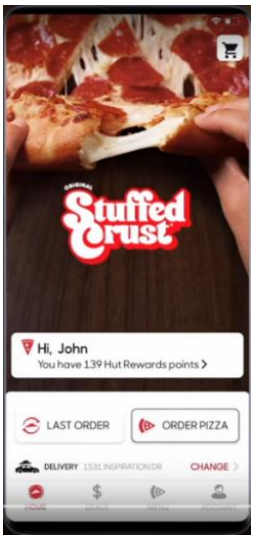
Vypracoval: **Erik Mešina**
Študijná skupina: **5ZYR34**
Predmet: **Vývoj aplikácií pre mobilné zariadenia**
Cvičiaci: **doc. Ing. Patrik Hrkút, PhD.**

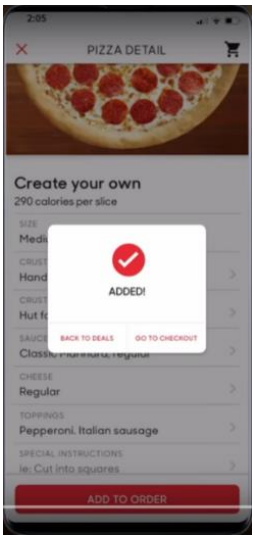
Obsah

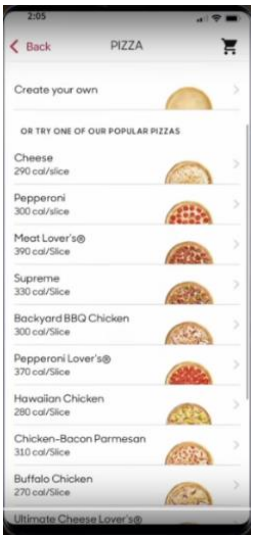
Prehľad aplikácií s podobným zameraním

Pizza Hut

Pizza Hut ponúka najjednoduchší spôsob objednávania svojej obľúbenej pizze, kuracích krídielok, dezertov a ďalších lahôdok. Obsahuje mnohé funkcie ako napríklad selekciu produktov, pridanie príloh a bezkontaktné objednávanie so zaručením rýchleho a bezpečného doručenia.







Zdroj: https://apkpure.com/pizza-hut-food-delivery-ta/com.yum.pizzahut

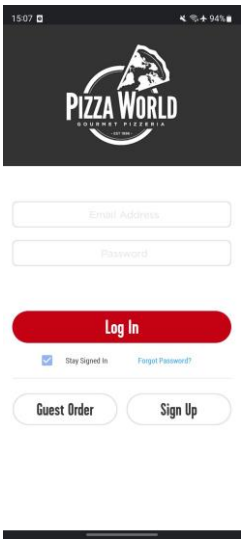
Domino's Pizza Brasil

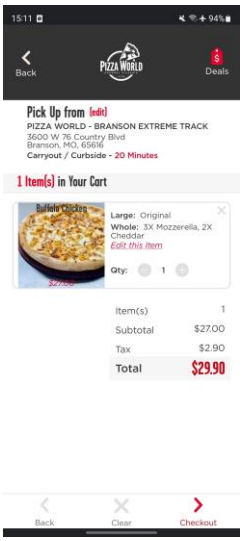
Domino's, jeden z najväčších reťazcov pizze na svete, sprístupňuje väčšinu miest, v ktorých je prítomná služba doručovania. Aplikácia poskytuje ponuku siete pobočiek Domino’s a umožňuje priame platby prostredníctvom aplikácie a sledovanie doručovania v reálnom čase.

Zdroj: https://apkpure.com/domino-s-pizza-brasil/br.com.dominos.mobile

Pizza World

Pizza World je obchodným podnikom, ktorý spája vzrušujúci nový koncept gurmánskej pizze svetovej triedy s rýchlosťou, efektivitou a pohodlím tradičných reštaurácií na donášku pizze. Aplikácia taktiež ponúka možnosť zaregistrovania sa, dáva možnosť výberu z viacerých typov jedál ako aj sprístupnenie online platby.





Zdroj: https://apkpure.com/pizza-world/com.hungerrush.pizzaworldusa

Analýza navrhovanej aplikácie

Aplikácia pre objednávanie pizze je praktickým a efektívnym spôsobom, ako si zákazníci môžu objednať svoju obľúbenú pizzu z pohodlia svojho domova.

Z hľadiska funkcionality: Aplikácia by mala byť pre zákazníkov jednoduchá na používanie a ponúkať všetky potrebné funkcie, ako napríklad prehľad menu, objednávka, spätná väzba,...

Z hľadiska používateľského rozhrania: Rozhranie aplikácie by malo byť intuitívne, prehľadné a atraktívne pre zákazníkov, aby sa s ňou ľahko pracovalo a zákazníci sa v nej dokázali zorientovať.

Z hľadiska spôsobu platby: Aplikácia by mala ponúkať rôzne spôsoby platby, aby boli zákazníci pohodlní pri používaní aplikácie. Medzi bežné spôsoby patrí platba kartou, online bankovníctvom alebo platobnými bránami.

Z hľadiska bezpečnosti: Aplikácia by mala mať zabezpečené dáta zákazníkov, aby sa zabránilo akejkoľvek neoprávnenej manipulácii s informáciami.

Hodnotenie a spätná väzba: Zákazníci by mali mať možnosť hodnotiť aplikáciu a jedlo, čo povedie k neustálej optimalizácii služieb a produktov.

Celkovo by mala byť aplikácia pre objednávanie pizze rýchla, spoľahlivá a jednoduchá na používanie. Ide hlavne o dosiahnutie spokojnosti zákazníkov, čo zabezpečí kladné recenzie, opätovné používanie a odporučenie aplikácie.

Návrh architektúry aplikácie

Prehľad menu: Zákazník by mal byť schopný prehliadať kompletné menu reštaurácie, vrátane cien, popisu a obrázkov. Pizza Pro obsahuje široký sortiment produktov. Pomocou selekcie filtrov si môžete zvoliť presne tú, po ktorej túžite.

Objednávanie: Zákazník by mal mať možnosť vytvoriť si vlastnú objednávku pomocou interaktívneho menu. Pizza Pro má k dispozícii rôzne možnosti pre vytváranie objednávky, ako napríklad voľba veľkosti a zmena počtu kusov.

Platba online formou: Aplikácia by mala umožniť zákazníkom jednoducho a bezpečne zaplatiť za svoju objednávku. Je tu výber z rôznych možností platby, ako napríklad kreditná karta, PayPal, alebo Apple Pay.

Nastavenia účtu: Zákazníci by mali mať možnosť vytvoriť si svoj vlastný účet, kde by mohli spravovať svoje objednávky a získavať výhody.

Recenzie a hodnotenia: Zákazníci by mali mať možnosť zanechať recenzie a hodnotenia na jednotlivé jedlá, aby mohli pomôcť ostatným zákazníkom pri výbere.

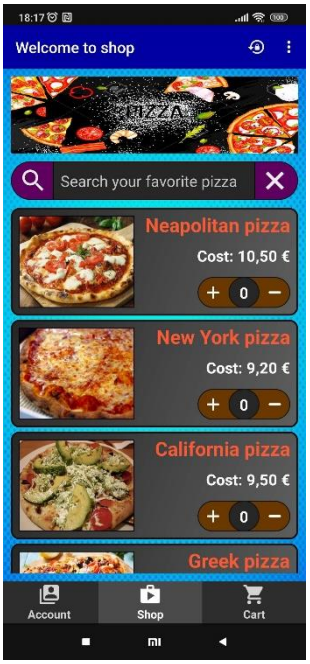
Ukážka návrhu obrazoviek aplikácie

Úvodná obrazovka

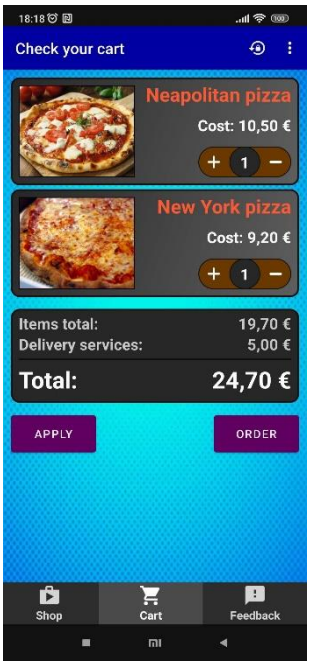
Registrácia

Prihlásenie

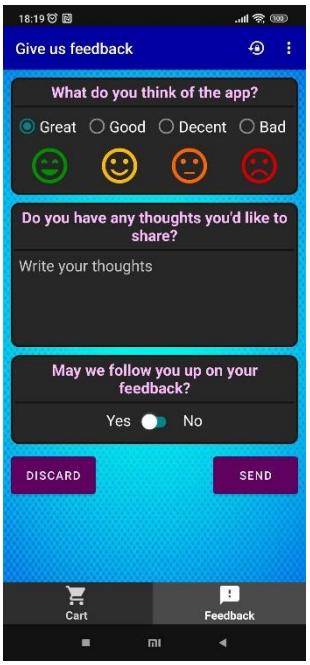
Hlavné menu



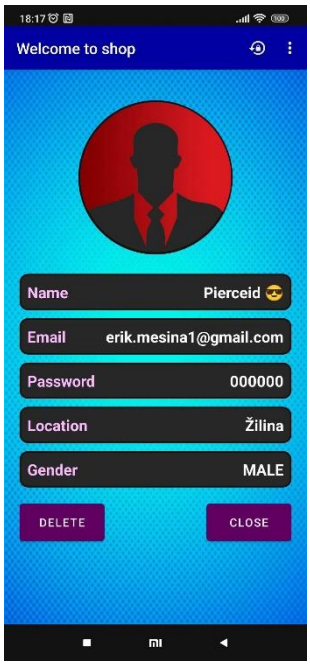
Košík zákazníka



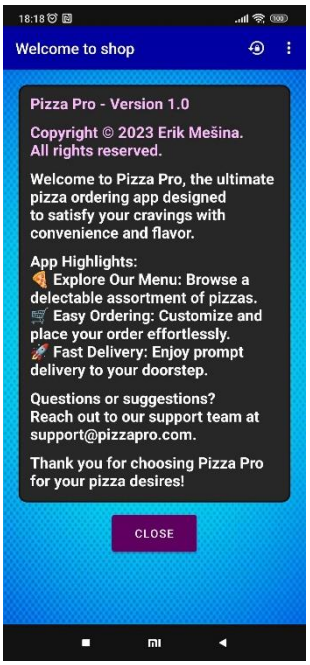
Spätná väzba



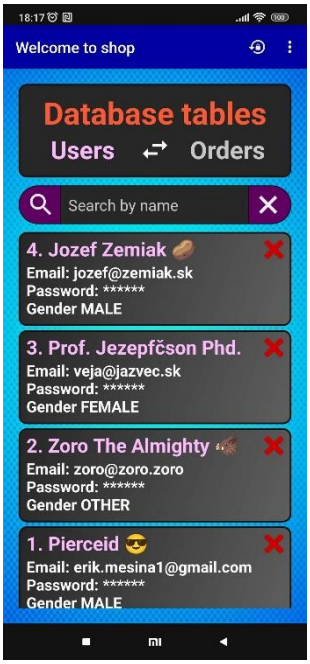
Profil



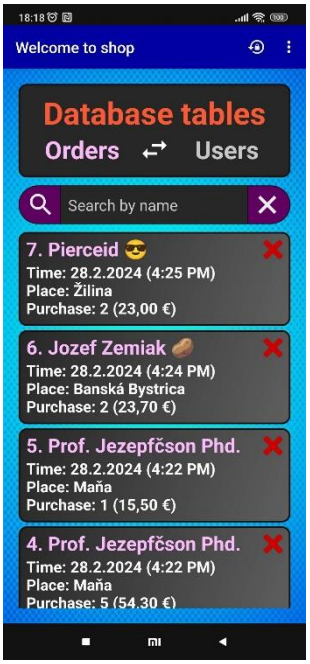
O aplikácií



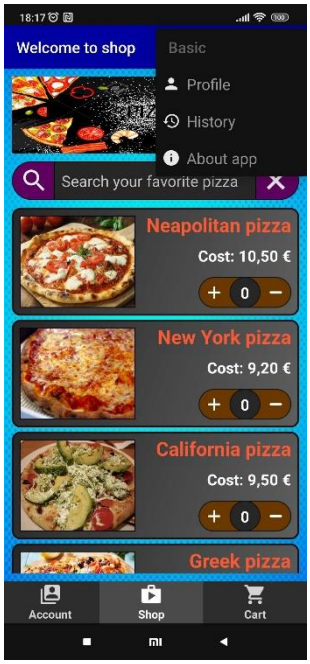
História používateľov



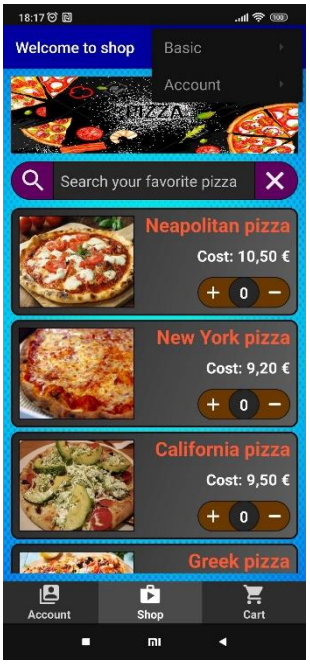
História objednávok



Nastavenia základné



Nastavenia účtu



Fragmenty

IntroFragment

Úvodný fragment, obsahuje len tlačidlo, ktoré používateľa presunie na nasledujúci fragment.

AccountFragment

Fragment, kde si používateľ môže vytvoriť účet. Musí zadať svoje základné informácie, ako sú meno, email, heslo, aktuálnu polohu a pohlavie. Pri nevyplnení niektorej z informácií ho na to systém upozorní. Po úspešnom vyplnení všetkých potrebných informácií môže používateľ pokračovať do hlavného menu, čiže obchodu.

ShopFragment

V tejto časti si zákazník môže vyhľadať a vybrať konkrétne produkty, ako aj si navoliť ich ľubovoľné množstvá. Ak je zákazník spokojný s výberom, môže pokračovať do košíkovej časti.

CartFragment

V tejto časti si používateľ môže skontrolovať výber z predošlého fragmentu, poprípade upraviť množstvá. Ak je zákazník spokojný s výberom môže si dané produkty objednať. Objednávka je vizuálne potvrdená vyskakujúcim oknom. Ak by mal zákazník záujem, môže pokračovať do poslednej hlavnej časti, v ktorej môže podať spätnú väzbu.

FeedbackFragment

Fragment, kde používateľ môže vyjadriť svoju spokojnosť a podať nápady na zlepšenie.

DetailFragment

Fragment, v ktorom sú bližšie popísané jednotlivé produkty (názov, hodnotenie, čas čakania, množstvo kalórií a stručný popis). Dostupný po kliknutí na názov alebo obrázok produktu.

ProfileFragment

Fragment, v ktorom sú uvedené informácie o používateľovi zadané pri vytváraní konta. Dostupný po kliknutí na „Profile“ v nastaveniach (vpravo hore).

HistoryFragment

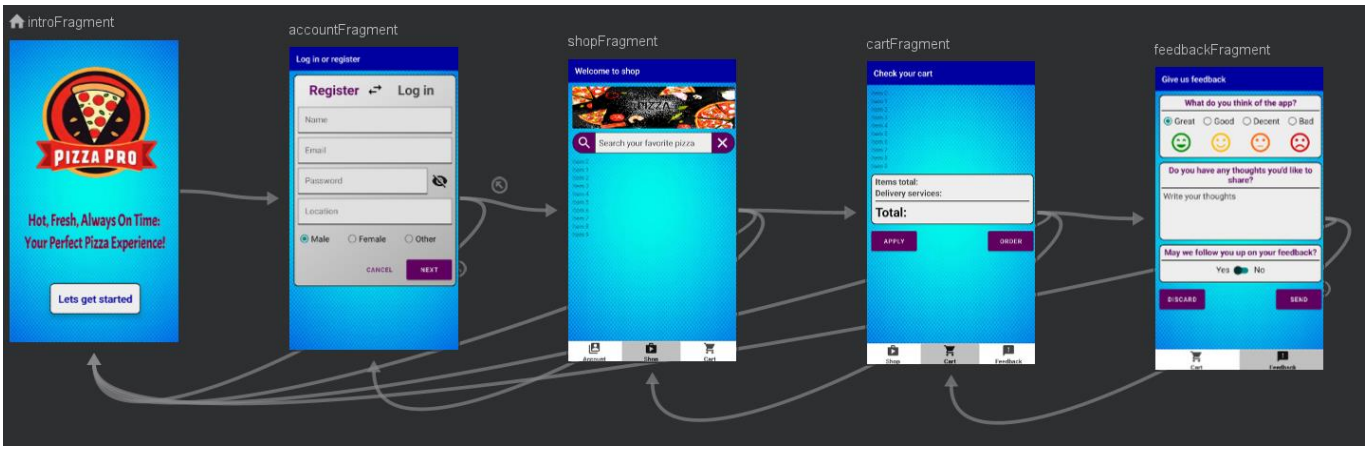
Fragment, v ktorom sú uvedené databázy používateľov a objednávok. Používateľ má možnosť si vyhľadať konkrétnych používateľov a objednávky pomocou zadania mena do vyhľadávacej kolonky. Taktiež má možnosť vymazať účty, stornovať objednávky alebo vymazať celé jednotlivé databázy. Dostupný po kliknutí na „History“ v nastaveniach (vpravo hore).

AboutAppFragment

Fragment, v ktorom sú uvedené informácie o aplikácií (verzia, autorské práva, popis aplikácie,...). Dostupný po kliknutí na „About app“ v nastaveniach (vpravo hore)

Navigačný graf

V aplikácií je využitý 1 navigačný graf, ktorý spravuje navigáciu pre všetky hlavné fragmenty (intro, account, shop, cart, feedback).



Navigácia pre doplňujúce fragmenty

Pre navigáciu k doplňujúcim fragmentom, ako sú detail produktu a profil zákazníka, sa využíva nižšie uvedená statická metóda v triede Util. Taktiež je tu ošetrená duplikácia fragmentov pri opätovnom kliknutí na tú istú možnosť (napr. viac-krát za sebou kliknem na profil).

```
// navigates to an additional fragment
fun navigateToFragment(
    fragmentManager: FragmentManager, fragment: Fragment, bundle: Bundle? = null
) {
    val size = fragmentManager.fragments.size
    if (size > 1) {
        val currentTag = fragmentManager.fragments[size - 1].tag
        val previousTags = fragmentManager.fragments.subList(size - 2, size).map { it.tag }

        if (previousTags.contains(currentTag)) {
            fragmentManager.popBackStack()
        }
    }
    fragment.arguments = bundle
    fragmentManager.beginTransaction().replace(R.id.fragmentContainer, fragment)
        .addToBackStack(null).commit()
}
```

Odhlásenie

Používateľ má možnosť kedykoľvek sa odhlásiť, čiže presunúť sa na úvodné okno. Dostupné po kliknutí na „Log out“ v nastaveniach (vpravo hore).

Adaptéry

Adaptér slúži ako most medzi zdrojom údajov a RecyclerView. Je zodpovedný za vytváranie potrebných zobrazení pre každú položku a prepojenie údajov s týmito zobrazeniami.

PizzaAdapter

- 2 listenery (pre plusButton a MinusButton) upravujúce množstvo produktu
- 2 listenery (pre obrázok a názov pizze) na navigáciu do Detail fragmentu
- funkcie na filtrovanie produktov a inicializáciu zoznamu produktov

```
class PizzaAdapter(
    private val fragmentManager: FragmentManager, private val pizzas: MutableList<Pizza>
) : RecyclerView.Adapter<PizzaAdapter.PizzaViewHolder>() {

    // for recycle view behaviour
    inner class PizzaViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val image: ImageView = itemView.findViewById(R.id.iv_pizza)
        val name: TextView = itemView.findViewById(R.id.tv_name)
        val count: TextView = itemView.findViewById(R.id.tv_itemCount)
        val cost: TextView = itemView.findViewById(R.id.tv_cost)

        private val plusButton: ImageView = itemView.findViewById(R.id.iv_plus)
        private val minusButton: ImageView = itemView.findViewById(R.id.iv_minus)

        init {
            plusButton.setOnClickListener {
                val position = adapterPosition
                if (position != RecyclerView.NO_POSITION && pizzas[position].count < 10) {
                    pizzas[position].count++
                    notifyItemChanged(position)
                }
            }

            minusButton.setOnClickListener {
                val position = adapterPosition
                if (position != RecyclerView.NO_POSITION && pizzas[position].count > 0) {
                    pizzas[position].count--
                    notifyItemChanged(position)
                }
            }
```

```

            image.setOnClickListener {
                val position = adapterPosition
                val pizza = pizzas[position]
                openDetailFragment(pizza)
            }

            name.setOnClickListener {
                val position = adapterPosition
                val pizza = pizzas[position]
                openDetailFragment(pizza)
            }
        }
    }

    // creates the view holder
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PizzaViewHolder {
        return PizzaViewHolder(
            LayoutInflater.from(parent.context).inflate(R.layout.item_pizza, parent, false)
        )
    }

    // binds data from pizzas to views
    override fun onBindViewHolder(holder: PizzaViewHolder, position: Int) {
        val pizza = pizzas[position]

        holder.image.setImageResource(pizza.imageSource)
        holder.name.text = pizza.name
        holder.count.text = pizza.count.toString()
        holder.cost.text =
            String.format("Cost: %s", NumberFormat.getCurrencyInstance().format(pizza.cost))
    }
```

HistoryAdapter

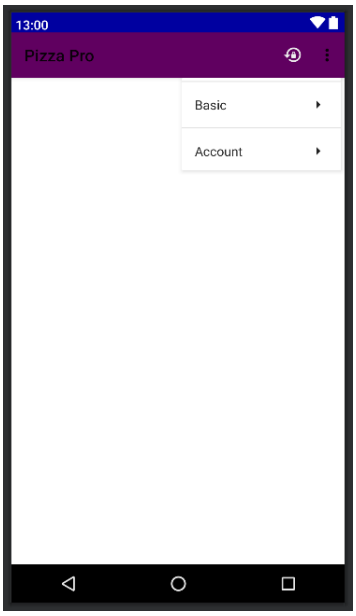
- 1 listener (pre x na kartičke) na odstránenie položky
- funkciu na inicializáciu zoznamu produktov

```
class HistoryAdapter(private val myContext: MyContext) :
    RecyclerView.Adapter<HistoryAdapter.HistoryViewHolder>() {

    private var items: MutableList<Any> = mutableListOf()

    inner class HistoryViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val header: TextView = itemView.findViewById(R.id.tv_header)
        val body: TextView = itemView.findViewById(R.id.tv_body)

        private val xButton: ImageView = itemView.findViewById(R.id.btn_x)

        init {
            xButton.setOnClickListener {
                var alertType = ""
                var runnable = { }

                when (myContext.type) {
                    "users" -> {
                        alertType = "remove_user"
                        runnable = {
                            val position = adapterPosition
                            val user = items[position] as User
                            runBlocking { myContext.myViewModel.removeUser(user) }
                            notifyItemRemoved(position)
                        }
                    }
                    "orders" -> {
                        alertType = "cancel_order"
                        runnable = {
                            val position = adapterPosition
                            val order = items[position] as Order
                            runBlocking { myContext.myViewModel.removeOrder(order) }
```

```
                when (myContext.type) {
                    "users" -> {
                        alertType = "remove_user"
                        runnable = {
                            val position = adapterPosition
                            val user = items[position] as User
                            runBlocking { myContext.myViewModel.removeUser(user) }
                            notifyItemRemoved(position)
                        }
                    }
                    "orders" -> {
                        alertType = "cancel_order"
                        runnable = {
                            val position = adapterPosition
                            val order = items[position] as Order
                            runBlocking { myContext.myViewModel.removeOrder(order) }
                            notifyItemRemoved(position)
                        }
                    }
                    else -> {}
                }

                Util.createAlertDialog(
                    myContext.activity,
                    alertType,
                    runnable,
                    myContext.layoutInflater,
                    myContext.parentView
                )
            }
        }
    }
```

Nastavenia – menu

Slúži na poskytnutie používateľom spôsobu, ako prechádzať medzi rôznymi časťami aplikácie. Menu tejto aplikácie obsahuje 2 časti: základnú a účtovú. Základná časť obsahuje prepojenie na profil používateľa, históriu objednávok a informácie o aplikácií, zatiaľ čo účtová ponúka možnosť odhlásenia. Súčasťou menu je aj priamo viditeľná možnosť zamknúť/odomknúť rotáciu obrazovky.



Room Databáza

Room databáza pre používateľov a objednávky je komponent v Android aplikácii, ktorý umožňuje efektívne uchovávanie a spravovanie údajov. Poskytuje prostredie na efektívne ukladanie, získavanie a aktualizovanie dát v rámci Android aplikácie. Taktiež zabezpečuje, že dáta sú konzistentné, spoľahlivé a je s nimi možné pracovať na hlavnom vlákne alebo asynchrónne podľa potreby.

User a Order

Entity, ktoré uchovávajú základné informácie o korešpondujúcich objektoch.

```
package com.example.pizza_pro.database

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
import com.example.pizza_pro.options.Gender

@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) var id: Long = 0L,
    @ColumnInfo(name = "name") var name: String = "",
    @ColumnInfo(name = "email") var email: String = "",
    @ColumnInfo(name = "password") var password: String = "",
    @ColumnInfo(name = "location") var location: String = "",
    @ColumnInfo(name = "gender") var gender: Gender = Gender.OTHER
)

@Entity(tableName = "order_table")
data class Order(
    @PrimaryKey(autoGenerate = true) var id: Long = 0L,
    @ColumnInfo(name = "name") var name: String = "",
    @ColumnInfo(name = "time") var time: String = "",
    @ColumnInfo(name = "place") var place: String = "",
    @ColumnInfo(name = "items") var items: Int = 0,
    @ColumnInfo(name = "cost") var cost: String = ""
)
```

MyDatabase

Abstraktná trieda, ktorá deklaruje tabuľky v databáze. Tabuľky obsahujú stĺpce, ktoré reprezentujú jednotlivé údaje z entít User a Order.

```kotlin
package com.example.pizza_pro.database

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class, Order::class], version = 3, exportSchema = false)
abstract class MyDatabase : RoomDatabase() {

    abstract val dao: MyDao

    companion object {

        @Volatile
        private var INSTANCE: MyDatabase? = null

        fun getDatabase(context: Context): MyDatabase {
            synchronized(this) {
                var instance = INSTANCE
                if (instance == null) {
                    instance = Room.databaseBuilder(
                        context.applicationContext,
                        MyDatabase::class.java,
                        "my_database"
                    ).fallbackToDestructiveMigration().build()
                    INSTANCE = instance
                }
                return instance
            }
        }
    }
}
```

MyDao

Rozhranie, ktoré obsahuje SQL príkazy pre prístup a manipuláciu s údajmi v databáze. Obsahuje metódy na vkladanie, aktualizovanie a odstránenie (používateľa, objednávky), odstránenie všetkých používateľov a objednávok v databáze, filtrovanie používateľov a objednávok a metódy na prístup ku všetkým používateľom a objednávkam v databáze.

```kotlin
@Dao
interface MyDao {
    @Upsert
    suspend fun upsertUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)

    @Upsert
    suspend fun upsertOrder(order: Order)

    @Delete
    suspend fun deleteOrder(order: Order)

    @Query("DELETE FROM user_table")
    suspend fun clearAllUsers()

    @Query("DELETE FROM order_table")
    suspend fun clearAllOrders()

    @Query("SELECT * FROM user_table WHERE name = :name OR email = :email LIMIT 1")
    fun getUser(name: String, email: String): User?

    @Query("SELECT * FROM user_table WHERE name LIKE '%' || :regex || '%' ORDER BY id DESC")
    fun getFilteredUsers(regex: String): LiveData<MutableList<User>>

    @Query("SELECT * FROM order_table WHERE name LIKE '%' || :regex || '%' ORDER BY id DESC")
    fun getFilteredOrders(regex: String): LiveData<MutableList<Order>>

    @Query("SELECT * FROM user_table ORDER BY id DESC")
    fun getAllUsers(): LiveData<MutableList<User>>

    @Query("SELECT * FROM order_table ORDER BY id DESC")
    fun getAllOrders(): LiveData<MutableList<Order>>
}
```

MyRepository

Účelom tejto triedy je abstrahovať zdroj údajov, ako je databáza alebo lokálne úložisko, od zvyšku aplikácie. Poskytuje čisté a konzistentné API na prístup a správu údajov, čo uľahčuje údržbu a testovanie aplikácie.

```
package com.example.pizza_pro.database

import androidx.lifecycle.LiveData

class MyRepository(private val dao: MyDao) {

    var allUsers: LiveData<MutableList<User>> = dao.getAllUsers()
    var allOrders: LiveData<MutableList<Order>> = dao.getAllOrders()

    suspend fun addUser(user: User) = dao.upsertUser(user)

    suspend fun addOrder(order: Order) = dao.upsertOrder(order)

    suspend fun removeUser(user: User) = dao.deleteUser(user)

    suspend fun removeOrder(order: Order) = dao.deleteOrder(order)

    suspend fun clearAllUsers() = dao.clearAllUsers()

    suspend fun clearAllOrders() = dao.clearAllOrders()

    fun getUser(name: String, email: String) = dao.getUser(name, email)

    fun getFilteredOrders(regex: String) = dao.getFilteredOrders(regex)

    fun getFilteredUsers(regex: String) = dao.getFilteredUsers(regex)
}
```

MyViewModel

Trieda, ktorá slúži na komunikáciu medzi repozitárom a používateľským rozhraním.

```
class MyViewModel(application: Application) : AndroidViewModel(application) {

    private val repository: MyRepository
    var users: LiveData<MutableList<User>>
    var orders: LiveData<MutableList<Order>>
    var user: User?

    init {
        val dao = MyDatabase.getDatabase(application).dao
        repository = MyRepository(dao)
        orders = repository.allOrders
        users = repository.allUsers
        user = null
    }

    fun addUser(user: User) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.addUser(user)
        }
    }

    fun addOrder(order: Order) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.addOrder(order)
        }
    }

    fun removeUser(user: User) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.removeUser(user)
        }
    }

    fun removeOrder(order: Order) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.removeOrder(order)
        }
    }

    fun clearAllUsers() {
        viewModelScope.launch(Dispatchers.IO) {
            repository.clearAllUsers()
        }
    }

    fun clearAllOrders() {
        viewModelScope.launch(Dispatchers.IO) {
            repository.clearAllOrders()
        }
    }

    fun getUser(name: String, email: String) {
        viewModelScope.launch(Dispatchers.IO) {
            user = repository.getUser(name, email)
        }
    }

    fun getFilteredUsers(regex: String) {
        users = repository.getFilteredUsers(regex)
    }

    fun getFilteredOrders(regex: String) {
        orders = repository.getFilteredOrders(regex)
    }
}
```

Použité zdroje

- https://developer.android.com/teach#for-instructors-teaching-a-course
- https://www.geeksforgeeks.org/room-database-with-kotlin-coroutines-in-android/
- https://www.ezcater.com/lunchrush/office/most-popular-types-of-pizza-around-country/
- https://www.tasteatlas.com/50-most-popular-pizzas-in-the-world
- https://www.fileformat.info/info/unicode/char/search.htm
- https://www.javatpoint.com/kotlin-android-alertdialog
- https://github.com/DanielMartinus/Konfetti