ŽILINSKÁ UNIVERZITA
V ŽILINE

Semestrálna práca

BikeZone

E-Shop bicyklov

Adam Staňo

5ZYR34

Obsah

Popis a analýza riešeného problému

Špecifikácia zadania, definovanie problému

- Úvod

BikeZone je moderný a efektívny eshop, ktorý umožňuje zákazníkom nakupovať bicykle a príslušenstvo jednoducho a pohodlne. Cieľom tejto aplikácie je poskytnúť používateľom príjemný a intuitívny nákupný zážitok, ktorý zahŕňa všetky potrebné funkcie na vyhľadávanie, porovnávanie a objednávanie produktov.

- Definovanie problému

S rastúcim počtom používateľov mobilných zariadení sa zvyšuje aj dopyt po efektívnych a užívateľsky prívetivých mobilných aplikáciách pre online nakupovanie. Tradičné webové stránky často neponúkajú optimálny zážitok pre mobilných používateľov, čo môže viesť k frustrácii a strate potenciálnych zákazníkov. BikeZone si kladie za cieľ vyriešiť tento problém vytvorením špičkovej mobilnej aplikácie, ktorá bude poskytovať:

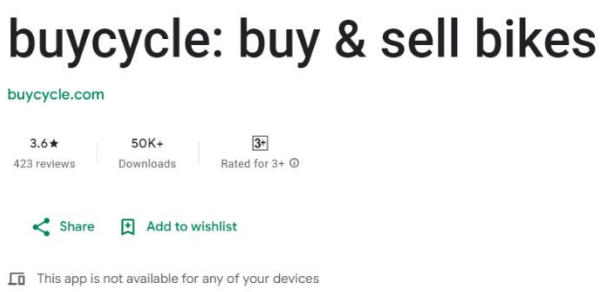
- Dostupnosť

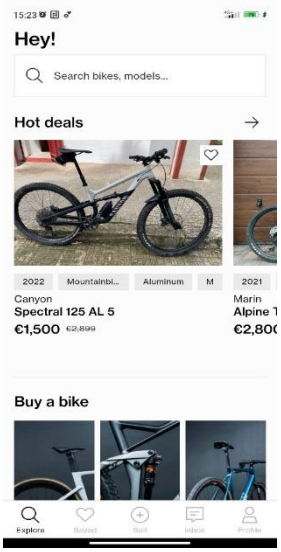
Výstupom projektu bude plne funkčná mobilná aplikácia pre platformu Android, ktorá bude ponúkať všetky vyššie uvedené funkcie a poskytovať používateľom najlepší možný nákupný zážitok pri nákupe bicyklov a príslušenstva.

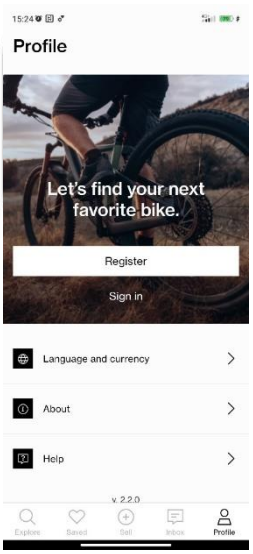
Spracovanie prehľadu dostupných aplikácií podobného zamerania

- buycycle: buy & sell bikes

Táto e-shopová aplikácia je určená na nákup bicyklov a príslušenstva. Umožňuje zákazníkom prezeranie širokej ponuky bicyklov rôznych typov, veľkostí a značiek. Okrem toho môžu zákazníci vyhľadávať produkty podľa ich preferencií a potrieb.



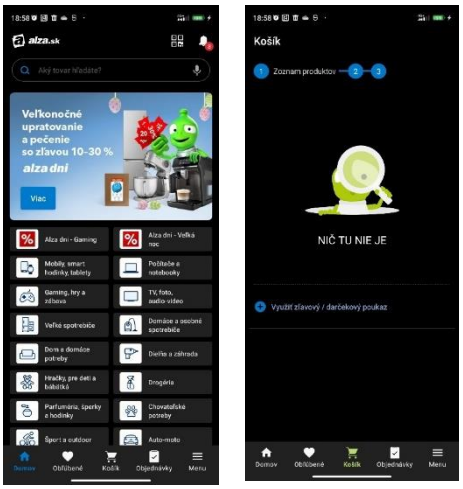




Zdroj : [buycycle by buycycle.com]

- Alza.sk by Alza.cz a.s.

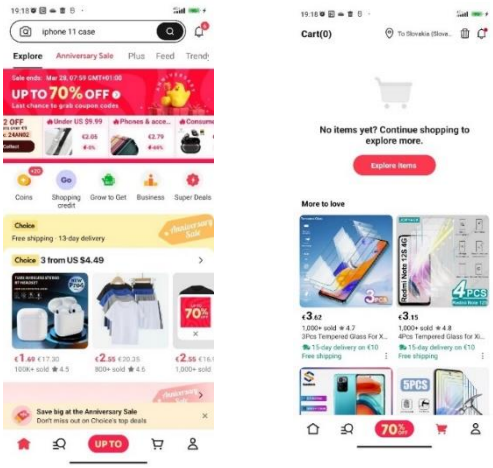
Alza.sk aplikácia na Android je podobná aplikácii zameranej na predaj bicyklov v tom, že ponúka široký výber produktov a možnosti platby, avšak s rozdielnym sortimentom.



Zdroj : **Alza.sk**

- AliExpress by Alibaba Mobile

AliExpress aplikácia je využívaná na online nakupovanie a ponúka obrovský výber produktov. Používatelia majú možnosť si vyhľadať konkrétny produkt, pridať ho do košíka a objednať si ho.



Zdroj : **AliExpress**

Analýza navrhovanej aplikácie

Efektívny nástroj pre nákup bicyklov: BikeZone eshop dáva možnosť nákupu bicyklov jednoducho a efektívne.

Interaktívne možnosti prehľadávania: Aplikácia by mala ponúkať interaktívne možnosti prehľadávania, vrátane možnosti filtrovania podľa typu bicykla, veľkosti, značky atď.

Jednoduché a prehľadné rozhranie: Rozhranie aplikácie by malo byť jednoduché a prehľadné, aby sa zákazníci mohli ľahko orientovať a nájsť si požadovaný bicykel.

Celkový dizajn: Dizajn aplikácie by mal byť príťažlivý a moderný, zabezpečujúci príjemný nákupný zážitok.

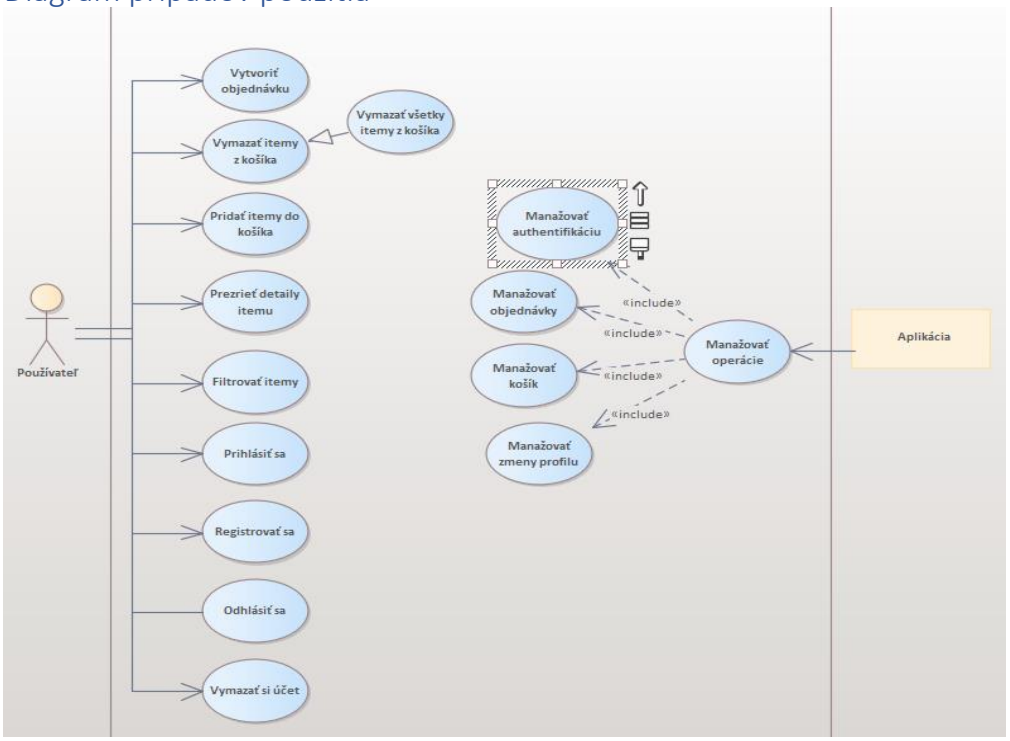
Prísup k svojím objednávkam: Používatelia aplikácie BikeZone môžu jednoducho sledovať a spravovať svoje objednávky. Aplikácia umožňuje prehliadať históriu objednávok, sledovať aktuálny stav objednávky a prijímať notifikácie o zmenách stavu objednávky.

Zabezpečenie dát: Aplikácia by mala mať zabezpečené dáta zákazníkov, aby sa zabránilo akejkoľvek neoprávnenej manipulácii s informáciami.

Prístup k produktom a popisom : BikeZone ponúka detailné popisy produktov vrátane technických špecifikácií a vysoko kvalitných obrázkov. Používatelia majú možnosť porovnávať produkty a získať všetky potrebné informácie pre správny výber bicykla.

Aplikácia BikeZone by mala byť spoľahlivá a užívateľsky prívetivá, čo zabezpečí pohodlný a jednoduchý nákup bicyklov.

Diagram prípadov použitia



Návrh architektúry aplikácie

Svojím zameraním na poskytovanie širokej škály bicyklov a príslušenstva, spolu s intuitívnym používateľským rozhraním, umožňuje zákazníkom pohodlné a bezproblémové nakupovanie.

Aplikácia sa skladá z:

1. ***Prehľadná prezentácia produktov:*** Aplikácia bude poskytovať prehľadný a atraktívny spôsob prezentácie dostupných bicyklov vrátane ich technických parametrov, obrázkov a popisov.

2. ***Filtre na vyhľadávanie:*** Zákazníci budú mať možnosť jednoducho prehľadávať sortiment podľa kategórií, ako sú napríklad horské bicykle, cestné bicykle, elektrické bicykle atď., a taktiež použiť filtre na presné nastavenie ich preferencií.

3. ***Košík a objednávky:*** Aplikácia bude umožňovať zákazníkom pridávať produkty do košíka a jednoducho uskutočňovať objednávky s rôznymi možnosťami platby a doručenia.

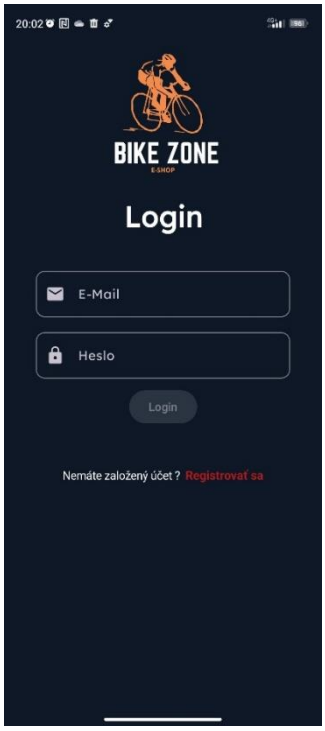
4. ***Osobné účty:*** Zákazníci bude mať možnosť vytvoriť si osobný účet, kde by mohli sledovať svoje predchádzajúce objednávky, upravovať svoje údaje a spravovať svoje preferencie.

5. ***Notifikácie pri rôznych akciach:*** Aplikácia pushuje notifikácie v prípade vytvorenia objednávky, vymazania objednávky a vymazania účtu

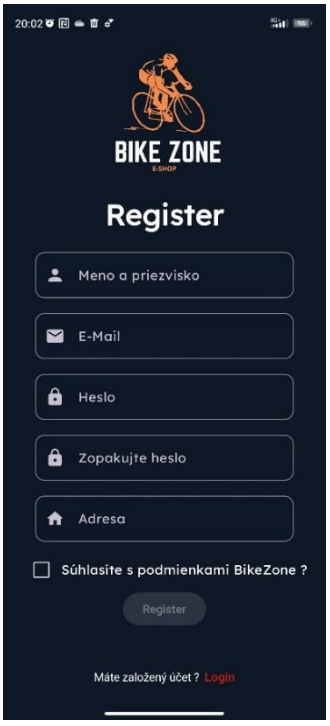
6. ***Zapamätanie predošlých aktivít:*** Aplikácia si zapamätáva aktivity používateľa, produkty v košíku, autentifikáciu...

Ukážka návrhu obrazoviek aplikácie

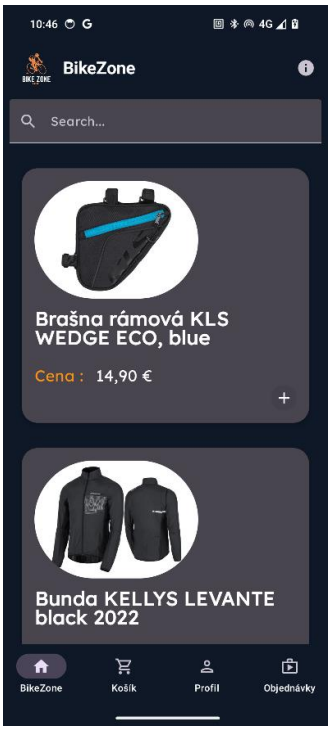
LoginScreen



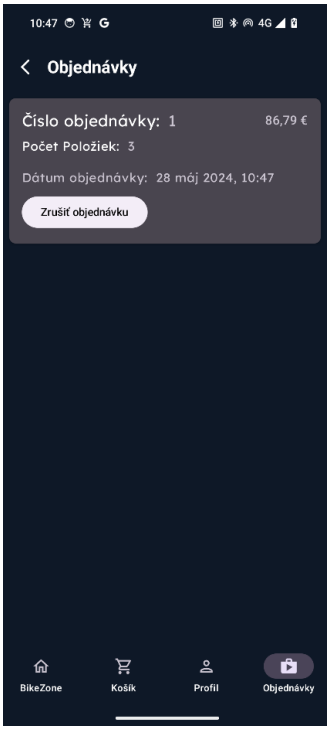
RegisterScreen



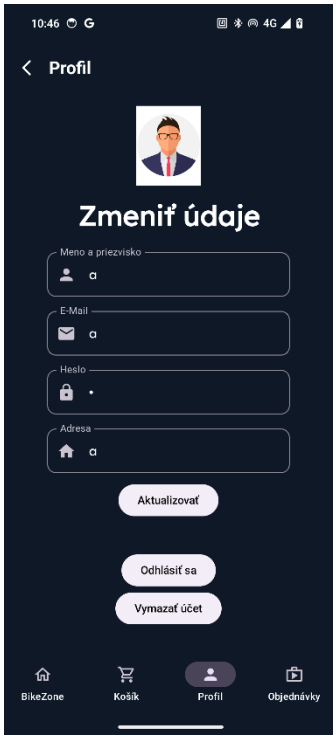
HomeScreen



OrderScreen

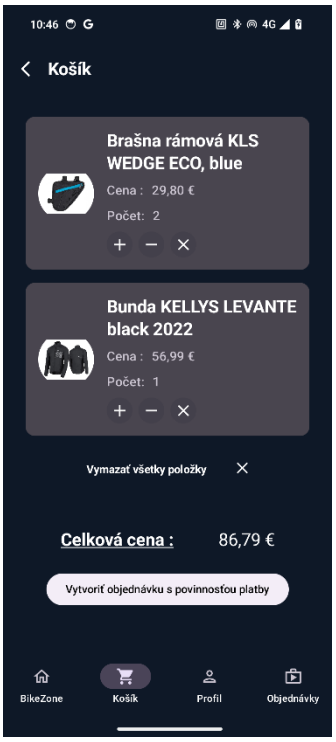


ProfileScreen



AboutScreen

CartScreen



ItemDetailScreen

Popis Implementácie

Obrazovky

HomeScreen

Obrazovka, v ktorej sa **zobrazujú všetky položky** vo forme Card komponentu. Zároveň je možné produkt vložiť do košíka.

ItemDetailScreen

Obrazovka, v ktorej sa zobrazujú **detaily konkrétnej zvolenej položky**.

CartScreen

Obrazovka, v ktorej sa zobrazujú **položky v košíku** zvolené v homeScreen. S položkami môžeme manipulovať (zmenšovať počet, zväčšovať počet, odstrániť položku, odstrániť všetky položky z košíka), pričom sa aktualizuje celková cena košíku a zároveň cena produktu. Po dôkladnom zvážení môžeme vytvoriť objednávku.

OrderScreen

Obrazovka, ktorá **zobrazuje všetky objednávky**, ktoré patria aktuálne prihlásenému používateľovi.

AboutScreen

Časť aplikácie, v ktorej sú uvedené **informácie o aplikácií**

ProfileScreen

Obrazovka, v ktorej sú uvedené **informácie o používateľovi** zadané pri vytváraní konta. Tu si môže používateľ **zmeniť osobné údaje**, ako aj **odhlásiť sa** alebo **odstrániť svoj účet**.

LoginScreen

Obrazovka, kde sa môže **používateľ prihlásiť**. Pri vyplnení nesprávnych informácii aplikácia upozorní používateľa. Po úspešnom vyplnení všetkých potrebných informácií môže používateľ pokračovať do aplikácie, čiže obchodu.

RegisterScreen

Obrazovka, kde si používateľ môže **vytvoriť účet**. Systém kontroluje, či používateľ zadal email, ktorý je unikátny, ak nie upozorní ho. Takisto je potrebné zadať všetky informácie, inak nie je umožnené vytvoriť účet. Po úspešnej registrácii sa používateľ môže prihlásiť.

ViewModel

Takmer každej obrazovke prislúcha vlastného spracovateľa udalostí ViewModel (pri kliknutí na tlačidlo, zmenu hodnôt atribútov,...). V niektorých prípadoch bolo nutné pracovať s databázou, preto bol vytvorený poskytovateľ modelu pohľadu (**AppViewModelProvider**) pre dodatočnú inicializáciu repozitáru.

```
adams
object AppViewModelProvider {
    val Factory = viewModelFactory { this: InitializerViewModelFactoryBuilder
        initializer { this: CreationExtras
            LoginViewModel(application().container.userRepository)
        }

        initializer { this: CreationExtras
            RegisterViewModel(application().container.userRepository)
        }

        initializer { this: CreationExtras
            AuthViewModel(application().container.userRepository)
        }
        initializer { this: CreationExtras
            ProfileViewModel(
                application().container.userRepository,
                application().container.cartRepository,
                application().container.orderRepository,
            )
        }
        initializer { this: CreationExtras
            HomeViewModel(
                application().container.itemRepository,
                application().container.cartRepository,
                application().container.userRepository
            )
        }
        initializer { this: CreationExtras
            ItemDetailsViewModel(
                this.createSavedStateHandle(),
                application().container.itemRepository
            )
        }
        initializer { this: CreationExtras
            CartViewModel(
                application().container.cartRepository,
                application().container.itemRepository,
                application().container.orderRepository,
                application().container.userRepository
```

Navigácia

- Nested Navigation Graph
  Navigačný graf aplikácie je rozdelený na dve časti:

AppNavigationGraph a **AuthNavigationGraph**. Prepínanie medzi týmito grafmi je zabezpečené pomocou **routov**. Ak je používateľ už prihlásený, aplikácia automaticky prepne na AppGraph. V opačnom prípade, pri spustení aplikácie, je predvolený graf AuthGraph.

```
adams +1
@Composable
fun BikeZoneNavHost(
    navController: NavHostController , auth: Boolean
) {
    val startDestination = if (!auth) Routes.AuthRoute.route else Routes.AppRoute.route
    NavHost(
        navController = navController,
        startDestination = startDestination,
    ) { this: NavGraphBuilder
        setupAuthGraph(navController)
        setupAppGraph(navController)

    }
}
```

Na definovanie **routov** obrazoviek bol vytvorený interface **Destination**. Každá obrazovka má svoj vlastný objekt, napríklad **{"name of screen"}Destination**, ktorý implementuje toto rozhranie.

```
adams
interface AppNavigationDestination {
    val route: String
    val titleRes: Int
    val selectedIcon: ImageVector
    val unselectedIcon : ImageVector
}
```

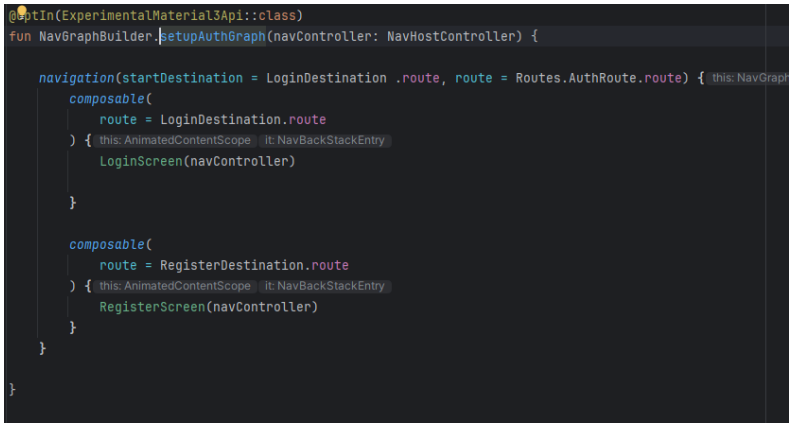
```
interface AuthNavigationDestinantion {
    val route: String
    val titleRes: Int
}
```

```
object HomeDestination : AppNavigationDestination{
    override val route: String = "nav_home"
    override val titleRes: Int = "BikeZone"
    override val selectedIcon: ImageVector = Icons.Filled.Home
    override val unselectedIcon: ImageVector = Icons.Outlined.Home

}

adams
object CartDestination : AppNavigationDestination {
    override val route: String = "cart"
    override val titleRes: Int = "Košík"
    override val selectedIcon = Icons.Filled.ShoppingCart
    override val unselectedIcon = Icons.Outlined.ShoppingCart
}
adams
object ItemDetailsDestination : AppNavigationDestination {
    override val route = "item_details"
    override val titleRes = "Detaily"
    override val selectedIcon: ImageVector = Icons.Filled.Home
    override val unselectedIcon: ImageVector = Icons.Outlined.Home
    const val ITEM_ID_ARG = "itemId"
    val routeWithArgs = "$route/{$ITEM_ID_ARG}"
}

adams
object ProfileDestination : AppNavigationDestination {
    override val route: String = "profile"
    override val titleRes: Int = "Profil"
    override val selectedIcon = Icons.Filled.Person
    override val unselectedIcon = Icons.Outlined.Person
}

adams
object AboutDestination : AppNavigationDestination {
    override val route: String = "about"
    override val titleRes: Int = "O nás"
    override val selectedIcon = Icons.Filled.Info
    override val unselectedIcon = Icons.Outlined.Info
```

- AuthNavigationGraph

Udelenie prístup do nasledujúcich obrazoviek:

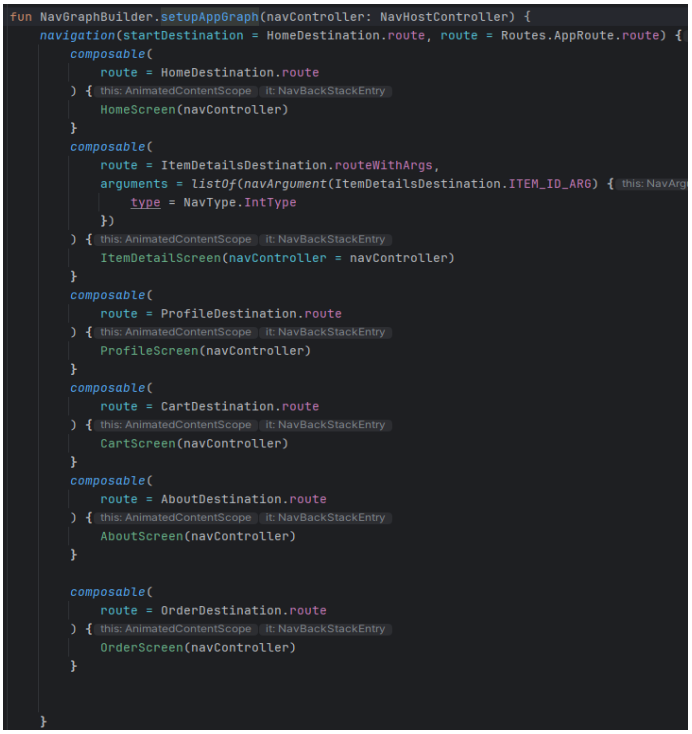
1. LoginScren
2. RegisterScreen



- AppNavigationGraph

Udelenie prístup do obrazoviek:

1. HomeScreen
2. ItemDetailScreen s argumentmi (ITEM_ID_ARG : Int)
3. ProfileScreen
4. CartScreen
5. AboutScreen
6. OrderScreen



Notifikácie

Pri implementovaní bolo potrebné udeliť oprávnenie od užívateľa na vytvorenie notifikácii.

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

Metóda na **request oprávnenia**:

```
adams
@RequiresApi(Build.VERSION_CODES.TIRAMISU)
fun requestNotificationPermission(activity: Activity) {
    ActivityCompat.requestPermissions(
        activity,
        arrayOf(Manifest.permission.POST_NOTIFICATIONS),
        REQUEST_CODE_NOTIFICATION_PERMISSION
    )
}
```

```
adams
@RequiresApi(Build.VERSION_CODES.TIRAMISU)
fun checkNotificationPermission(context: Context): Boolean {
    return ContextCompat.checkSelfPermission(
        context,
        Manifest.permission.POST_NOTIFICATIONS
    ) == PackageManager.PERMISSION_GRANTED
}
```

- Notfikácia pri vymazaní účtu
- Notifikácia pri vytvorení objednávky
- Notifikácia pri vymazaní objednávky

Lokálna Databáza Room

Room databáza pre **používateľov**, **objednávok**, **položiek v košíku** a **položiek** je kľúčovou súčasťou Android aplikácie, ktorá umožňuje efektívne uchovávanie a správu údajov. Poskytuje prostredie na efektívne ukladanie, získavanie a aktualizáciu dát v rámci aplikácie. Zabezpečuje, že dáta sú konzistentné a spoľahlivé, a umožňuje prácu s nimi buď na hlavnom vlákne, alebo asynchrónne podľa potreby.

- Entities
    a) User
    b) Item
    c) CartItem
    d) Order

```
Kotvaldo +1
@Entity(tableName = "orders",
    foreignKeys = [ForeignKey(
        entity = User::class,
        parentColumns = ["user_id"],
        childColumns = ["user_id"],
        onDelete = ForeignKey.CASCADE
    )])


data class Order(
    @PrimaryKey(autoGenerate = true) @ColumnInfo(name = "order_id") val id: Int = 0,
    @ColumnInfo(name = "price") val price: Double,
    @ColumnInfo(name = "item_count") val count: Int,
    @ColumnInfo(name = "user_id") val userId: Int,
    @ColumnInfo(name = "order_date") val date: Date
)
```

- Daos
    a) UserDao
    b) ItemDao
    c) CartItemDao
    d) OrderDao

```
adams
@Dao
interface UserDao {

    adams
    @Query("SELECT * from users WHERE auth = :auth")
    fun getAuthUser(auth: Boolean): Flow<User?>
    adams
    @Query("SELECT * from users WHERE user_id = :id")
    fun getUserById(id: Int): Flow<User>
    adams
    @Query("SELECT * from users WHERE email = :email and password = :password")
    fun getUserByEmailAndPassword(email:String, password:String): Flow<User?>
    adams
    @Query("SELECT * from users WHERE email = :email")
    fun getUserByEmail(email:String): Flow<User?>
    adams
    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insert(user: User)

    adams
    @Update
    suspend fun update(user: User)

    adams
    @Delete
    suspend fun delete(user: User)
}
```

- Repositories

AppContainer

Slúži na implementáciu kontejneru pre rôzne repozitáre v Android aplikácii. Tento kontejner sa stará o poskytovanie **inštancií repozitárov**, ktoré sú potrebné pre prácu s databázou.

```kotlin
interface AppContainer {
    val itemRepository : ItemRepository
    val userRepository : UserRepository
    val cartRepository : CartRepository
    val orderRepository : OrderRepository
}

adams
class AppDataContainer(private val context: Context) : AppContainer {
    override val itemRepository: ItemRepository by lazy {
        OfflineItemRepozitory(BikeZoneDatabase.getDatabase(context).itemDao())
    }
    override val userRepository: UserRepository by lazy {
        OfflineUserRepository(BikeZoneDatabase.getDatabase(context).userDao())
    }
    override val cartRepository: CartRepository by lazy {
        OfflineCartRepository(BikeZoneDatabase.getDatabase(context).cartDao())
    }
    override val orderRepository: OrderRepository by lazy {
        OfflineOrderRepository(BikeZoneDatabase.getDatabase(context).orderDao())
    }

}
```

- Database

Abstraktná trieda, ktorá deklaruje tabuľky v databáze, obsahuje stĺpce reprezentujúce jednotlivé údaje z rôznych entít. Obsahuje metódu na získanie existujúcej inštancie databázy alebo vytvorenie novej inštancie, ak databáza ešte neexistuje.

```kotlin
@Database(entities = [Item::class, User::class, CartItem::class, Order::class], version = 1, exportSchema = false)
@TypeConverters(Converters::class)
abstract class BikeZoneDatabase : RoomDatabase() {
    adams
    abstract fun itemDao(): ItemDao
    adams
    abstract fun userDao(): UserDao
    adams
    abstract fun cartDao(): CartDao
    adams
    abstract fun orderDao(): OrderDao

    adams
    companion object {
        @Volatile
        private var INSTANCE: BikeZoneDatabase? = null

        adams
        fun getDatabase(context: Context): BikeZoneDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    BikeZoneDatabase::class.java,
                     name: "bikezone_dtb"
                ).build()
                INSTANCE = instance
                instance ^synchronized
            }
        }
    }
}
```

Použité zdroje

Android Basics:
https://developer.android.com/courses/android-basics-compose/course

Notifikácie:
https://www.youtube.com/watch?v=bHlLYhSrXvc
https://www.youtube.com/watch?v=gn0isU0Wa_g&list=PLSrm9z4zp4mFttjku-3wiRkPH1lDRQLYy

Nested Graph
https://developer.android.com/guide/navigation/design/nested-graphs
https://www.youtube.com/watch?v=FIEnIBq7Ups

Android UI Components Tutorials:
https://www.youtube.com/watch?v=cDabx3SjuOY&list=PLQkwcJG4YTCSpJ2NLhDTHhi6XBNfk9WiC

Images and resources:
https://www.bike-eshop.cz/
https://bicykle-eshop.sk/
https://www.alza.sk/