Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №6

«Верстка и навигация экранов на языке kotlin»

Выполнил:
студент группы РТ5-31Б:
Паншин М.В.

Проверил:
преподаватель кафедры ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

Постановка задачи

Необходимо реализовать несколько экранов и настроить навигацию для перехода между экранами, используя Compose.

Текст программы

Файл MainActivity.kt

```
package com.example.a3kotlin

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.ui.Modifier
import androidx.navigation.compose.rememberNavController

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            val navController = rememberNavController()
            Scaffold(
                topBar = { TopBar() },
                bottomBar = { BottomNavigationBar(navController) }
            ) { paddingValues ->
                Box(modifier = Modifier.padding(paddingValues)) {
                    Navigation(navController)
                }
            }
        }
    }
}
```

Файл AdressPickScreen.kt

```
package com.example.a3kotlin


import androidx.compose.animation.AnimatedVisibility
import androidx.compose.animation.expandVertically
import androidx.compose.animation.fadeIn
import androidx.compose.animation.fadeOut
import androidx.compose.animation.shrinkVertically
import androidx.compose.foundation.clickable
import androidx.compose.foundation.interaction.MutableInteractionSource
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
```

```
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Button
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.Icon
import androidx.compose.material.IconButton
import androidx.compose.material.RadioButton
import androidx.compose.material.RadioButtonDefaults
import androidx.compose.material.Scaffold
import androidx.compose.material.Text
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Delete
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavHostController


@Composable
fun AdressPickScreen(navController: NavHostController, adressViewModel:
AdressViewModel = viewModel()){
    val adresses = AdressViewModel.getAdresses()
    val selectedAdress = remember { mutableStateOf<String?>(null) }
    val buttonColor = colorResource(id =
R.color.lil_button_or_add_pay_address)

    Scaffold(
        topBar = {
            Text(
                modifier = Modifier.fillMaxWidth(),
                text = "Адрес доставки",
                style = TextStyle(fontSize = 20.sp, fontWeight =
FontWeight.Bold),
                textAlign = TextAlign.Center
            )
        },
        bottomBar = {
            Column(
                modifier = Modifier.fillMaxWidth().padding(10.dp)
            ) {
                Button(
                    onClick = { adressViewModel.addAdress("Адрес
${adresses.size + 1}") },
                    modifier = Modifier.fillMaxWidth(),
                    colors = ButtonDefaults.buttonColors(buttonColor)
                ) {
                    Text(text = "Добавить адрес", color = Color.Black)
                }

                Spacer(modifier = Modifier.height(10.dp))
```

```
                Button(
                    onClick = {

navController.navigate(NavigationItemsSec.Payment.route) {
                            launchSingleTop = true
                            restoreState = true
                        }
                    },
                    modifier = Modifier.fillMaxWidth(),
                    colors = ButtonDefaults.buttonColors(buttonColor)
                ) {
                    Text(text = "Перейти к оплате", color = Color.Black)
                }
            }
        }
    ){ innerValues ->
        if(adresses.isEmpty()){
            Box(modifier = Modifier.fillMaxSize(),
                contentAlignment = Alignment.Center ){
                Text(text = "Адреса доставки пока отсутвуют, добавьте новый
адрес, чтобы продолжить заказ",
                    style = TextStyle(fontSize = 20.sp, fontWeight =
FontWeight.Bold),
                    color = Color.Gray,
                    textAlign = TextAlign.Center)
            }
        }
        LazyColumn(
            modifier = Modifier
                .fillMaxSize()
                .padding(innerValues)
    ) {
            items(adresses, key = {it}) {adress ->
                AnimatedVisibility(
                    visible = true,
                    enter = fadeIn() + expandVertically(),
                    exit = fadeOut() + shrinkVertically()
                ) {
                AdressItem(
                    adress = adress,
                    isSelected = adress == selectedAdress.value,
                    onClick = {selectedAdress.value = adress},
                    onDelete = {adressViewModel.deleteAdress(adress)}
                )
            }
        }
        }
    }

}

@Composable
fun AdressItem(adress:String,
               isSelected: Boolean,
               onClick: ()-> Unit,
               onDelete:() -> Unit){
    val interactionSource = remember { MutableInteractionSource() }

    Row (
        modifier = Modifier
            .fillMaxWidth()
            .padding(8.dp)
            .clickable(onClick = onClick,
                interactionSource = interactionSource, indication = null),
```

```
        verticalAlignment = Alignment.CenterVertically
    ) { RadioButton(colors = RadioButtonDefaults.colors(
        selectedColor = colorResource(id = R.color.final_buttons)
    ),
        selected = isSelected,
        onClick = onClick
    )
        Column (
            modifier = Modifier.weight(1f)
                .padding(start = 6.dp)
        ){
            Text(text = adress,
                style = TextStyle(fontSize = 20.sp, fontWeight =
FontWeight.Bold) ,
                color = Color.Black)
            Text(text = "Дополнительная информация об адресе
${AdressViewModel.getAdresses().indexOf(adress) + 1}",
                style = TextStyle(fontSize = 14.sp, color = Color.Gray))
        }
        IconButton(onClick = onDelete) {
            Icon(imageVector = Icons.Default.Delete,
                contentDescription = "Удалить")
        }
    }
}
```

Файл AdressViewModel.kt

```
package com.example.a3kotlin

import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel

object AdressViewModel: ViewModel() {
    private var _adressList = mutableStateOf<List<String>>(listOf())

    fun addAdress(adress: String){
        _adressList.value += adress
    }

    fun deleteAdress(adress:String){
        _adressList.value = _adressList.value.filter { it != adress }
    }

    fun isAdressExist(adress: String): Boolean{

        return _adressList.value.contains(adress)
    }

    fun getAdresses(): List<String> = _adressList.value
}
```

Файл BottomNavigationBar.kt

```
package com.example.a3kotlin

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.widthIn
```

```
import androidx.compose.foundation.layout.wrapContentSize
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material.BottomNavigation
import androidx.compose.material.BottomNavigationItem
import androidx.compose.material3.BadgedBox
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import androidx.navigation.compose.currentBackStackEntryAsState


@Composable
fun BottomNavigationBar(navController: NavController) {
    val cartItemCount = CartViewModel.getTotalItems()
    val favItemCount = FavoritesViewModel.getTotalItems()

    val items = listOf(
        NavigationItems.Home,
        NavigationItems.Catalog,
        NavigationItems.ShoppingCard,
        NavigationItems.Favorites
    )

    val itemColor = colorResource(id = R.color.white)
    val backgroundColor = colorResource(id = R.color.top_down_color)
    val accentColor = colorResource(id =
R.color.lil_button_or_add_pay_address)

    BottomNavigation(
        backgroundColor = backgroundColor,
        contentColor = itemColor,
        modifier = Modifier
            .height(64.dp)
            .background(
                color = backgroundColor,
                shape = MaterialTheme.shapes.medium
            )
    ) {
        val navBackStackEntry by navController.currentBackStackEntryAsState()
        val currentRoute = navBackStackEntry?.destination?.route

        items.forEach { item ->
            BottomNavigationItem(
                    icon = {
                        when (item.route) {
                            NavigationItems.Favorites.route -> {
                                if (favItemCount != 0) {
                                    BadgedBox(
                                        modifier =
Modifier.wrapContentSize(),
                                        badge = {
                                            Surface(
                                                modifier = Modifier
                                                    .padding(top = 4.dp)
                                                    .widthIn(min = 16.dp),
```

```
                                                shape = CircleShape,
                                                color = accentColor,
                                                tonalElevation = 4.dp
                                            ) {
                                                Text(
                                                    text = if (favItemCount >
99) "99+" else favItemCount.toString(),

                                                    color = backgroundColor,
                                                    style =
MaterialTheme.typography.labelSmall,

                                                    modifier =
Modifier.padding(

                                                        horizontal = 4.dp,
                                                        vertical = 2.dp
                                                    ),
                                                    textAlign =
TextAlign.Center,

                                                    maxLines = 1
                                                )
                                            }
                                        }
                                    ) {
                                        Icon(
                                            painter = painterResource(id =
item.icon),

                                            contentDescription = item.title,
                                            modifier = Modifier.size(24.dp),
                                            tint = if (currentRoute ==
item.route) accentColor else itemColor
                                        )
                                    }
                                } else {
                                    Icon(
                                        painter = painterResource(id =
item.icon),

                                        contentDescription = item.title,
                                        modifier = Modifier.size(24.dp),
                                        tint = if (currentRoute ==
item.route) accentColor else itemColor
                                    )
                                }
                            }

                            NavigationItems.ShoppingCard.route -> {
                                if (cartItemCount != 0) {
                                    BadgedBox(
                                        modifier =
Modifier.wrapContentSize(),

                                        badge = {
                                            Surface(
                                                modifier = Modifier
                                                    .padding(top = 4.dp)
                                                    .widthIn(min = 16.dp),
                                                shape = CircleShape,
                                                color = accentColor,
                                                tonalElevation = 4.dp
                                            ) {
                                                Text(
                                                    text = if (cartItemCount
> 99) "99+" else cartItemCount.toString(),

                                                    color = backgroundColor,
                                                    style =
MaterialTheme.typography.labelSmall,

                                                    modifier =
```

```
Modifier.padding(
                                                        horizontal = 4.dp,
                                                        vertical = 2.dp
                                                    ),
                                                    textAlign =
TextAlign.Center,
                                                    maxLines = 1
                                                )
                                            }
                                        }
                                    ) {
                                        Icon(
                                            painter = painterResource(id =
item.icon),
                                            contentDescription = item.title,
                                            modifier = Modifier.size(24.dp),
                                            tint = if (currentRoute ==
item.route) accentColor else itemColor
                                        )
                                    }
                                } else {
                                    Icon(
                                        painter = painterResource(id =
item.icon),
                                        contentDescription = item.title,
                                        modifier = Modifier.size(24.dp),
                                        tint = if (currentRoute ==
item.route) accentColor else itemColor
                                    )
                                }
                            }

                            else -> {
                                Icon(
                                    painter = painterResource(id =
item.icon),
                                    contentDescription = item.title,
                                    modifier = Modifier.size(24.dp),
                                    tint = if (currentRoute == item.route)
accentColor else itemColor
                                )
                            }
                        }
                    },
                    label = {
                        Text(
                            text = item.title,
                            style = MaterialTheme.typography.labelSmall,
                            maxLines = 1,
                            color = itemColor
                        )
                    },
                    selectedContentColor = accentColor,
                    unselectedContentColor = itemColor,
                    alwaysShowLabel = true,
                    selected = currentRoute == item.route,
                    onClick = {
                        navController.navigate(item.route) {
                            launchSingleTop = true
                            restoreState = true
                        }
                    }
                )
            }
```

```
    }
}
```

Файл FavoritesScreen.kt

```
package com.example.a3kotlin

import android.content.res.Configuration
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.wrapContentSize
import androidx.compose.foundation.lazy.grid.GridCells
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
import androidx.compose.foundation.lazy.grid.items
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalConfiguration
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavHostController

@Composable
fun FavoritesScreen(navController: NavHostController) {

    val products = FavoritesViewModel.getProducts()
    val configuration = LocalConfiguration.current
    val columns = if (configuration.orientation ==
Configuration.ORIENTATION_PORTRAIT) 2 else 3
    val buttonColor = colorResource(id =
R.color.lil_button_or_add_pay_address)


    Column(modifier = Modifier.fillMaxSize()) {
        Box(
            modifier = Modifier.fillMaxWidth()
                .height(30.dp),
            contentAlignment = Alignment.Center
        ) {
            Text(
                text = "Избранное",
                style = TextStyle(fontSize = 20.sp,fontWeight =
FontWeight.Bold)
            )
        }
        if (products.isEmpty()) {
```

```
                Box(
                    modifier = Modifier.fillMaxSize(),
                    contentAlignment = Alignment.Center
                ) {
                    Column (verticalArrangement = Arrangement.Center,
                        horizontalAlignment = Alignment.CenterHorizontally){
                        Text(
                            text = "У вас пока нет избранных товаров",
                            style = TextStyle(fontSize = 25.sp, color =
Color.Gray),
                            textAlign = TextAlign.Center
                        )
                        Spacer(Modifier.height(40.dp))
                        Button(
                            onClick = {
navController.navigate(NavigationItems.Home.route)},
                            colors = ButtonDefaults.buttonColors(
                                buttonColor
                            )
                            )


                             {
                            Text(
                                text = "Вернуться на главную",
                                color = Color.Black
                            )
                        }
                    }
                }
            }


    else {



            Box(
                modifier = Modifier.fillMaxWidth()
                  ,
                contentAlignment = Alignment.CenterEnd
            )
            {
                Button(
                    onClick = {FavoritesViewModel.clear()},
                    modifier = Modifier.padding(10.dp)
                        .wrapContentSize(),
                    colors = ButtonDefaults.buttonColors(
                        buttonColor
                    )
                    ) {
                    Text(
                        text = "Очистить избранное",
                        style = TextStyle(fontSize = 15.sp),
                        color = colorResource(id = R.color.black)
                    )
                }
            }
            LazyVerticalGrid(
                columns = GridCells.Fixed(columns), // Устанавливаем
количество столбцов в зависимости от ориентации
                contentPadding = PaddingValues(16.dp),
                modifier = Modifier.fillMaxSize()
```

```
            ) {
                items(products) { product ->
                    ProductCard(product = product, navController =
navController)
                }
            }
        }
    }
}
```

Файл FavoritesViewModel.kt

```
package com.example.a3kotlin

import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel

object FavoritesViewModel: ViewModel() {
    private var _favItems = mutableStateOf<List<Product>>(listOf())

    fun addProduct(product: Product) {
        _favItems.value = _favItems.value + product
    }

    fun removeProduct(product: Product) {
        _favItems.value = _favItems.value.filter { it.id != product.id }
    }

    fun getProducts(): List<Product> = _favItems.value

    fun getTotalItems(): Int = _favItems.value.size

    fun clear() {
        _favItems.value = listOf()
    }
}
```

Файл HomeScreen.kt

```
package com.example.a3kotlin

import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel

object FavoritesViewModel: ViewModel() {
    private var _favItems = mutableStateOf<List<Product>>(listOf())

    fun addProduct(product: Product) {
        _favItems.value = _favItems.value + product
    }

    fun removeProduct(product: Product) {
        _favItems.value = _favItems.value.filter { it.id != product.id }
    }

    fun getProducts(): List<Product> = _favItems.value

    fun getTotalItems(): Int = _favItems.value.size

    fun clear() {
        _favItems.value = listOf()
    }
}
```

Файл HomeViewModel.kt

```
package com.example.a3kotlin

import androidx.lifecycle.SavedStateHandle
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch

class HomeViewModel(private val savedStateHandle: SavedStateHandle) :
ViewModel() {

    private val _products = MutableStateFlow<List<Product>>(emptyList())
    val products: StateFlow<List<Product>> = _products

    private val _isLoading = MutableStateFlow(false)
    val isLoading: StateFlow<Boolean> = _isLoading

    private val isLoadedKey = "is_loaded"
    private var isLoaded = savedStateHandle.get<Boolean>(isLoadedKey) ?:
false

    private val mockProducts = MockData.getMockedProducts()

    fun loadProducts() {
        // Проверяем, нужно ли загружать данные
        if (!isLoaded && !_isLoading.value) {
            _isLoading.value = true
            viewModelScope.launch {
                delay(2000) // Симулируем задержку
                _products.value = mockProducts
                isLoaded = true
                savedStateHandle[isLoadedKey] = isLoaded
                _isLoading.value = false
            }
        }
    }
}
```

Файл MockData.kt

```
package com.example.a3kotlin

object MockData {
    val productList = listOf(
        Product(
        id = "1",
        name = "Apple Iphone 15",
        price = 279900,
        description = "Пока нет описания",
        imageRes = R.drawable.iphone_15,
        features = listOf("Чёрный", "512ГБ", "eSim")
    ),
        Product(
            id = "2",
            name = "Apple Watch 15",
            price = 279900,
            description = "Пока нет описания",
            imageRes = R.drawable.apple_watch,
```

```
            features = listOf("Чёрный", "12", "Есть")
        ),Product(
            id = "3",
            name = "MacBook Pro",
            price = 1279900,
            description = "Пока нет описания",
            imageRes = R.drawable.macbook_pro,
            features = listOf("Чёрный", "1024ГБ")
        ),
        Product(
            id = "4",
            name = "Apple Iphone 15",
            price = 279900,
            description = "Пока нет описания",
            imageRes = R.drawable.iphone_15,
            features = listOf("Чёрный", "512ГБ", "eSim")
        ),

    )
    fun getMockedProducts(): List<Product> {
        return productList
    }
}
```

Файл Navigation.kt

```
package com.example.a3kotlin

import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.navArgument

@Composable
fun Navigation(navController: NavHostController) {

    NavHost(navController, startDestination = NavigationItems.Home.route) {

        composable(NavigationItems.Home.route) {
            HomeScreen(navController)
        }

        composable(NavigationItems.Catalog.route) {
            CatalogScreen(navController)
        }

        composable(NavigationItems.ShoppingCard.route) {
            ShoppingCardScreen(navController)
        }
        composable(NavigationItems.Favorites.route) {
            FavoritesScreen(navController)
        }
        composable(
            "productDetail/{productId}",
            arguments = listOf(navArgument("productId") { type =
NavType.StringType })
        ) { backStackEntry ->
            val productId = backStackEntry.arguments?.getString("productId")
```

```
?: ""
            val product = MockData.productList.firstOrNull { it.id ==
productId }

            if (product != null) {
                ProductDetailScreen(
                    product = product,
                    navController = navController
                )
            } else {
                Text("Product not found")
            }
        }
        composable(NavigationItemsSec.Address.route){
            AdressPickScreen(navController)
        }
        composable(NavigationItemsSec.Payment.route){
            PaymentScreen(navController)
        }
        composable(NavigationItemsSec.Success.route){
            Success(navController)
        }
    }
}
```

Файл NavigationItems.kt

```
package com.example.a3kotlin

sealed class NavigationItems(var route: String, var icon: Int, var title:
String)
{
    data object Home : NavigationItems("home", R.drawable.home, "Главная")
    data object Catalog : NavigationItems("catalog", R.drawable.menu,
"Каталог")
    data object ShoppingCard : NavigationItems("shopping_cart",
R.drawable.shopping_cart_outlined, "Корзина")
    data object Favorites : NavigationItems("favorites",
R.drawable.favorite_outlined, "Избранное")
}
```

Файл NavigationItemsSec.kt

```
package com.example.a3kotlin

sealed class NavigationItemsSec(var route: String) {
    data object Address : NavigationItemsSec("address")
    data object Payment : NavigationItemsSec("payment")
    data object Success : NavigationItemsSec("success")
}
```

Файл PaymentScreen.kt

```
package com.example.a3kotlin
```

```
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.itemsIndexed
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Delete
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.DropdownMenu
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.RadioButton
import androidx.compose.material3.RadioButtonDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavHostController
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.style.TextAlign
import androidx.navigation.NavController
import androidx.compose.ui.graphics.Color
import androidx.lifecycle.viewmodel.compose.viewModel


@Composable
fun PaymentScreen(navController: NavHostController, favoritesViewModel:
FavoritesViewModel = viewModel()) {

    val paymentMethods = PaymentViewModel.getMethods()
    val selectedOption = remember { mutableStateOf<String?>(null) }
    val isDropdownVisible = remember { mutableStateOf(false) }

    val isAddingCard = remember { mutableStateOf(false) }
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Text(
            text = "Выберите способ оплаты",
            style = TextStyle(fontSize = 20.sp, fontWeight =
FontWeight.Bold),
            modifier = Modifier.padding(bottom = 16.dp)
        )


        PaymentOption(
```

```
            label = "Оплатить картой при получении",
            isSelected = selectedOption.value == "card_on_delivery",
            onSelect = { selectedOption.value = "card_on_delivery" }
        )
        PaymentOption(
            label = "Оплатить наличными курьеру",
            isSelected = selectedOption.value == "cash",
            onSelect = { selectedOption.value = "cash" }
        )
        PaymentOption(
            label = "Оплатить картой онлайн",
            isSelected = selectedOption.value == "online",
            onSelect = {
                selectedOption.value = "online"
                isDropdownVisible.value = true
            }
        )

        if (selectedOption.value == "online" && isDropdownVisible.value) {
            Spacer(modifier = Modifier.height(16.dp))

                Column (
                    modifier = Modifier
                        .fillMaxWidth()
                        .padding(top = 8.dp)
                ) {
                    Text(
                        "Сохраненные карты:",
                        style = TextStyle(fontSize = 20.sp, fontWeight =
FontWeight.Bold)
                    )
                    Spacer(Modifier.height(10.dp))

                        LazyColumn(
                            modifier = Modifier.fillMaxWidth()
                        ) {
                            if (paymentMethods.isNotEmpty()) {
                                itemsIndexed(paymentMethods) { index, card ->
                                    Row(
                                        modifier =
Modifier.fillMaxWidth().padding(8.dp),
                                        verticalAlignment =
Alignment.CenterVertically
                                    ) {
                                        Text(text = card, modifier =
Modifier.weight(1f))

                                        IconButton(
                                            onClick = {

PaymentViewModel.removeMethod(card)
                                            }
                                        ) {
                                            Icon(
                                                imageVector =
Icons.Default.Delete,
                                                contentDescription = "Удалить
карту"
                                            )
                                        }
                                    }
                                }
                            }
                            else {
```

```
                                item {
                                    Box (contentAlignment =
Alignment.Center){
                                        Text(text = "Сохранненые карты
отсутствуют, добавьте новый способ оплаты",
                                            textAlign = TextAlign.Center,
                                            style = TextStyle(fontSize =
15.sp, fontWeight = FontWeight.Bold),
                                            color = Color.Gray)
                                    }
                                    Spacer(modifier = Modifier.height(16.dp))
                                }
                            }
                                item {
                                Button(
                                    onClick = {
                                        isAddingCard.value = true
                                        PaymentViewModel.addMethod(

"***${kotlin.random.Random.nextInt(1000, 10000)}"
                                        )
                                    },
                                    modifier = Modifier.fillMaxWidth(),
                                    colors =
ButtonDefaults.buttonColors(colorResource(id = R.color.add_button))
                                ) {
                                    Text(text = "Добавить новую карту")
                                }
                            }
                            item {
                                Button(
                                    onClick = {
                                        CartViewModel.clear()

navController.navigate(NavigationItemsSec.Success.route) {
                                            launchSingleTop = true
                                            restoreState = true
                                        }
                                    },
                                    modifier = Modifier.fillMaxWidth(),
                                    colors =
ButtonDefaults.buttonColors(colorResource(id = R.color.add_button))
                                ) {
                                    Text(text = "Оплатить!")
                                }
                            }
                        }
                    }

                    }


                }
        }


@Composable
fun PaymentOption(label: String, isSelected: Boolean, onSelect: () -> Unit) {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .clickable { onSelect() }
            .padding(vertical = 8.dp),
```

```
        verticalAlignment = Alignment.CenterVertically
    ) {
        RadioButton(
            selected = isSelected,
            onClick = { onSelect() },
            colors = RadioButtonDefaults.colors(
                selectedColor = colorResource(id = R.color.final_buttons)
            )
        )
        Spacer(modifier = Modifier.width(8.dp))
        Text(text = label, style = TextStyle(fontSize = 16.sp))
    }
}
```

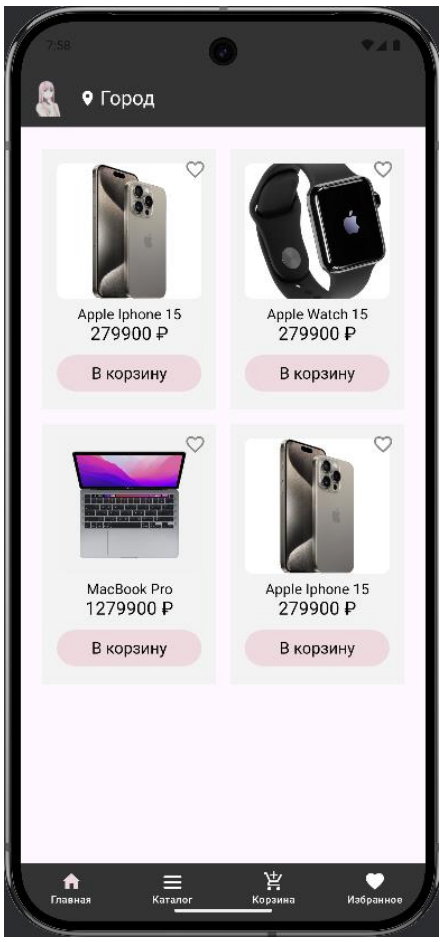
Файл PaymentViewModel.kt

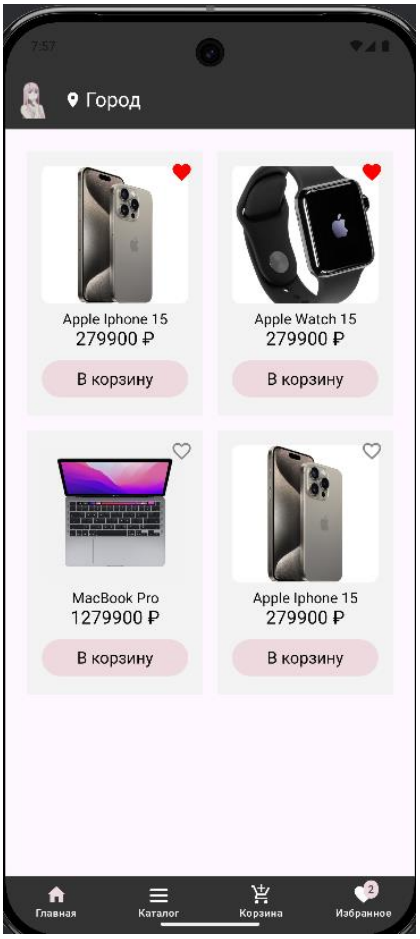
```
package com.example.a3kotlin

import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel

object PaymentViewModel: ViewModel() {
    private var _paymentMethods = mutableStateOf<List<String>>(listOf())

    fun addMethod (name: String){
        _paymentMethods.value = _paymentMethods.value + name
    }

    fun removeMethod (name: String){
        _paymentMethods.value = _paymentMethods.value.filter {it != name}
    }

    fun getMethods() : List<String> = _paymentMethods.value

}
```

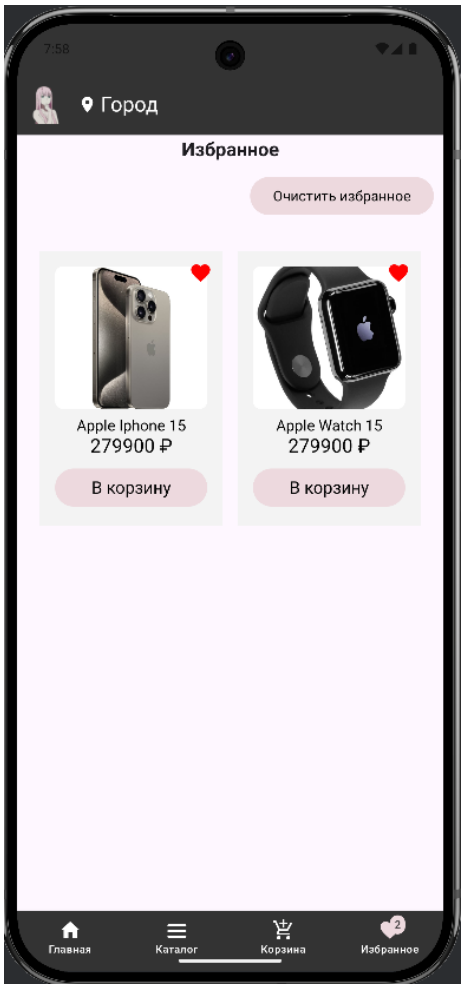
Файл Product.kt

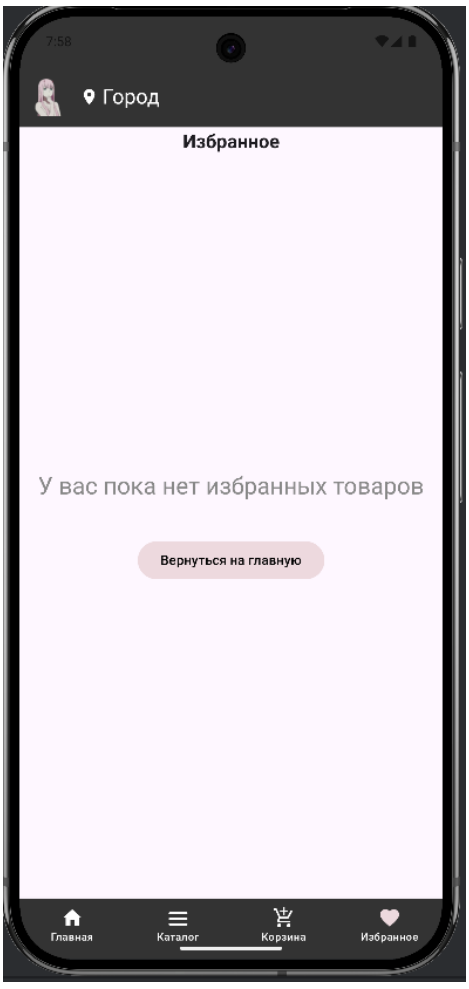
```
package com.example.a3kotlin

data class Product(
    val id: String,
    val name: String,
    val price: Int,
    val description: String,
    val imageRes: Int,
    val features: List<String>) {
}
```

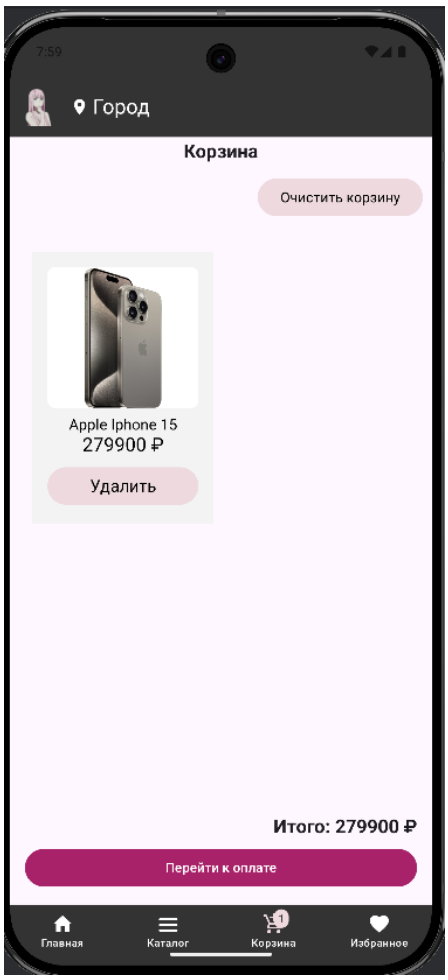
Результаты работы программы

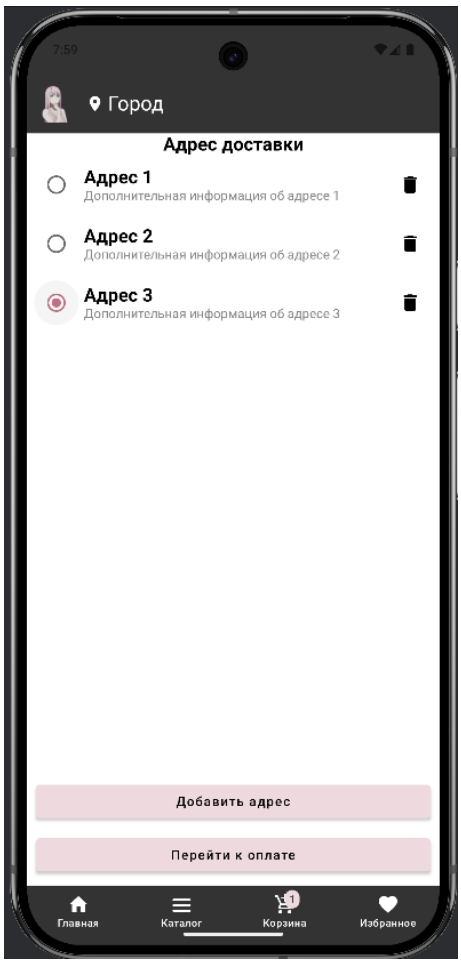

Город
Apple Iphone 15
279900 ₽
В корзину
Apple Watch 15
279900 ₽
В корзину
MacBook Pro
1279900 ₽
В корзину
Apple Iphone 15
279900 ₽
В корзину
Главная
Каталог
Корзина
Избранное

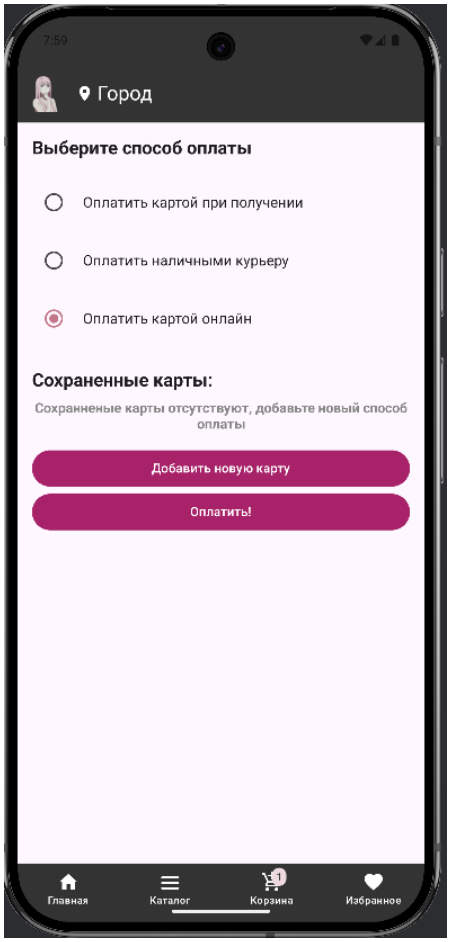

Город
Apple Iphone 15
279900 ₽
В корзину
Apple Watch 15
279900 ₽
В корзину
MacBook Pro
1279900 ₽
В корзину
Apple Iphone 15
279900 ₽
В корзину
Главная
Каталог
Корзина
Избранное
2

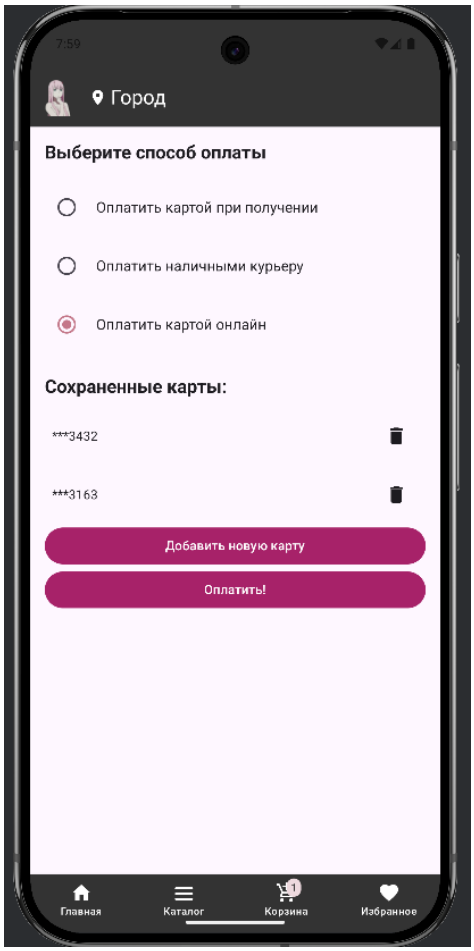

Город
Избранное
Очистить избранное
Apple Iphone 15
279900 ₽
В корзину
Apple Watch 15
279900 ₽
В корзину
Главная
Каталог
Корзина
Избранное
2

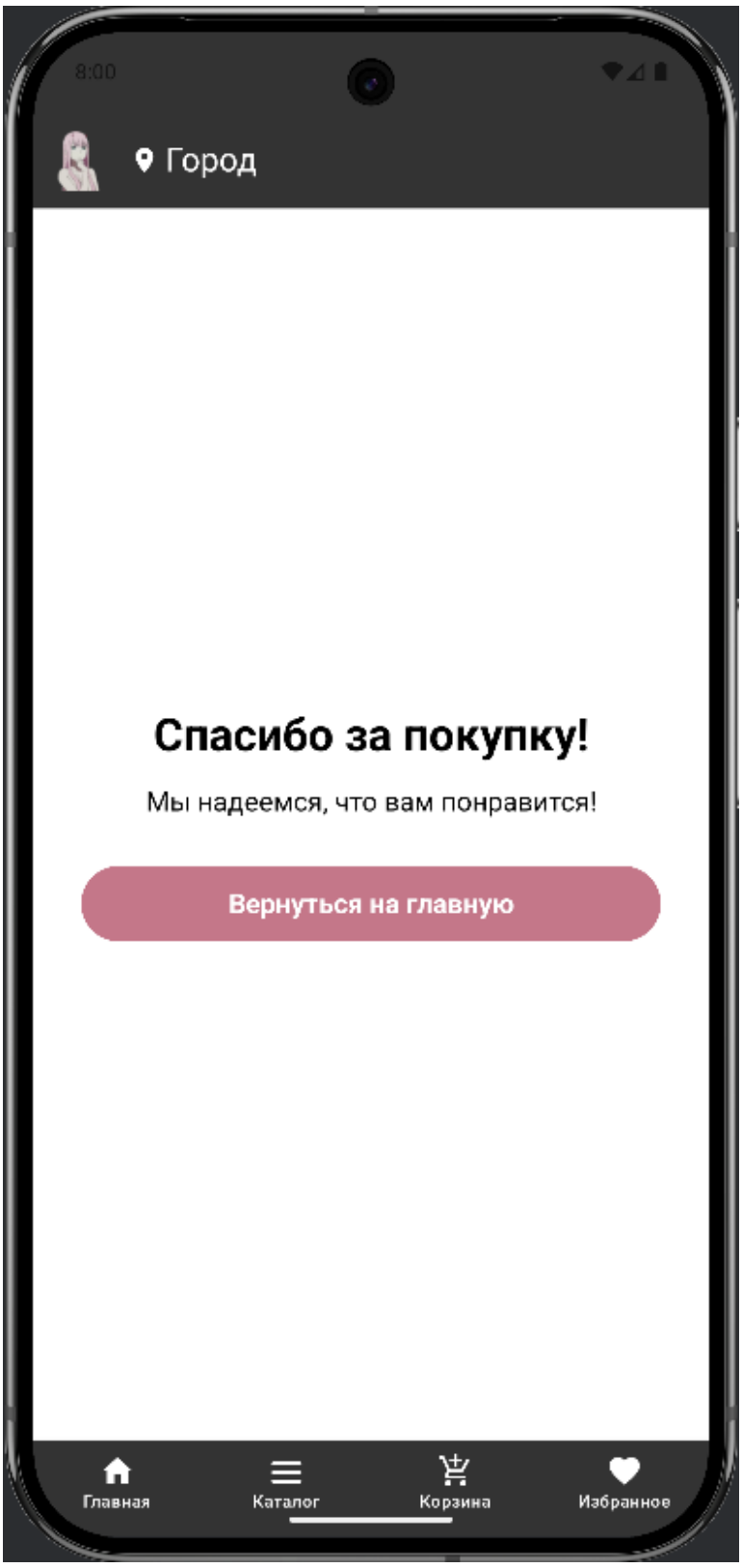

Город
Избранное
У вас пока нет избранных товаров
Вернуться на главную
Главная
Каталог
Корзина
Избранное


Город
Корзина
Очистить корзину
Apple Iphone 15
279900 ₽
Удалить
Итого: 279900 ₽
Перейти к оплате
Главная
Каталог
Корзина
Избранное


Город
Адрес доставки
Адрес 1
Дополнительная информация об адресе 1
Адрес 2
Дополнительная информация об адресе 2
Адрес 3
Дополнительная информация об адресе 3
Добавить адрес
Перейти к оплате
Главная
Каталог
Корзина
Избранное


Город
Выберите способ оплаты
Оплатить картой при получении
Оплатить наличными курьеру
Оплатить картой онлайн
Сохраненные карты:
Сохранненые карты отсутствуют, добавьте новый способ оплаты
Добавить новую карту
Оплатить!
Главная
Каталог
Корзина
Избранное


Город
Выберите способ оплаты
Оплатить картой при получении
Оплатить наличными курьеру
Оплатить картой онлайн
Сохраненные карты:
***3432
***3163
Добавить новую карту
Оплатить!
Главная
Каталог
Корзина
Избранное


8:00
Город
Спасибо за покупку!
Мы надеемся, что вам понравится!
Вернуться на главную
Главная
Каталог
Корзина
Избранное