КАФЕДРА

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчет о лабораторной работе №5

Организация навигации в многоэкранном приложении

По дисциплине: Программирование мобильных устройств

РАБОТУ ВЫПОЛНИЛИ

СТУДЕНТЫ ГР. № _____ _____ _____

подпись, дата инициалы, фамилия

Санкт-Петербург 2024

Задание:

Организовать навигацию между несколькими Activity (не менее 3-х) при помощи обработки нажатий на интерфейсные элементы, выполнить сборку, отладку и запуск приложения. Навигация должна позволять возможность перехода из любой экранной формы (Activity) в любую другую, возможны различные способы организации (переход из основной и обратно, прямой переход из каждой в каждую, циклический переход по цепочке) Описать результаты в отчете (в т.ч. исх. тексты java и xml для activity).

Цель работы:

Получение навыков организации навигации в многоэкранных мобильных приложениях

Выполнение задания:

1. **Добавление нового экрана "Лог операций"**:
    - Был добавлен новый экран под названием "Лог операций", расширяющий функционал приложения.
    - Этот экран предназначен для отображения всех событий, связанных с балансом пользователя, включая выигрыши, проигрыши и покупки.
    - На экране реализовано отображение списка событий. Каждая запись в списке содержит информацию о характере операции (выигрыш, проигрыш, покупка), сумме операции и других деталях.
    - В случае отсутствия событий отображается сообщение "Записей в логе нет".
    - Записи в логе отображаются в обратном порядке, так что последние события показываются в начале списка.
2. **Добавление нижнего меню для навигации между экранами**:
    - Было реализовано нижнее меню навигации, что позволяет пользователям легко переключаться между различными экранами приложения.
    - Нижнее меню включает вкладки для перехода на основной игровой экран, экран магазина и экран лога операций.
    - Использование **TabView** обеспечивает удобство навигации и улучшает пользовательский интерфейс.
    - Каждая вкладка меню имеет уникальный идентификатор и соответствующий значок, что облегчает ориентацию в приложении.
3. **Нововведения в логике приложения**:
    - Внесены изменения в логику игрового процесса и процесса покупки для интеграции системы логирования.

- При выполнении операций (выигрыш, проигрыш, покупка) соответствующие события записываются в лог. Это позволяет пользователям отслеживать и анализировать свои действия в приложении.
- Внедрено логирование выигрышей и проигрышей с указанием суммы и диапазона чисел в игре.
- Реализована функция логирования покупок в магазине, включая название и стоимость приобретаемого товара.
- Все записи в логе снабжены эмодзи монеты для единообразия отображения валютных операций.

Каждое из этих нововведений направлено на улучшение пользовательского опыта и расширение функциональности приложения, делая его более интерактивным и информативным для пользователя.

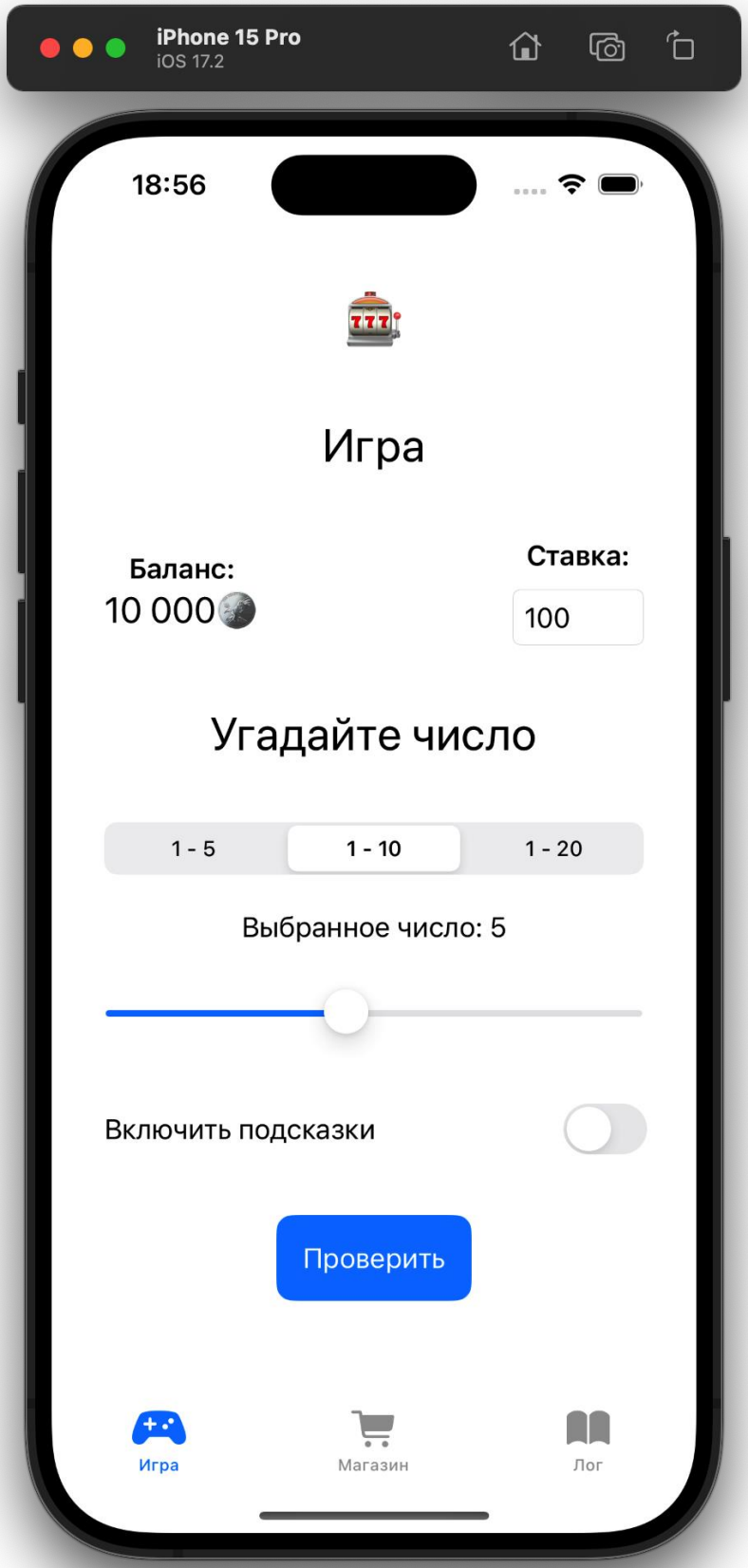

Рисунок 1 – вид приложения при запуске

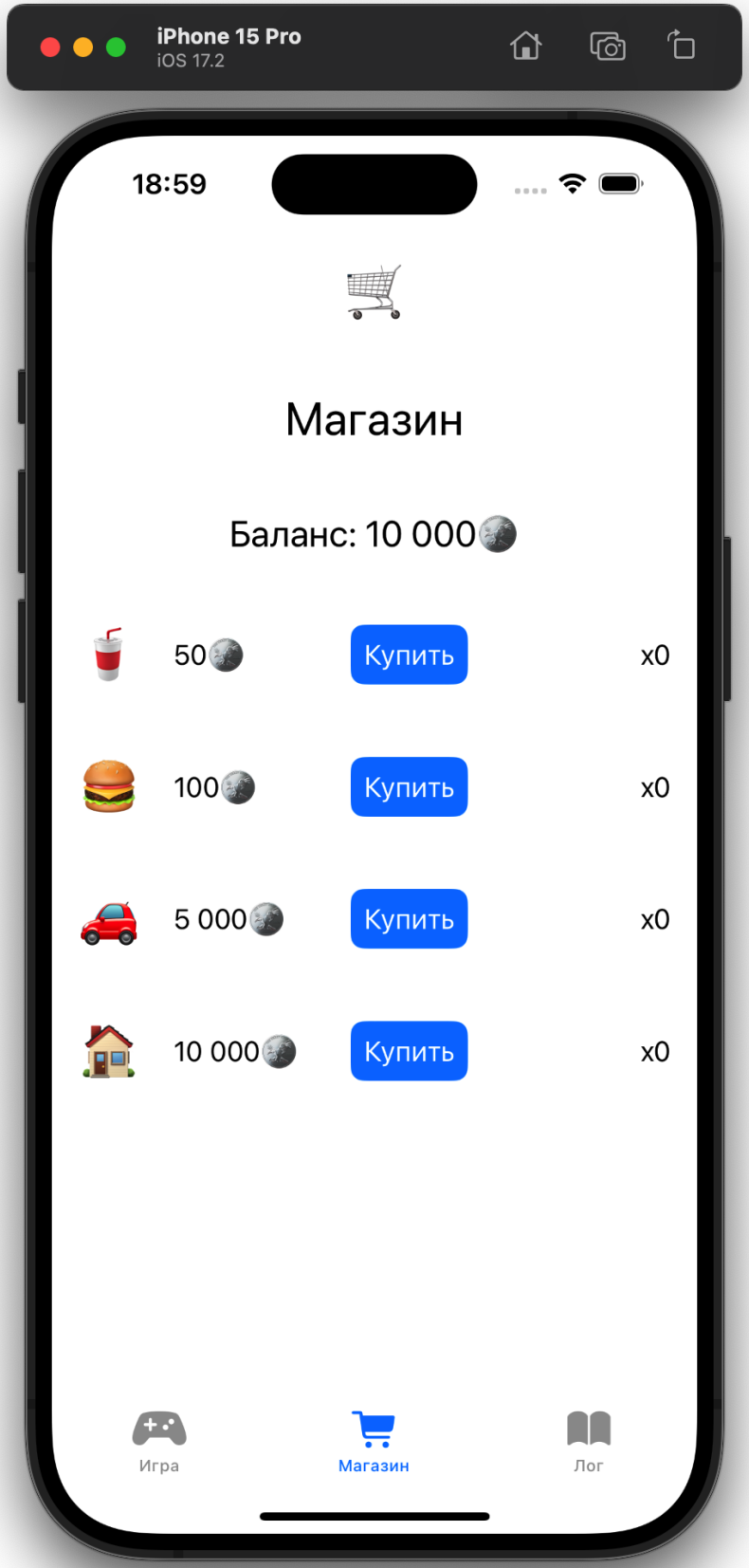

Рисунок 2 – переход на экран Магазин, используя меню

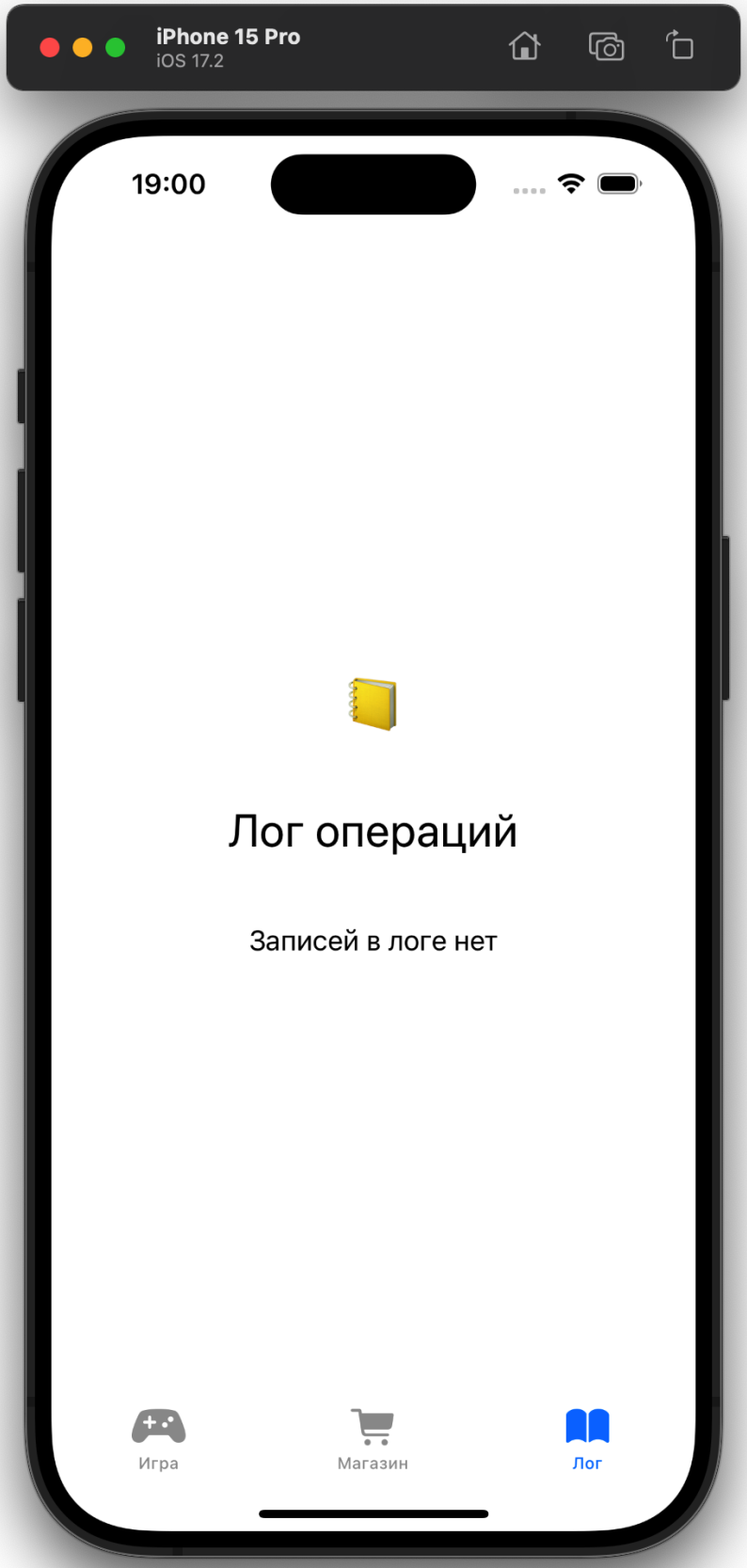

Рисунок 3 – переход на экран Лог, используя меню

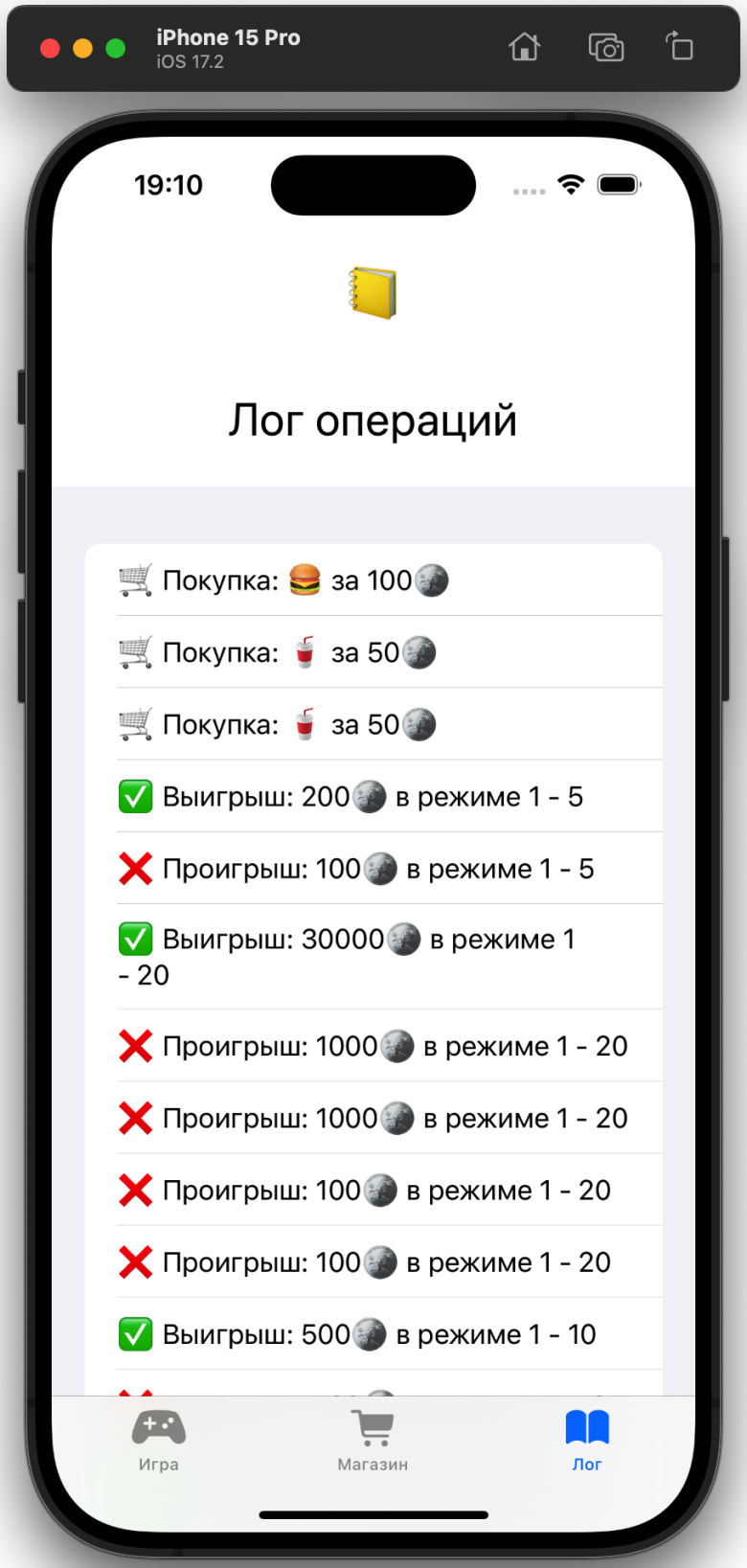

Рисунок 4 – лог операций после использования приложения

Листинг:

LabaApp.swift

```
//
//  LabaApp.swift
//  Laba
//
//  Created by Андрей Захаров on 28.02.2024.
//

import SwiftUI

@main
struct LabaApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

ContentView.swift

```
//
//  ContentView.swift
//  Laba
//
//  Created by Андрей Захаров on 28.02.2024.
//

import SwiftUI

struct ContentView: View {
    @State private var userGuess: Double = 5
    @State private var randomNumber = Int.random(in: 1...10)
    @State private var showAlert = false
    @State private var alertTitle = ""
    @State private var isHintEnabled: Bool = false
    @State private var numberRange = "1 - 10"
    let numberRanges = ["1 - 5", "1 - 10", "1 - 20"]
    @State private var balance: Int = 10000
    @State private var betAmount: Int = 100
    @State private var logRecords: [String] = []

    var body: some View {
        TabView {
            // Вкладка игрового экрана
```

```
        gameView()
            .tabItem {
                Label("Игра", systemImage: "gamecontroller")
            }
            .tag(1)

        // Вкладка магазина
        ShopView(balance: $balance, logRecords: $logRecords)
            .tabItem {
                Label("Магазин", systemImage: "cart")
            }
            .tag(2)

        // Вкладка лога операций
        LogView(logs: logRecords)
            .tabItem {
                Label("Лог", systemImage: "book")
            }
            .tag(3)
    }
}

func gameView() -> some View {
    NavigationView {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Игра")
                .font(.title)
                .padding()

            HStack {
                VStack {
                    Text("Баланс:")
                        .font(.headline)
                    Text("\(balance)□")
                        .font(.title2)
                }
                Spacer()
                VStack {
                    Text("Ставка:")
                        .font(.headline)
                    TextField("100", value: $betAmount, format: .number)
                        .textFieldStyle(RoundedBorderTextFieldStyle())
                        .keyboardType(.numberPad)
                        .frame(width: 80)
                }
            }
            .padding()
            Spacer(minLength: 10)
```

```
            Text("Угадайте число")
                .font(.title)
            Spacer(minLength: 20)

            Picker("Выберите диапазон чисел", selection: $numberRange) {
                ForEach(numberRanges, id: \.self) {
                    Text($0)
                }
            }
            .pickerStyle(SegmentedPickerStyle())
            .padding()
            .onChange(of: numberRange) { _ in
                updateRandomNumber()
            }
            Text("Выбранное число: \(Int(userGuess))")

            Slider(value: $userGuess, in: 1...CGFloat(getUpperRangeLimit()), step: 1)
                .padding()

            Toggle("Включить подсказки", isOn: $isHintEnabled)
                .padding()
            Spacer(minLength: 20)

            Button("Проверить") {
                checkGuess()
            }
            .padding()
            .background(Color.blue)
            .foregroundColor(.white)
            .cornerRadius(10)
            Spacer(minLength: 20)

                .alert(isPresented: $showAlert) {
                    Alert(title: Text(alertTitle), dismissButton: .default(Text("OK")))
                }
            Spacer(minLength: 20)
        }
        .padding()
    }
}

func updateRandomNumber() {
    let limit = getUpperRangeLimit()
    randomNumber = Int.random(in: 1...limit)
    userGuess = min(userGuess, Double(limit))
}

func getUpperRangeLimit() -> Int {
    switch numberRange {
    case "1 - 20":
```

```
        return 20
    case "1 - 5":
        return 5
    default: // "1 - 10"
        return 10
    }
}

func checkGuess() {
    if betAmount > balance {
        alertTitle = "Ваша ставка превышает баланс!"
        showAlert = true
        return
    }

    balance -= betAmount
    let guess = Int(userGuess)

    if guess == randomNumber {
        var winMultiplier = 1.0
        switch numberRange {
        case "1 - 5":
            winMultiplier = 2.0
        case "1 - 10":
            winMultiplier = 5.0
        case "1 - 20":
            winMultiplier = 10.0
        default:
            break
        }

        let winAmount = Int(Double(betAmount) * winMultiplier)
        balance += winAmount
        alertTitle = "Правильно! Вы угадали число!"
        logRecords.append("✅Выигрыш: \(winAmount)□ в режиме \(numberRange)")
        updateRandomNumber()
    } else {
        if isHintEnabled {
            if guess < randomNumber {
                alertTitle = "Слишком мало! Попробуйте число побольше."
            } else {
                alertTitle = "Слишком много! Попробуйте число поменьше."
            }
        } else {
            alertTitle = "Не угадали! Попробуйте еще раз."
        }
        logRecords.append("❌Проигрыш: \(betAmount)□ в режиме \(numberRange)")
    }

    showAlert = true
}
```

```
}

// Preview section
struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

ShopView.swift

```
//
//  ShopView.swift
//  Laba
//
//  Created by Андрей Захаров on 07.03.2024.
//

import SwiftUI

struct ShopView: View {
    @Binding var balance: Int
    @Binding var logRecords: [String]
    @State private var showAlert = false
    @State private var alertTitle = ""

    struct Product {
        var emoji: String
        var price: Int
        var count: Int
    }

    @State private var products: [Product] = [
        Product(emoji: "□", price: 50, count: 0),
        Product(emoji: "□", price: 100, count: 0),
        Product(emoji: "□", price: 5000, count: 0),
        Product(emoji: "□", price: 10000, count: 0)
    ]

    var body: some View {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Магазин")
                .font(.title)
                .padding()
            Text("Баланс: \(balance)□")
                .font(.title2)
                .padding()
```

```
            // Список товаров
            ForEach($products.indices, id: \.self) { index in
                HStack {
                    Text(products[index].emoji)
                        .font(.largeTitle)
                        .frame(width: 50, alignment: .leading)
                    Text("\(products[index].price)□")
                        .frame(width: 100, alignment: .leading)
                    Button("Купить") {
                        buyProduct(index: index)
                    }
                    .padding(8)
                    .background(Color.blue)
                    .foregroundColor(.white)
                    .cornerRadius(8)
                    Spacer()
                    Text("x\(products[index].count)")
                }
                .padding()
            }

            Spacer()
        }
        .alert(isPresented: $showAlert) {
            Alert(title: Text(alertTitle), dismissButton: .default(Text("OK")))
        }
    }

    func buyProduct(index: Int) {
        if products[index].price <= balance {
            balance -= products[index].price
            products[index].count += 1
            logRecords.append("□ Покупка: \(products[index].emoji) за
\(products[index].price)□")
        } else {
            alertTitle = "Недостаточно средств для покупки!"
            showAlert = true
        }
    }
}

struct ShopView_Previews: PreviewProvider {
    @State static var tempBalance = 10000
    @State static var tempLogRecords: [String] = []

    static var previews: some View {
        ShopView(balance: $tempBalance, logRecords: $tempLogRecords)
    }
}
```

LogView.swift

```
//
//  LogView.swift
//  Laba
//
//  Created by Андрей Захаров on 07.03.2024.
//

import SwiftUI

struct LogView: View {
    var logs: [String] // Массив строк с записями лога

    var body: some View {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Лог операций")
                .font(.title)
                .padding()

            // Проверяем, есть ли записи в логе
            if logs.isEmpty {
                Text("Записей в логе нет")
                    .padding()
            } else {
                List(logs.reversed(), id: \.self) { log in
                    Text(log)
                }
            }
        }
    }
}

// Предпросмотр
struct LogView_Previews: PreviewProvider {
    static var previews: some View {
        // Создание примера данных для предпросмотра
        let sampleLogs = ["Покупка: □ за 50□", "Выигрыш: 500 в режиме 1 - 10"]

        // Возвращаем LogView с примерными данными
        LogView(logs: sampleLogs)
    }
}
```