МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ И СОЗДАНИЕ ЭКРАНОВ С РАСПИСАНИЕМ ДЛЯ ПРИЛОЖЕНИЯ УНИВЕРСИТЕТА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Швеца Антона Эдуардовича

Научный руководитель
к. п. н., доцент _____ В. А. Векслер

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2021

ВВЕДЕНИЕ

В жизни современного студента неотъемлемую часть занимает мобильное устройство. Оно помогает студенту пользоваться учебниками и электронными материалами университета. Помогает сохранять информацию, участвовать в удаленных занятиях. Многие из этих функций, ученик выполняет в различных приложениях, которые направлены на предоставление удобного рабочего пространства. Некоторые ресурсы и инструменты также доступны для студента, но находятся на удаленных веб сайтах. Одним из них является сайт СГУ, который является источником информации для студентов и преподавателей. Каждый день студенты пользуются им, чтобы получить актуальную информацию о расписании, но не всегда бывает удобно заходить на него и искать нужную группу во время трудового дня, тем более когда похожие действия приходится делать каждый день. С этой проблемой может справиться реализация собственного приложения для сайта. В нем студенты и преподаватели смогут находить свое расписание, сохранять его. Также будут доступны и другие разделы сайта. Реализация такого проекта принесет новые возможности и сократит время, которое люди тратят на пользование официальным сайтом СГУ через мобильное устройство.

Таким образом, целью выпускной квалификационной работы является создание части мобильного приложения под операционную систему IOS для реализации функционала поиска расписания для университета.

Из цели вытекают следующие задачи:

- исследовать область мобильной разработки под операционную систему IOS;
- создать сервисы для работы с сервером и получения данных;
- создать экран поиска преподавателя для поиска преподавателя по введенному параметру;
- создать экраны вывода расписания для отображения расписания студента, расписания преподавателя, сохраненного расписания;
- протестировать разработанную часть приложения с помощью симуляторов.

Структуры работы состоит из двух глав:

1. Средства разработки мобильных приложений под ОС IOS;
2. Разработка мобильного приложения.

1 Средства разработки мобильных приложений под ОС IOS

1.1 Требования к проекту

Мобильное приложение, дублирующее сайт университета, должно удовлетворять ряду требований:

- приложение должно быть написано для операционную IOS;
- должна присутствовать удобная навигация и интуитивно понятный интерфейс;
- все экраны должны иметь единый стиль оформления;
- все экраны должны быть выполнены в едином цветовом стиле;
- в нижней части экрана должна располагаться панель с тремя вкладками: «расписание», «новости», «меню»;
- необходимо наличие сервисов отправки запросов на сервер для получения списка преподавателей, факультетов, групп и расписания;
- при нажатии на кнопку «расписание» в нижней панели вкладок:
  - в случае наличия закэшированного расписания должен открываться экран вывода сохраненного расписания;
  - иначе, экран–заглушка с сообщением.
- при нажатии на кнопку «меню» должен открываться экран меню с пунктами: «Расписание студентов», «Расписание преподавателей»;
- при выборе в меню пункта «Расписание преподавателей» должен открываться экран поиска преподавателя;
- экран поиска преподавателя должна состоять из окна ввода параметра поиска и таблицы с выводом найденных преподавателей;
- при выборе преподавателя на экране поиска преподавателя должен открываться экран с расписанием выбранного преподавателя;
- экран вывода расписания преподавателя должен отображать инициалы выбранного преподавателя и его расписание на неделю;
- при выборе в меню пункта «Расписание студентов» должен отображаться экран выбора курса и факультета, а затем экран выбора номера группы. После выбора номера группы должен открываться экран вывода расписания группы студентов;
- экран расписания группы студентов должен содержать номер группы и расписание указанной группы на неделю;

1.2 Операционная система IOS

Поскольку приложение будет создаваться для мобильных устройств с операционной системой IOS, то познакомимся с ней поближе.

IOS — операционная система для мобильных устройств, разрабатываемая компанией Apple. Данная операционная система является актуальной и современной, поскольку обновления для нее компания Apple публикует каждый год начиная с 2007 года. В связи с этим IOS широко используется как для личного использования так и в различных отраслях.

Изначально для проектирования приложений по ОС IOS использовался язык Objective–C, но начиная с 2014 года для этого используется язык программирования Swift.

1.3 Язык программирования Swift

Swift — открытый мультипарадигмальный компилируемый язык программирования для написания приложений под устройства с операционной системой IOS, MacOS, WatchOS, TvOS.

Специалисты выделяют следующие преимущества языка Swift:

- актуальный — последний выпуск языка: 26 апреля 2021 года;
- безопасный — аккуратно работает с опциональными значениями;
- простой для изучения, поскольку имеет простой и лаконичный синтаксис;
- быстрый — предназначен для замены языков на основе C и должен быть сопоставим с этими языками по производительности.

Для компиляции приложений на языке Swift используется компилятор Xcode.

1.4 Среда разработки Xcode

Xcode — интегрированная среда разработки программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple. Чтобы добавить данную программу на устройство, ее необходимоо скачать из App Store. Поддерживает языки C, C++, Objective-C, Swift, Java, AppleScript, Python и Ruby. Последняя версия Xcode 12 была выпущена 22 июня 2020 года.

Среда разработки Xcode — единственный компилятор для языка Swift под мобильную платформу IOS. Компания Apple тщательно оберегает зашифрованные коды компилирования и запрещает Xcode передавать их третьим

лицам. В связи с этим разработка приложений по ОС IOS возможна только на компьютерах и ноутбуках, произведенных компанией Apple, а скачать приложение Xcode можно только из официального магазина приложений AppStore.

1.5 Архитектура Rx + MVVM

Для использования реактивного программирования в проекте будет использоваться архитектура Rx. Это позволит упростить и улучшить структуру проекта. Легче всего реактивное программирование будет объяснить через паттерн «Наблюдатель». В данном паттерне присутствует класс «издатель», публикующий свои изменения, и классы «подписчики», подписывающиеся на публикуемые издателем изменения. Таким образом, при реактивном программировании действия пользователя в приложении отслеживаются и в зависимости от заложенной программы, система реагирует. Реактивное программирование позволяет ускорить и оптимизировать проектирование архитектуры приложения и дает возможность быстрее и качественнее создавать экраны, а также позволяет улучшить происходящие внутри процессы.

Архитектура Rx + MVVM состоит из двух основных реактивных классов и нескольких вспомогательных структур. Основными являются классы ViewModel (далее модель) и ViewController (далее контроллер). Начнем с контроллера, его задача — отслеживать события — действия пользователя (например, нажатие кнопки) и отправлять модели информацию о них. После получения информации модель обрабатывает эти данные и возвращает их в контроллер, который отображает эти данные пользователю на экране.

Для общения модели и контроллера задаются вспомогательные структуры: input, output. Структура input передается от контроллера модели и содержит информацию о событиях, а структура output передается от модели контроллеру и содержит выходные данные, полученные после обработки входных данных.

Подобная архитектура предоставляет широкие возможности настройки и является достаточно гибкой, что позволяет настроить ее для решения различных видов задач. Одним из способов такой настройки является дополнительная структура — DataSource, содержащая в себе описание конфигурации данных, получаемых контроллером из модели и отображаемых пользователю на экране. Данная структура также обрабатывает и настраивает передаваемые данные.

1.6 Библиотеки (Pods)

Чтобы проект смог выполнять требуемые функции, в него необходимо импортировать сторонние библиотеки с открытым исходным кодом. Для скачивания и использования таких библиотек используется менеджер зависимостей — Cocoa Pods, написанный на языке Ruby.

2 Разработка мобильного приложения

2.1 Настройка структуры проекта

Создадим файловую структуру проекта. Основной папкой проекта является папка «SSUMobile», поскольку в ней содержаться все модули с экранами, сервисы, переходы, ячейки и прочее. Внутри создадим следующие разделы:

- Раздел Flow для хранения всех переходов между экранами;
- Раздел Service будет содержать файл LocalUserStorage. Он будет хранить номер группы, наименование факультета, тип сохраненного расписания (расписание преподавателя или студента), идентификатор преподавателя и полное имя преподавателя. Хранение указанных значений в хранилище приложения (на устройстве пользователя) позволяет получать расписание в зависимости от сохраненных значений и выводить его, освобождая пользователя от необходимости поиска расписания при каждом открытии приложения. Кроме этого раздел Service будет содержать настройки запросов в сеть для получения таких данных как список факультетов, расписание группы, список факультетов, расписание указанного преподавателя, список групп для заданного курса и факультета;
- Раздел Managers будем использовать для того, чтобы вынести некоторую бизнес–логику, которая нагружает ViewModel, либо которая дублируется в разных ViewModel;
- Раздел Common будет состоять из классов для работы с загрузкой и сериализацией данных из json–файла в модель проекта;
- В раздел Models сложим все базовые сущности проекта;
- Раздел Modules будет содержать основные файлы для каждого из экранов приложения. Каждый экран имеет представление View, состоящее из двух файлов–контроллеров storyboard для верстки и swift для изменения объектов экрана с помощью кода и отслеживания действий пользователя. Также каждый экран содержит раздел ViewModel, внутри которого расположена модель ViewModel, занимающаяся обработкой входящих данных, структура ViewModelInput, которая содержит формат данных, приходящих от контроллера, и структура ViewModelOutput, которая содержит формат данных, отправляемых контроллеру. Ну и наконец, каждый экран имеет раздел DataSource, содержащий одноименный файл, в котором содержится описание конфигурации данных, получаемых кон-

троллером из модели;

— Раздел Components будет содержать описание ячеек для таблиц и коллекций (подразделы, заканчивающиеся на Cell), а также описание нижней панели вкладок и расписания. Описания ячеек состоят из файла типа xib, используемого для верстки дизайна, а также из файла swift, используемого для настройки и заполнения ячейки.
— Раздел Common будет содержать различные вспомогательные сущности. Например, класс Pallete, содержащий использующиеся в приложении цвета, заданные в виде константных значений.

2.2 Настройка навигации между экранами (RxFlow)

В любом мобильном приложении требуется настройка навигации, без нее мы не сможем перемещаться между экранами. В данном проекте для настройки навигации будем использовать библиотеку с открытым исходным кодом — RxFlow, которая позволяет легко и удобно настроить любые переходы между экранами приложения. Одним из основных классов настройки навигации будет файл AppStep. Он представляет из себя перечисление — enum, в котором указаны все возможные шаги. Шаг — это единичный переход между экранами приложения. После определения набора шагов создадим базовый Flow приложения. Flow — это набор экранов, которые входят в один или несколько стеков навигации и могут быть объединены в один компонент. Базовым первоначальным Flow является AppFlow. В нем происходит инициализация корневого окна отображения. Затем запускается новый конкретный Flow приложения со своим стеком навигации.

В проекте создано 4 дополнительных Flow: TabBarFlow позволяет настраивать переходы между экранами по нажатию кнопок в нижней панели вкладок; TimetableFlow настраивает переходы между экранами поиска и вывода расписания; следующий набор экранов — NewFlow. Он настраивает переходы для экранов вывода новостей; MenuFlow используется для настройки переходов экрана меню.

2.3 Настройка сервисов для отправки запросов на сервер

Для получения данных существует два способа: сетевой слой, позволяющий отправлять запросы на реальный сервер и преобразовывать ответ сервера в объекты классов проекта; mock–слой, позволяющий получать данные

без отправки запросы на сервер. Данные берутся из json–файлов, написанных заранее и имеющих такую же структуру, как и json–файлы приходящие с сервера. Такие файлы сохраняются в папке проекта и при необходимости обрабатываются и преобразовываются объекты классов проекта. Использование mock–данных позволяет быстрее и удобнее протестировать приложение.

Рассмотрим подробнее настройку сервисов проекта. Настройкой сетевого слоя занимается библиотека Moya. Она на низком уровне посылает HTTP–запросы на сервер по заданному URL–адресу. От разработчиков требуется только наладить каждое ApiTarget. Внутри файла ApiTarget необходимо указать настройки: базовый URL сервера, конкретный путь запроса, тип запроса (POST, GET, DELETE, и т.д.), параметры запросы (если они требуются), заголовок запроса и прочее. После описание таргетов запросов добавляем обертку над вызовом методов в сущности сервиса. В нем мы отправляем конкретный запрос, фильтрацию ошибок, преобразование в локальную модель и преобразование в реактивный объект типа Single для возвращения в модель.

Рассмотрим подробнее имеющиеся сервисы.

Сервис преподавателей «TeacherService» содержит следующие методы: teachers помогает находит преподавателя по заданному в виде строки параметру. Используется на экране поиска преподавателя; timelesson помогает находить расписание преподавателя по его Id. Используется на экране расписания преподавателя при поиске расписания.

Сервис факультетов «DepartmentService» содержит следующие методы: allDepartments помогает получать список факультетов. Используется на экране выбора курса и факультета для группы; groupList помогает получить список групп для заданного факультета и курса. Используется при открытии экрана со списком групп; timelesson помогает получать расписание группы студентов для заданной группы и факультета. Используется при открытии экрана вывода расписания группы.

2.4 Настройка сервис локатора (DI)

При проектировании архитектуры сервисов хорошим подходом является наличие DI (Dependency Injection). Главной его задачей является регистрация, хранение, а также возврат единственного созданного экземпляра класса–сервиса. Для того, чтобы при каждом обращении не создавать сервис заново и контролировать настройку зависимостей в каждой модели ViewModel, бу-

дем хранить все сервисы в одном месте — в DI контейнере под названием ServiceLocator.Такая структура требует, чтобы добавление сервисов в контейнер происходило только в отдельном методе при запуске приложения. Это необходимо для однозначности, чтобы сервисы не создавались повторно и беспорядочно. Создадим все необходимые сервисы и произведем их регистрацию в сервис локаторе на основе их протоколов. Создадим и привяжем сервисы, все они хранятся в одном месте и однозначно соответствуют своему протоколу, а также настраиваются единожды.

2.5 Создание экранов с расписанием

Перейдем к созданию экранов. Одной их основных задач данного приложения является отображение расписания для студентов и преподавателей. Для реализации работы с расписанием используются 3 экрана:

1. Расписание в нижней панели вкладок — стандартный экран, открывающийся при нажатии кнопки «Расписание» в нижней панели вкладок. В нем отображается актуальная версия выбранного ранее расписания;
2. Расписание преподавателя открывается после клика по ФИО преподавателя на экране выбора преподавателя. Для получения расписания используется менеджер для работы с преподавательским расписанием;
3. Расписание студента открывается после клика по номеру группы на экране выбора группы. По визуальным компонентам является одинаковым с расписанием преподавателя. Главным отличием между этими экранами будет используемый менеджер. Для данного экрана используется менеджер для работы со студенческим расписанием.

2.5.1 Визуальный компонент экрана «Расписание»

Все три экрана объединены общей компонентой — ячейкой таблицы расписания. По своей структуре ячейка расписания является сложной компонентой. Она состоит еще из нескольких ячеек. Итак, рассмотрим подробнее полученную структуру ячеек. Начнем с более общих ячеек и перейдем к внутренним ячейкам.

Внутри раздела TimetableMainTableViewCell содержатся две основные ячейки для отображения расписания. TimetableView будет содержать коллекцию ячеек типа TimetableContainerCollectionViewCell. Коллекция позволит организовать на экране перелистывание расписания влево и вправо.

Перейдем к разделу TimetableContainerViewCell. Здесь описывается длинная ячейка на которой будут располагаться дни недели. Для каждого дня недели добавлен отдельный элемент–контейнер View, внутри которого будет использоваться дополнительная ячейка TimetableDayView.

Перейдем к ячейке для отображения пар на один день — TimetableDayView. Она содержит заголовок для вывода дня недели и таблицу, внутри которой лежит список ячеек с парами типа TimetableLessonViewCell.

Рассмотрим ячейку TimetableLessonViewCell. Она необходима для отображения пары, в ней отображаются все данные по одной паре: сверху располагается время начала и время конца пары, затем тип пары (практика или лекция), четность пары (по числителям или знаменателям), название предмета, ФИО преподавателя, номер корпуса университета и номер аудитории.

Файл EmptyTimetableLessonView содержит ячейку для отображения дня, когда пар нет. Ячейка состоит только из заголовка.

Стоит отметить, что весь этот сложный элемент полностью располагается в отдельной ячейке на таблице, которая обеспечивает всему расписанию вертикальную прокрутку (вверх–вниз). Это необходимо в случае использования устройства с маленькой диагональю экрана.

2.5.2 Настройка главного экрана расписания

Перейдем к основному экрану вывода расписания. Для него, внутри раздела Modules, определен подраздел TimetableScreen. Экран будет состоят из разделов: View, содержащего контроллер экрана, ViewModel, содержащего модель для обработки действий пользователя на экране, и DataSource, используемого для дополнительной настройки общения контроллера и модели.

Начнем с раздела с контроллером — View. Визуальное представление экрана располагается в файле
TimetableViewController.storyboard. Главный экран расписания состоит из таблицы — TableView, которая используется для отображения расписания. Перейдем к файлу TimetableViewController.swift, в котором создадим переменную tableView для таблицы и зададим свойство невидимости, чтобы показывать стандартный экран–заглушку при отсутствии расписания; а также добавим отслеживание действий пользователя на экране. Как только расписание будет найдено и сохранено, свойству isHidden присвоится значение свойство false и таблица с расписанием станет видимой. Для верхней панели View созда-

дим переменную emptyView и зададим цвет фона, а для заголовка создадим переменную emptyTitle и зададим цвет шрифта, сам шрифт и текст.

Перейдем к модели расписания — TimetableViewModel. В данном случае модель занимается проверкой наличия добавленного расписания при открытии экрана с сохраненным расписанием. Рассмотрим ее подробнее. Начнем с инициализатора. Внутри него сохраняются передаваемые менеджеры: previewTimetableManager — менеджер для работы с расписанием студентов, teacherTimetableManager — менеджер для работы с расписанием преподавателя, savedTimetableManager — менеджер для работы с сохраненным расписанием. После их инициализации вызывается функция проверки наличия сохраненного расписания — checkSavedTimetable. Внутри нее из менеджера savedTimetableManager получаем тип сохраненного расписания. По типу определяем дальнейшие действия: если сохранено расписание для студенческой группы, то делаем таблицу видимой; если сохранено расписание для преподавателя, то вызываем функцию загрузки расписания преподавателя — loadTimetableForSavedTeacher и делаем таблицу с расписанием видимой; иначе делаем таблицу с расписанием невидимой.

Основной функцией модели является загрузка расписания. Рассмотрим подробнее сам процесс загрузки расписания преподавателя. Для этого откроем функцию loadTimetableForSavedTeacher в классе модели. В первой строке проверим наличие сохраненного идентификатора преподавателя. Если окажется, что идентификатор еще не сохранен, то вернем пустое значение, иначе продолжим. Затем получим ФИО преподавателя, причем, если оно не указано будет выведено сообщение об этом.

Перейдем к загрузке расписания, для этого у менеджера расписания преподавателей — teacherTimetableManager вызовем метод loadFullTimelesson с заданным идентификатором преподавателя. После получения ответа с сервера обработаем список пар и преобразуем данные в модели проекта. Внутри метода преобразования найдем максимальное количество пар в день, чтобы задать высоту ячейки для одного дня. В конце вызова вернем список типа MainTimetableSectionType и передадим в него ФИО преподавателя и список пар на неделю.

Вернемся к контроллеру и рассмотрим его основной метод viewDidLoad. Настроим таблицу с помощью метода configureTableView, где укажем ячейки

таблицы, отключим индикатор пролистывания, зададим отступы. После настройки таблицы зададим ее отображение. Из выходного параметра возьмем параметр isDiplayTimetable, который позволит определить показывать ли таблицу на экране и для кого. Если значение true, то скрываем стандартный экран–заглушку с помощью hideEmptyView (он меняет значение параметра isHidden у экрана–заглушки на true) и показываем экран расписания с помощью displayTableView (меняем значение параметра isHidden на false для таблицы с расписанием), иначе с помощью displayEmptyView делаем стандартный экран–заглушку видимым (меняем значение параметра isHidden на false для экрана–заглушки).

Запустим приложение для устройства iPhone 8, чтобы проверить, как экран будет отображаться на маленьких устройствах. Сохраним расписание преподавателя на экране поиска и откроем экран с сохраненным расписанием. Увидим, что данные для выбранного преподавателя отображаются корректно. Попробуем найти расписание на среду. Таблица удобно листается влево и вправо. Просмотрим все пары на один день, пролистнув вниз, увидим, что таблица легко пролистывается вниз, а также вверх.

2.5.3 Создание экранов расписания преподавателя и студента

Для экрана расписания преподавателя и экрана расписания студента определены только контроллер — View и модель — ViewModel, поскольку структура DataSource для них заимствуется из главного экрана расписания.

Для экрана студента модель ViewModel настраивается с помощью параметров факультета и курса, которые пользователь выбирает на экране выбора курса и факультета, а также номера группы, который пользователь выбирает на экране выбора номера группы. Для экрана преподавателя модель настраивается с помощью идентификатора преподавателя, которого пользователь выбрал на экране поиска преподавателя.

После конфигурации ViewModel обращается к соответствующему менеджеру и загружает через него необходимое расписание (для преподавателя — teacherTimetableManager, а для студента — previewTimetableManager), а затем форматирует и передает отображение в контроллер — View. Поскольку эти методы схожи для обоих экранов рассмотрим такой метод только у одного из них. Например, возьмем экран расписания студента. Такой метод называется loadTimetable. У менеджера вызывается метод отправки запроса на сервер

для получения расписания студента — getTimetable. В качестве входных параметров передаются имеющиеся значения факультета и группы. После получения ответа от сервера, полученные данные обрабатываются. Для начала полученное расписание проверяется на наличие, а затем находится максимальное количество пар за день, это необходимо чтобы найти высоту для таблицы. После чего настраивается модель и расписание добавляется в перечисление SectionType и возвращается как результат. При создании данные преподавателя заполняются пустыми значениями, поскольку мы искали расписание для студента и этих данных у нас нет, значение номера группы заполняется соответствующим входным параметров, а расписание записывается в перечисление основной таблицы расписания MainTimetableSectionType.

Перейдем к контроллеру — View экранов. Дело в том, что данные экраны являются одинаковыми с точки зрения визуальных компонентов, поэтому рассмотрим настройку только для одного из них.

Запустим приложение для устройства iPhone 12, выберем на экране меню пункт «Расписание преподавателя», а затем найдем преподавателя на экране поиска. После нажатия на ФИО преподавателя откроется экран с его расписанием. В верхней части экрана увидим заголовок с ФИО преподавателя, а также кнопку, позволяющую сохранить это расписание.

Отметим также, что экран легко листается влево–вправо и вверх–вниз, а дни без пар отображаются корректно в виде ячейки с заголовком.

2.5.4 Создание экрана «Отсутствие расписания»

Экран отсутствия расписания — стандартный экран–заглушка, который открывается при нажатии на кнопку «Расписание» в нижней панели вкладок, если не существует сохраненного расписания. Такой экран является заглушкой и отображает заданное изображение и текст с сообщением, оповещающим пользователя о том, что расписание пустое. На экране используется элемент UIImageView, который используется для отображения картинки с иконкой расписания. Под этим элементов располагается заголовок — UILabel с текстом оповещения. Заглушка реализована в виде отдельного представления UIView, которое способно анимированно пропадать или появляться при изменении состояния видимости расписания.

2.6 Создание экрана «Поиск преподавателя»

Экран поиска преподавателя открывается при выборе в меню пункта «Расписание преподавателя». Его задача — поиск преподавателя, для которого в дальнейшем будет выведено расписание.

В контроллере TeacherSearchViewController.storyboard зададим дизайн для будущего экрана. В верхнюю часть экрана добавим элемент SearchBar, в который будет вводится параметр поиска. Ниже добавим таблицу для отображения результатов поиска.

Для ячеек таблицы зададим дизайн в файле TeacherTableViewCell. Каждая ячейка будет отображать ФИО одного преподавателя, поэтому для ячейки следует добавить только элемент для заголовка — UILabel.

Перейдем к модели экрана. В модели TeacherSearchViewModel реализуем механизм обработки введенного в строке поиска символа. Добавим такую обработку в метод bindUpdateTeachersByWord. После каждого добавления или удаления символа в строке вызывается реактивная связка, которая в первую очередь отменяет предыдущий запрос, если он существует, и ждет выполнения (ожидает данные с сервера). Эти действия необходимы, чтобы ответы с сервера не двоились и отображались корректные результаты поиска.

Затем создается новый запрос к API. Когда ответ с сервера успешно получен, происходит обработка массива преподавателей и в ФИО преподавателей выделяются префиксы, совпавшие с параметром поиска. Для реализации анимированного обновления списка преподавателей, получаемых с сервера, используется анимированный DataSource, позволяющий быстро работать с потоками обновления данных. Если строка с параметром поиска становится пустой, то таблица также очищается. При выборе преподавателя из списка происходит переход на экран с его расписанием.

Запустим приложение и откроем экран поиска преподавателя. Введем в строку поиска фамилию преподавателя. Достаточно быстро в таблице появляются результаты поиска. Действительно, результат соответствует параметру поиска. Также отметим, что введенный в поисковую строку параметр выделяется синим цветом во всех выведенных в таблицу ФИО. Попробуем очистить строку поиска и увидим что список также очистился.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы были изучены следующие инструменты:

- язык программирования Swift;
- среда разработки Xcode;
- основы операционной системы IOS;
- архитектура построения мобильных приложений;
- реактивный подход к архитектуре мобильных приложений;
- библиотеки (Pods) для приложений на языке Swift.

При помощи описанных инструментов была разработана часть мобильного приложения для университета, позволяющая отправлять запросы на сервер, производить поиск преподавателя по заданному параметру, а также открывать расписание преподавателя, студента или сохраненное расписание. Экран меню, а также экран с поиском номера группы студентов описаны в другой работе. Проект был протестирован с помощью симуляторов для различных версий устройства iPhone. В дальнейшем планируется добавление дополнительных функций. Список таких функций находится на этапе обсуждения.