Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе «Приложение на Swift»

Выполнил:
студент группы ИУ5-33Б
Горенков А.А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.
Подпись и дата:

Постановка задачи

Постановка задачи для приложения "Менеджер плейлистов":

Функциональные требования:

1. **Управление плейлистами:**
   - Добавление нового плейлиста с возможностью задания названия.
   - Удаление существующего плейлиста.
   - Возможность изменения названия плейлиста.
   - Отображение списка плейлистов с функцией поиска по названию.
2. **Управление треками в плейлисте:**
   - Добавление треков в выбранный плейлист через файловую систему устройства.
   - Отображение списка треков в каждом плейлисте с возможностью удаления конкретного трека.
3. **Онлайн радио:**
   - Возможность включения/выключения онлайн радио.
   - Отображение статуса воспроизведения радио и соответствующей кнопки управления.
4. **Интерфейс:**
   - Интуитивно понятный и удобный интерфейс.
   - Предоставление контекстного меню для операций с плейлистами (например, удаление).

Шаги для реализации основных функций:

1. **Управление плейлистами:**
   - **Добавление нового плейлиста:**
     - `addPlaylist()`: Функция для добавления нового плейлиста в массив `playlists`.
   - **Удаление плейлиста:**
     - `deletePlaylist(at index: Int)`: Метод для удаления плейлиста из массива `playlists`.
   - **Изменение названия плейлиста:**
     - `changePlaylistName(newName: String)`: Функция для изменения названия плейлиста.
2. **Онлайн радио:**
   - **Включение/выключение радио:**
     - `playRadio()`: Метод для запуска воспроизведения онлайн радио.
     - `stopRadio()`: Метод для остановки воспроизведения онлайн радио.
     - `toggleRadio()`: Функция для переключения статуса радио (включение/выключение).
3. **Интерфейс и пользовательский опыт:**
   - **Обновление интерфейса:**
     - `updateInterface()`: Функция для обновления интерфейса после внесения изменений.
   - **Разделение логики и представления:**

- `PlaylistView()`: Представление для управления плейлистами и их отображением.
- `TrackView()`: Представление для управления треками и их отображением.

Текст программы:

```
import SwiftUI
import AVKit
import UniformTypeIdentifiers

struct Song: Identifiable {
    var id = UUID()
    var title: String
}

struct Playlist: Identifiable {
    var id = UUID()
    var name: String
    var songs: [Song]
}

struct ContentView: View {
    @State private var playlists: [Playlist] = [
        Playlist(name: "Любимые плейлисты", songs: []),
        Playlist(name: "Для тренировки", songs: []),
    ]
    @State private var searchQuery: String = ""

    let radioURL = URL(string: "https://radiorecord.hostingradio.ru/chillhouse96.aacp")!
    @State private var isRadioPlaying = false
    @State private var player: AVPlayer?

    var filteredPlaylists: [Playlist] {
        if searchQuery.isEmpty {
            return playlists
        } else {
            return playlists.filter { $0.name.localizedCaseInsensitiveContains(searchQuery) }
        }
    }

    var body: some View {
        NavigationView {
            VStack {
                TextField("Поиск плейлистов", text: $searchQuery)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding()

                List {
                    Button(action: addPlaylist) {
                        HStack {
                            Image(systemName: "plus.circle.fill")
                                .foregroundColor(.green)
                            Text("Добавить плейлист")
                                .foregroundColor(.green)
                        }
```

```
                }
                .buttonStyle(BorderlessButtonStyle())

                ForEach(filteredPlaylists.indices, id: \.self) { index in
                    NavigationLink(destination: PlaylistDetailView(playlistIndex: index, playlists:
$playlists)) {
                        PlaylistRowView(playlist: $playlists[index])
                            .contextMenu {
                                Button(action: {
                                    deletePlaylist(at: index)
                                }) {
                                    Text("Delete")
                                    Image(systemName: "trash")
                                }
                            }
                    }
                }
            }
            .navigationTitle("Мои плейлисты")
            .toolbar {
                ToolbarItem(placement: .bottomBar) {
                    Button(action: {
                        toggleRadio()
                    }) {
                        Text(isRadioPlaying ? "Остановить радио" : "Включить радио")
                    }
                    .foregroundColor(.white)
                    .padding()
                    .background(Color.blue)
                    .cornerRadius(8)
                }
            }
            .overlay(
                Button(action: {
                    if let url = URL(string: "https://bmstu.ru/chair/sistemy-obrabotki-informacii-i-
upravlenia") {
                        UIApplication.shared.open(url)
                    }
                }) {
                    Image(systemName: "questionmark.circle")
                        .foregroundColor(.white)
                        .font(.title)
                        .padding()
                }
                .frame(maxWidth: .infinity, alignment: .leading)
                .padding(.top, 40)
                .padding(.leading, 20)
            )
        }
        .onAppear {
            playRadio()
        }
        .onDisappear {
            stopRadio()
        }
        .foregroundColor(.purple)
        .background(
```

```
                LinearGradient(gradient: Gradient(colors: [Color(hex: "F5DEB3"), Color(hex: "8B4513")]), startPoint: .top, endPoint: .bottom)
                    .edgesIgnoringSafeArea(.all)
            )
        }
    }

    private func addPlaylist() {
        playlists.append(Playlist(name: "Новый плейлист", songs: []))
    }

    private func deletePlaylist(at index: Int) {
        playlists.remove(at: index)
    }

    private func playRadio() {
        let playerItem = AVPlayerItem(url: radioURL)
        self.player = AVPlayer(playerItem: playerItem)
        self.player?.play()
        self.isRadioPlaying = true
    }

    private func stopRadio() {
        self.player?.pause()
        self.isRadioPlaying = false
    }

    private func toggleRadio() {
        if self.isRadioPlaying {
            self.stopRadio()
        } else {
            self.playRadio()
        }
    }
}

struct PlaylistRowView: View {
    @Binding var playlist: Playlist
    @State private var isEditing = false

    var body: some View {
        HStack {
            if isEditing {
                TextField("Название плейлиста", text: $playlist.name)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .onTapGesture {
                        isEditing = true
                    }
                    .onDisappear {
                        isEditing = false
                    }
            } else {
                Text(playlist.name)
                    .onTapGesture {
                        isEditing = true
                    }
            }
```

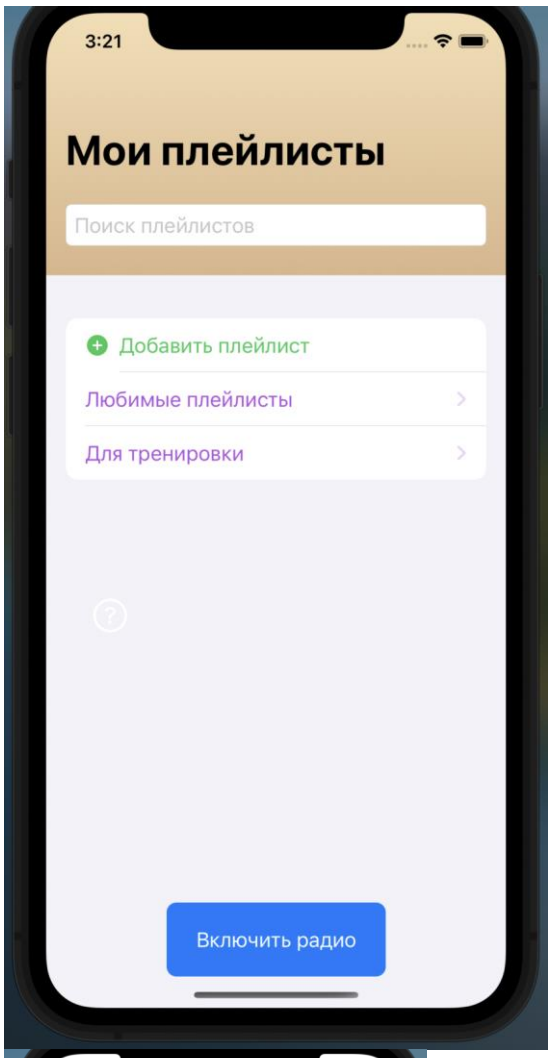
```
        }
    }
}

struct PlaylistDetailView: View {
    var playlistIndex: Int
    @Binding var playlists: [Playlist]

    var body: some View {
        List {
            ForEach(playlists[playlistIndex].songs) { song in
                Text(song.title)
            }
        }
        .navigationTitle(playlists[playlistIndex].name)
        .navigationBarItems(trailing: addButton)
    }

    private var addButton: some View {
        Button(action: addTrack) {
            Label("Добавить трек", systemImage: "plus")
        }
        .foregroundColor(.red)
    }

    private func addTrack() {
        let picker = UIDocumentPickerViewController(forOpeningContentTypes: [UTType.audio])
        picker.allowsMultipleSelection = true
        picker.delegate = Coordinator(parent: self, playlistIndex: playlistIndex)
        UIApplication.shared.windows.first?.rootViewController?.present(picker, animated: true,
completion: nil)
    }

    class Coordinator: NSObject, UIDocumentPickerDelegate {
        var parent: PlaylistDetailView
        var playlistIndex: Int

        init(parent: PlaylistDetailView, playlistIndex: Int) {
            self.parent = parent
            self.playlistIndex = playlistIndex
        }

        func documentPicker(_ controller: UIDocumentPickerViewController, didPickDocumentsAt
urls: [URL]) {
            let selectedSongs = urls.map { url in
                Song(title: url.lastPathComponent)
            }
            self.parent.playlists[self.playlistIndex].songs.append(contentsOf: selectedSongs)
        }
    }
}

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

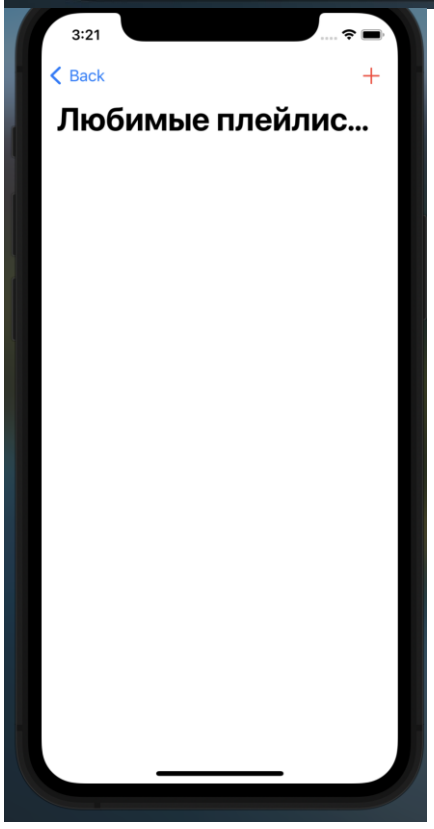
```
extension Color {
    init(hex: String) {
        let scanner = Scanner(string: hex)
        _ = scanner.scanString("#")
        var rgb: UInt64 = 0
        scanner.scanHexInt64(&rgb)

        self.init(
            .sRGB,
            red: Double((rgb & 0xFF0000) >> 16) / 255.0,
            green: Double((rgb & 0x00FF00) >> 8) / 255.0,
            blue: Double(rgb & 0x0000FF) / 255.0
        )
    }
}
```

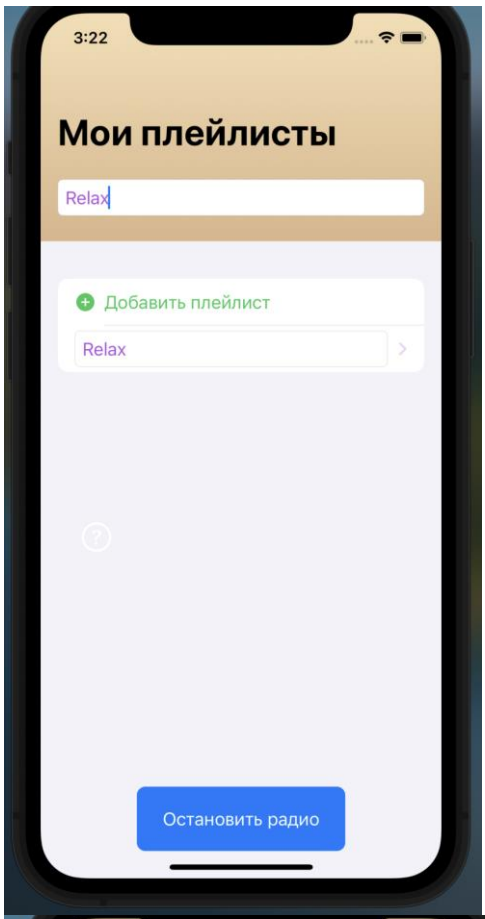
Скриншоты приложения:

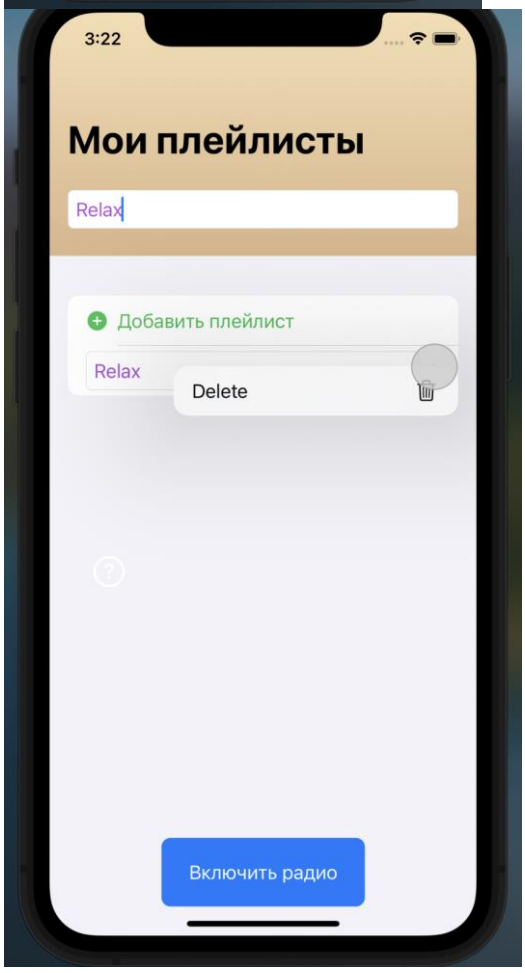
Иконка приложения:

Главное меню:

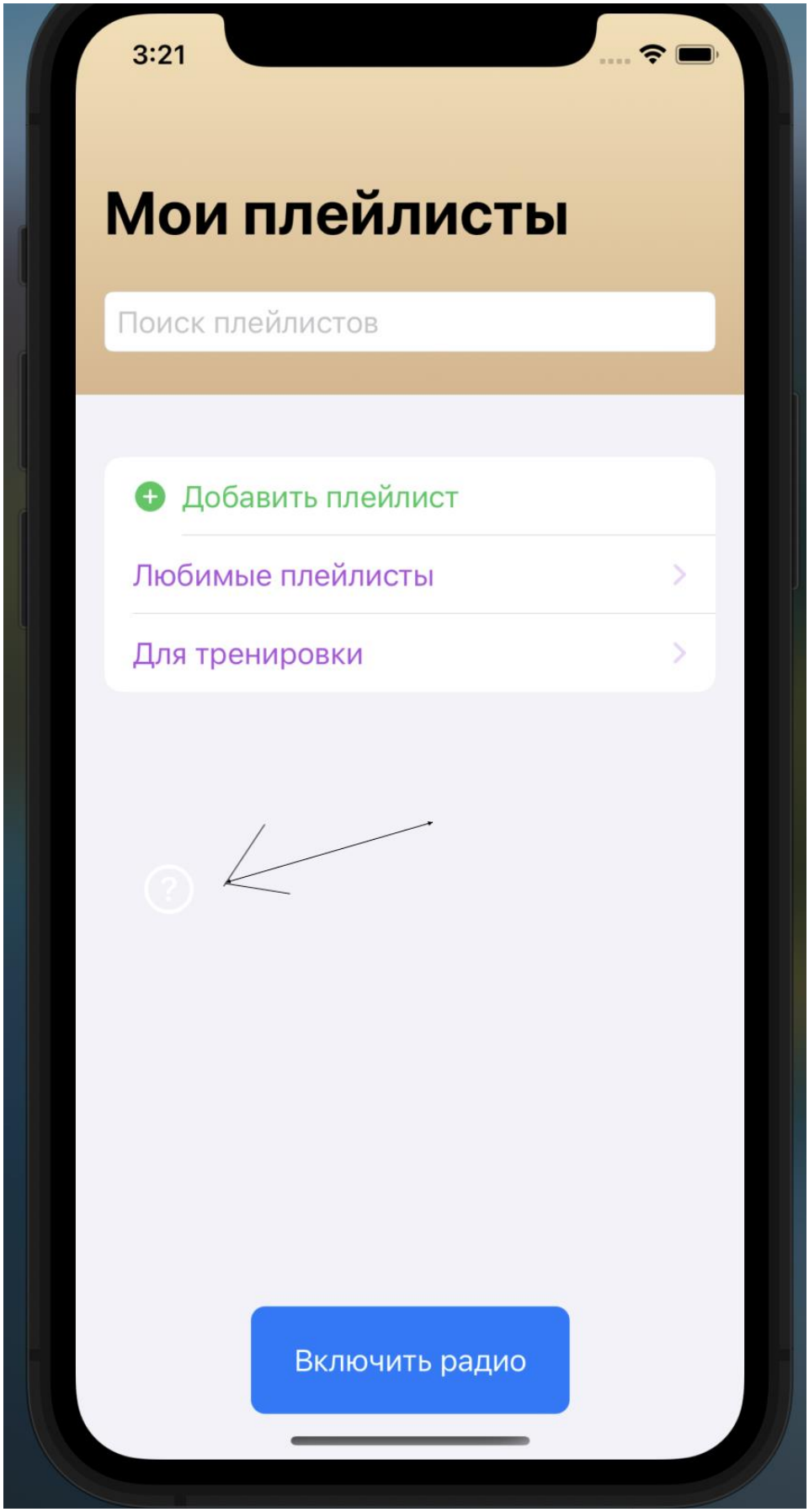

3:21
Мои плейлисты
Поиск плейлистов
Добавить плейлист
Любимые плейлисты
Для тренировки
Включить радио


3:21
Back
Любимые плейлис...


3:22
Мои плейлисты
Relax
Добавить плейлист
Relax
Остановить радио


3:22
Мои плейлисты
Relax
Добавить плейлист
Relax
Delete
Включить радио

Также есть «пасхалка»:



При нажатии на эту кнопку, пользователь оказывается на сайте bmstu.ru: