**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «РАСПИСАНИЕ»

**ТЕКСТ ПРОГРАММЫ**

КП.ПО6.200125-05 12 00

## Листов 10

|  |  |
| --- | --- |
| Руководитель | А. А. Крощенко |
| Выполнил | Д. М. Мартынович |
| Консультант |  |
| по ЕСПД | А. А. Крощенко |
|  |  |

2023

**СОДЕРЖАНИЕ**

ScheduleApp.swift

//

// ScheduleApp.swift

// Schedule

//

// Created by Obi-Van Kenobi on 2.03.23.

//

**import** SwiftUI

**@main**

**struct** ScheduleApp: App {

**var** body: **some** Scene {

WindowGroup {

ContentView()

}

}

}

ContentView.swift

//

// ContentView.swift

// Schedule

//

// Created by Obi-Van Kenobi on 2.03.23.

//

**import** SwiftUI

**struct** ContentView: View {

// Data for the dropdowns

**let** faculties = ["Пусто","ФЭИС", "ФИСЭ", "ЭФ", "МСФ"]

**let** courses = ["Пусто","1 курс", "2 курс", "3 курс","4 курс", "5 курс"]

**let** groups = ["Пусто"]

**let** subGroups = ["Пусто","1 подгруппа","2 подгруппа"]

// Selected values from the dropdowns

@State **private** **var** selectedFaculty = ""

@State **private** **var** selectedCourse = ""

@State **private** **var** selectedGroup = ""

@State **private** **var** selectedSubGroup = ""

@State **private** **var** draw = **false**//animation

**var** availableGroups: [String] {

**switch** (selectedFaculty, selectedCourse) {

**case** ("ФЭИС", "3 курс"):

**return** ["Пусто","ПО6","ПО7"]

**default**:

**return** groups

}

}

**var** body: **some** View {

NavigationView {

VStack {

Text("Расписание")

.font(.custom("Zaychik", size: 65))

.padding(EdgeInsets(top: 100, leading: 0, bottom: 40, trailing: 0))

Spacer()

// Dropdown for faculty selection

Picker("Faculty", selection: $selectedFaculty) {

ForEach(faculties, id: \.**self**) {

Text($0)

}

}

// Dropdown for course selection

Picker("Course", selection: $selectedCourse) {

ForEach(courses, id: \.**self**) {

Text($0)

}

}

.disabled(selectedFaculty.isEmpty)

// Dropdown for group selection

Picker("Group", selection: $selectedGroup) {

ForEach(availableGroups, id: \.**self**) {

Text($0)

}

}

.disabled(selectedCourse.isEmpty)

Picker("subGroup", selection: $selectedSubGroup){

ForEach(subGroups, id: \.**self**){

Text($0)

}

}

.disabled(selectedGroup.isEmpty)

Spacer()

// Next button

NavigationLink(destination: NextView(selectedGroup: selectedGroup, selectedSubGroup: selectedSubGroup)) {

Text("Далее")

.font(.custom("Zaychik", size: 65))

}

.disabled(selectedGroup.isEmpty)

.disabled(selectedSubGroup.isEmpty)

}

.onTapGesture {

// Save selected values to UserDefaults

UserDefaults.standard.set(selectedFaculty, forKey: "SelectedFaculty")

UserDefaults.standard.set(selectedCourse, forKey: "SelectedCourse")

UserDefaults.standard.set(selectedGroup, forKey: "SelectedGroup")

UserDefaults.standard.set(selectedSubGroup, forKey: "SelectedSubGroup")

}

.onAppear(perform: {

// Load selected values from UserDefaults

**if** **let** savedFaculty = UserDefaults.standard.string(forKey: "SelectedFaculty") {

selectedFaculty = savedFaculty

}

**if** **let** savedCourse = UserDefaults.standard.string(forKey: "SelectedCourse") {

selectedCourse = savedCourse

}

**if** **let** savedGroup = UserDefaults.standard.string(forKey: "SelectedGroup") {

selectedGroup = savedGroup

}

**if** **let** savedSubGroup = UserDefaults.standard.string(forKey: "SelectedSubGroup"){

selectedSubGroup = savedSubGroup

}

downloadDatabase()

})

}

}

}

**struct** ContentView\_Previews: PreviewProvider {

**static** **var** previews: **some** View {

ContentView()

}

}

NextView.swift

//

// NextView.swift

// Schedule

//

// Created by Obi-Van Kenobi on 2.03.23.

//

**import** SQLite3

**import** SwiftUI

**import** Foundation

**enum** Tabs : String {

**case** Сегодня

**case** Расписание

}

**struct** NextView: View {

@State **var** selectedTab: Tabs = .Сегодня

**var** selectedGroup: String // группа выбранная на первой странице

**var** selectedSubGroup: String //погруппа выбранная на первой странице

@State **private** **var** selectedWeekType = getAutoWeekType() // выбранный тип недели

//@State private var lessons = [Lesson]()// array to store lesson names

@State **private** **var** autoWeektype = getAutoWeekType()

**var** body: **some** View {

TabView(selection: $selectedTab) {

// Вкладка с контентом

TodayView(selectedGroup: selectedGroup, selectedSubGroup: selectedSubGroup)

.tabItem {

Image(systemName: "star")

Text("Текущий день")

}.tag(Tabs.Сегодня)

// Вкладка с настройками

ScheduleView(selectedGroup: selectedGroup, selectedSubGroup: selectedSubGroup)

.tabItem {

Image(systemName: "house.fill")

Text("Вся неделя")

}.tag(Tabs.Расписание)

//Вкладка с сайта

NewsWebView(urlString: "https://news.bstu.by")

.tabItem{

Image(systemName: "display.and.arrow.down")

Text("Новости")

}

}

.navigationBarHidden(**true**)

}

}

**struct** NextView\_Previews: PreviewProvider {

@State **static** **var** selectedGroup = "ПО6"

@State **static** **var** selectedSubGroup = "1 подгруппа"

**static** **var** previews: **some** View {

NextView(selectedGroup:selectedGroup, selectedSubGroup: selectedSubGroup)

}

}

**func** getCurrentWeekName() -> String {

**let** dateFormatter = DateFormatter()

dateFormatter.dateFormat = "EEEE"

dateFormatter.locale = Locale(identifier: "ru\_RU")

**let** currentWeekdayName = dateFormatter.string(from: Date())

**return** currentWeekdayName.capitalized

}

**enum** Weekday: String, CaseIterable {

**case** monday = "Понедельник"

**case** tuesday = "Вторник"

**case** wednesday = "Среда"

**case** thursday = "Четверг"

**case** friday = "Пятница"

**case** saturday = "Суббота"

}

**func** getAutoWeekType() -> String {

**let** calendar = Calendar.current

**let** currentWeekNumber = calendar.component(.weekOfYear, from: Date())

//print(currentWeekNumber)

**let** currentWeekType = currentWeekNumber % 2 == 0 ? "Верхняя неделя" : "Нижняя неделя"

**return** currentWeekType

}

TodayView.swift

//

// TodayView.swift

// Schedule

//

// Created by Obi-Van Kenobi on 5.04.23.

//

**import** SQLite3

**import** SwiftUI

**import** Foundation

**import** UserNotifications

**struct** TodayView: View {

@Environment(\.presentationMode) **var** presentationMode

**var** selectedGroup: String // группа выбранная на первой странице

**var** selectedSubGroup: String

@State **private** **var** selectedWeekType = getAutoWeekType() // выбранный тип недели

@State **private** **var** lessons = [Lesson]()// array to store lesson names

@State **private** **var** autoWeektype = getAutoWeekType()

**var** body: **some** View {

NavigationView{

VStack {

Text(getCurrentWeekName())

.bold()

.font(.none)

.foregroundColor(.secondary)

.frame(

maxWidth: .infinity,

alignment: .topLeading

)

.padding(.leading)

Text(typeText())

.font(.none)

.bold()

.foregroundColor(.secondary)

.frame(

maxWidth: .infinity,

alignment: .topLeading

)

.padding(.leading)

List(lessons, id: \.**self**) { lesson **in**

VStack(alignment: .leading) {

Text("\(lesson.subjectName)")

Text("\(lesson.time), \(lesson.roomNumber), \(lesson.sybjectType), \(lesson.teacher)")

.font(.caption)

.foregroundColor(.secondary)

}

}

}

.navigationTitle("Сегодня")

.navigationBarTitleDisplayMode(.inline)

.navigationBarItems(leading:

Button(action: {

presentationMode.wrappedValue.dismiss()

}, label: {

Image(systemName: "chevron.backward")

Text("Назад")

})

)

}

.onAppear {

loadLessons()

}

.onChange(of: selectedWeekType) { \_ **in**

loadLessons()

}

}

**func** loadLessons() {

lessons = []

**if** **let** db = connectToDatabase() {

**let** query = """

SELECT subject\_name, time, room\_number, subject\_type, teacher

FROM schedule

WHERE day\_of\_week = '\(getCurrentWeekName())'

AND week\_type = '\(autoWeektype)'

AND group\_name = '\(selectedGroup)'

AND sub\_group\_name = '\(selectedSubGroup)'

"""

**var** statement: OpaquePointer?

**if** sqlite3\_prepare\_v2(db, query, -1, &statement, **nil**) != SQLITE\_OK {

**let** errmsg = String(cString: sqlite3\_errmsg(db)!)

print("error preparing select: \(errmsg)")

**return**

}

**while** sqlite3\_step(statement) == SQLITE\_ROW {

**if** **let** subjectName = sqlite3\_column\_text(statement, 0),

**let** time = sqlite3\_column\_text(statement, 1),

**let** roomNumber = sqlite3\_column\_text(statement, 2),

**let** subjectType = sqlite3\_column\_text(statement, 3),

**let** teacher = sqlite3\_column\_text(statement, 4){

**let** lesson = Lesson(

subjectName: String(cString: subjectName),

time: String(cString: time),

roomNumber: String(cString: roomNumber),

sybjectType: String(cString: subjectType),

teacher: String(cString: teacher)

)

lessons.append(lesson)

}

}

sqlite3\_finalize(statement)

sqlite3\_close(db)

}

scheduleNotifications()

}

**func** scheduleNotifications() {

**let** center = UNUserNotificationCenter.current()

center.requestAuthorization(options: [.alert, .sound]) { granted, error **in**

**if** **let** error = error {

print("Error requesting authorization for user notifications: \(error.localizedDescription)")

**return**

}

**if** granted {

center.removeAllPendingNotificationRequests() // Удалить все предыдущие уведомления, если они есть

**for** lesson **in** lessons {

**let** content = UNMutableNotificationContent()

content.title = "Начало пары"

content.body = "\(lesson.subjectName) в \(lesson.time) в аудитории \(lesson.roomNumber)"

content.sound = UNNotificationSound.default

**let** timeComponents = lesson.time.components(separatedBy: ":")

**var** hour = Int(timeComponents[0]) ?? 0

**var** minute = Int(timeComponents[1].prefix(2)) ?? 0

minute -= 30 // уменьшаем значение на 30 минут

**if** minute < 0 {

hour -= 1

minute += 60

}

**var** dateComponents = Calendar.current.dateComponents([.year, .month, .day, .hour, .minute, .second], from: Date())

dateComponents.hour = hour

dateComponents.minute = minute

**let** trigger = UNCalendarNotificationTrigger(dateMatching: dateComponents, repeats: **false**)

**let** request = UNNotificationRequest(identifier: UUID().uuidString, content: content, trigger: trigger)

center.add(request)

}

}

}

}

}

**struct** Lesson: Hashable {

**let** subjectName: String

**let** time: String

**let** roomNumber: String

**let** sybjectType: String

**let** teacher: String

}

**func** typeText() -> String {

**if** getCurrentWeekName() == "Воскресенье" {

**return** "Выходной"

}

**else**{

**return** getAutoWeekType()

}

}

ScheduleView.swift

//

// ScheduleVIew.swift

// Schedule

//

// Created by Obi-Van Kenobi on 5.04.23.

//

**import** SwiftUI

**import** SQLite3

**import** Foundation

**struct** ScheduleView: View {

**var** selectedGroup: String // группа выбранная на первой странице

**var** selectedSubGroup: String

@State **private** **var** selectedWeekType = getAutoWeekType() // выбранный тип недели

//@State private var lessons = [Lesson]()// array to store lesson names

@State **private** **var** autoWeektype = getAutoWeekType()

**var** body: **some** View {

NavigationView{

List {

Picker("Week Type", selection: $selectedWeekType) {

Text("Верхняя неделя").tag("Верхняя неделя")

Text("Нижняя неделя").tag("Нижняя неделя")

}.pickerStyle(SegmentedPickerStyle())

ForEach(Weekday.allCases, id: \.**self**) { weekday **in**

Section(header: Text(weekday.rawValue)) {

ForEach(getLessonsForWeekday(weekday: weekday), id: \.**self**) { lesson **in**

VStack(alignment: .leading) {

Text("\(lesson.subjectName)")

Text("\(lesson.time), \(lesson.roomNumber), \(lesson.sybjectType), \(lesson.teacher)")

.font(.caption)

.foregroundColor(.secondary)

}

}

}

}

}

.navigationTitle("Расписание")

}

}

**func** getLessonsForWeekday(weekday: Weekday) -> [Lesson] {

**var** lessons = [Lesson]()

**if** **let** db = connectToDatabase() {

**let** query = """

SELECT subject\_name, time, room\_number, subject\_type, teacher

FROM schedule

WHERE day\_of\_week = '\(weekday.rawValue)'

AND week\_type = '\(selectedWeekType)'

AND group\_name = '\(selectedGroup)'

AND sub\_group\_name = '\(selectedSubGroup)'

"""

**var** statement: OpaquePointer?

**if** sqlite3\_prepare\_v2(db, query, -1, &statement, **nil**) != SQLITE\_OK {

**let** errmsg = String(cString: sqlite3\_errmsg(db)!)

print("error preparing select: \(errmsg)")

**return** lessons

}

**while** sqlite3\_step(statement) == SQLITE\_ROW {

**if** **let** subjectName = sqlite3\_column\_text(statement, 0),

**let** time = sqlite3\_column\_text(statement, 1),

**let** roomNumber = sqlite3\_column\_text(statement, 2),

**let** subjectType = sqlite3\_column\_text(statement, 3),

**let** teacher = sqlite3\_column\_text(statement, 4){

**let** lesson = Lesson(

subjectName: String(cString: subjectName),

time: String(cString: time),

roomNumber: String(cString: roomNumber),

sybjectType: String(cString: subjectType),

teacher: String(cString: teacher)

)

lessons.append(lesson)

}

}

sqlite3\_finalize(statement)

sqlite3\_close(db)

}

**return** lessons

}

}

NewsWebView.swift

**import** SwiftUI

**import** WebKit

**struct** NewsWebView: UIViewRepresentable {

**let** urlString: String

**func** makeUIView(context: Context) -> WKWebView {

**let** webView = WKWebView()

**if** **let** url = URL(string: urlString) {

**let** request = URLRequest(url: url)

webView.load(request)

}

**return** webView

}

**func** updateUIView(\_ webView: WKWebView, context: Context) {}

}

DatabaseHelper.swift

**import** Foundation

**import** SQLite3

**func** connectToDatabase() -> OpaquePointer? {

**var** db: OpaquePointer?

**let** fileURL = **try**! FileManager.default

.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: **nil**, create: **false**)

.appendingPathComponent("university.db")

**let** path = fileURL.path

**if** sqlite3\_open(path, &db) != SQLITE\_OK {

print("error opening database")

**return** **nil**

} **else** {

//print("database opened successfully")

//print("Database path: \(path)")

**return** db

}

}

**func** downloadDatabase() {

**guard** **let** url = URL(string: "https://github.com/iKrut0nardo/BRSTU\_PO/raw/sn0w/university.db") **else** {

print("Invalid URL")

**return**

}

**let** task = URLSession.shared.downloadTask(with: url) { (location, response, error) **in**

**guard** **let** location = location **else** {

print("Download failed: \(error?.localizedDescription ?? "Unknown error")")

**return**

}

**let** fileManager = FileManager.default

**let** documentsUrl = fileManager.urls(for: .documentDirectory, in: .userDomainMask).first!

**let** destinationUrl = documentsUrl.appendingPathComponent("university.db")

**do** {

// Remove old database if exists

**if** fileManager.fileExists(atPath: destinationUrl.path) {

**try** fileManager.removeItem(at: destinationUrl)

print("Old data removed")

}

// Move downloaded database to documents directory

**try** fileManager.moveItem(at: location, to: destinationUrl)

print("Database downloaded.")

} **catch** {

print("Error moving downloaded database: \(error.localizedDescription)")

}

}

task.resume()

}