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

СОЗДАНИЕ ЭКРАНОВ ПОИСКА РАСПИСАНИЯ ДЛЯ СТУДЕНТОВ В ПРИЛОЖЕНИИ ДЛЯ УНИВЕРСИТЕТА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Бич Дарьи Дмитриевны

Научный руководитель
доцент, к. ф.-м. н. _____ Ю. Н. Кондратова

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

Саратов 2021

ВВЕДЕНИЕ

В современном мире одним из главных ресурсов является время, его всегда не хватает. Исходя из этой философии и развиваются современные технологии, ставя одной из основных своих задач упрощение и сокращение времени доступа к ресурсам. Например, вполне заметным стало уменьшение времени включения компьютера, открытия приложений, загрузки фотографий.

В связи с этим активно развивается и расширяется рынок мобильных технологий, поскольку включить телефон гораздо быстрее и проще чем ноутбук, да и носить с собой его удобнее. Совершенствуются как сами устройства, увеличивая производительность с каждым новым выпуском, так и программное обеспечение. Остановимся на программном обеспечении.

Рынок мобильных приложений увеличивается с каждым днем. Это происходит из–за того, что компании из разных сфер стремятся к собственному приложению, поскольку это позволяет увеличить поток клиентов, а следовательно и прибыль. В связи с этим увеличивается спрос на компании из сферы IT, создаются и расширяются отделы мобильной разработки, растет спрос на специалистов по IOS и Android разработке.

В ФГБОУ ВО «СГУ имени Н.Г.Чернышевского» возникла необходимость создать мобильное приложение, которое бы дублировало сайт университета. Целью данной выпускной квалификационной работы является разработка части мобильного приложения под операционную систему IOS для дублирования сайта СГУ. В связи с этим необходимо решить следующие задачи:

- рассмотреть средства разработки мобильных приложений под операционную систему IOS;
- создать экраны поиска расписания для студентов в мобильном приложении
- встроить созданные экраны в мобильное приложение под операционную систему IOS
- протестировать разработанную часть мобильного приложения.

Выпускная квалификационная работа состоит из двух глав:

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

1 Средства для разработки мобильных приложений под IOS

В данной работе нужно разработать часть мобильного приложения для университета, которое должно уметь отображать расписание студентов и преподавателей. Для поиска расписания необходимо добавить экраны выбора и ввода данных. К мобильному приложению, которое должно дублировать сайт университета, был предъявлен ряд требований:

- оно должно быть написано под ОС IOS;
- должна быть удобная навигация и интуитивно понятный интерфейс;
- необходимо придумать и соблюдать единый стиль оформления экранов;
- в нижней части экрана должна располагаться нижняя панель вкладок со следующими элементами: центральная круглая кнопка с логотипом СГУ, кнопка «расписание» и кнопка «меню», расположенные по бокам от центральной кнопки;
- экран меню должен содержать пункты:«расписание студентов», «расписание преподавателей»;
- экран выбора курса и факультета должен открываться при нажатии в меню пункта «расписание студентов»;
- экран выбора номера группы, должен открываться после выборе курса и факультета на предыдущем экране;
- после выбора группы должен открываться экран с расписанием студента.

Перед созданием приложения рассмотрим средства разработки мобильных приложений под операционную систему IOS.

1.1 Мобильная операционная система

IOS — мобильная операционная система для смартфонов, планшетов разрабатываемая американской компанией Apple. Первая версия системы была выпущена 9 января 2007 года, а последняя 3 мая 2021 года. Обновления для операционной системы выпускаются каждый год, что позволяет поддерживать максимальную актуальность в мире современных технологий. С 2014 года для создания мобильных приложений под ОС IOS используется язык программирования Swift. Рассмотрим Swift подробнее.

1.2 Язык программирования Swift

Swift — открытый мультипарадигмальный компилируемый язык программирования для написания приложений под устройства с операционной систе-

мой IOS, MacOS, WatchOS, TvOS. Язык был выпущен компанией Apple 2 июня 2014 года. До 2014 года приложения для IOS производились на языке Objective–C.

Objective–C — компилируемый объектно-ориентированный язык программирования, является расширением языка C (любая программа на C является программой на Objective–C).

При выпуске языка Swift была реализована возможность работы кода на Swift параллельно с кодом на C и Objective–C при разработке приложений, что позволило продолжить работу над созданными ранее приложениями без переписывания их на новый язык.

Основные преимущества языка Swift:

- современный — последний выпуск языка 26 апреля 2021 года;
- безопасный — в языке присутствуют optional значения и для безопасного извлечения таких значений создано множество способов;
- выразительный — имеет простой и лаконичный синтаксис;
- быстрый — предназначен для замены языков на основе C и должен быть сопоставим с этими языками по производительности.

1.3 Компилятор XCode

Для разработки приложений на языке Swift используется среда разработки Xcode. Xcode — интегрированная среда разработки программного обеспечения для платформ macOS, iOS, watchOS и TvOS, разработанная корпорацией Apple. Первая версия была выпущена в 2003 году. Поддерживает языки C, C++, Objective-C, Swift, Java, AppleScript, Python и Ruby.

Xcode является единственным компилятором для языка Swift под платформу IOS, поскольку исходные коды, использующиеся для компиляции кода под IOS, можно использовать только через SDK установленный в Xcode. Компания Apple запретила передавать зашифрованные коды компилирования сторонним компаниям и разработчикам, а сама программа Xcode поддерживается только компьютерами произведенными компанией Apple.

Последняя версия Xcode 12 была выпущена 22 июня 2020 года, в ней появилась поддержка новой версии операционной системы macOS Big Sur, выпущенной в 2020 году.

1.4 Архитектура Rx + MVVM

Перейдем к архитектуре приложения. От архитектуры во многом зависит удобство и производительность приложения. Для данного проекта была выбрана архитектура Rx + MVVM.

Архитектура Rx позволяет применять в проекте реактивное программирование. Принципы реактивного программирования схожи с паттерном «Наблюдатель», когда есть класс «издатель», который публикует изменения своего состояния, и классы «подписчики», которые подписываются на изменения состояния издателя. С помощью реактивного подхода можно оптимизировать проектирование архитектуры приложения, чтобы быстрее и качественнее моделировать экраны, а также оптимизировать происходящие внутри процессы.

Архитектура Rx + MVVM подразумевает наличие реактивной модели и реактивного контроллера. И, если возвращаться к паттерну «Наблюдатель», то контроллер выступает в роли издателя, поскольку отображает действия пользователя и публикует их, а модель выступает в роли подписчика, получая информацию о действиях пользователя от контроллера.

Роль контроллера выполняет — ViewController, он посылает все события (например, нажатие) в модель — ViewModel, которая их обрабатывает, получает из них данные, кэширует, обрабатывает с помощью бизнес–логики, настраивает сервисы для получения данных из сети. После того, как ViewModel приводит данные к требуемому виду, она возвращает их во ViewController, который эти данные принимает и отображает на экране.

ViewModel и ViewController взаимодействуют между собой с помощью структур input и output. Структура input содержит входные данные в виде событий, получаемых моделью от контроллера, а структура output содержит выходные данные в виде секций, уведомлений и прочего, полученных после обработки input данных в модели и передаваемые из модели в контроллер.

Такая архитектура предоставляет возможности гибкой настройки под различные виды задач. Одним из возможных способов настройки является дополнительная структура — DataSource. Она содержит в себе описание того, как мы будем конфигурировать данные, получаемые контроллером из модели. Таким образом DataSource описывает конфигурацию данных, которые мы будем отображать на экране контроллера, занимается обработкой, настройкой и поставкой этих данных из ViewModel во ViewController. В данном проекте

используются анимированные DataSources, которые помимо ранее описанных настроек умеют делать анимацию для таблиц.

1.5 Библиотеки (Pods)

Чтобы реализовать задуманный функционал приложения необходимо использовать сторонние библиотеки с открытым исходным кодом. Они нужны для использования архитектуры Rx, навигации между экранами, скачивания фотографий и прочего. Такие библиотеки в IOS приложении называются Pods. Для скачивания и использования таких библиотек в данном проекте используется менеджер Cocoa Pods — менеджер зависимостей для проектов на языке Swift и Objective–C, написанный на языке Ruby.

Все необходимые библиотеки для импорта указываются в спецификации — файле с названием «Podfile», находящемся в папке проекта.

2 Разработка мобильного приложения

2.1 Настройка библиотек

Чтобы использовать все преимущества языка Swift, а также правильно настроить архитектуру Rx, необходимо установить библиотеки, так называемые Pods. Рассмотрим подробнее библиотеки, использующиеся в данном проекте:

- R.swift — отвечает за автозаполнение строго типизированных ресурсов, таких как изображения, шрифты и переходы;
- Reusable — позволяет использовать ячейки таблиц и коллекций с сохранением типа;
- Then — предоставляет возможности для синтаксического сахара при инициализации объектов;
- RxSwift — предоставляет возможности реактивного программирования;
- RxCocoa — предоставляет дополнительные возможности для реактивного программирования;
- RxDataSources — предоставляет возможность использовать анимированные DataSources, умеющие создавать анимацию для таблиц;
- RxFlow — помогает настроить навигацию между экранами;
- SwiftSoup — позволяет скачивать HTML страницы, парсить их и мапить в модели проекта;
- Kingfisher — позволяет скачивать фотографии по ссылке;
- Moya — позволяет работать с интернет сессиями.

2.2 Настройка иконок

Для иконки данного приложения было решено использовать символику университета. Для корректного отображения иконки приложения на различных устройствах, в программу добавили изображение нарезанное с различными размерами. Подготовим изображения и после нарезки получим и добавим в проект 18 изображений.

2.3 Создание экрана «Меню»

Начнем с экрана меню, поскольку остальные экраны будут открываться через него. Экран меню должен открываться при нажатии на кнопку «Меню», расположенную на нижней панели вкладок. На экране должна располагаться

таблица, состоящая из пунктов: «Расписание студентов», «Расписание преподавателей». Каждая ячейка таблицы должна содержать картинку и текст.

Для начала необходимо создать ячейку таблицы, внутри которой будет отображаться картинка и заголовок пункта меню. Настроим визуальную часть ячейки, для этого добавим все необходимые элементы: фон, заголовок, картинку. Для каждого элемента верстки создадим переменные класса, которые будут иметь аннотацию @IBOutlet. Также добавим функцию конфигурации, которая в качестве входных параметров будет получать заголовок в виде текста и конкретное изображение для ячейки. Внутри этой функции элементу заголовка в качестве поля text будет передаваться полученный текст, а у элемента изображения в качестве поля image будет сохраняться переданное изображение.

Создадим MenuSectionType, который будет хранить тип данных для ячейки, чтобы связать созданные ранее ячейки с таблицей на нашем экране. Тип будет хранить заголовок и изображение для ячейки. Также, внутри раздела DataSource расположим файл с классом
MenuDataSource, который будет посредником между контроллером и моделью.

Раздел ViewModel будет содержать обработку данных с контроллера. Внутри создадим файл с классом MenuViewModel, для которого переопределим метод протокола — bindOutput, производящий обработку входного параметра, получаемого от контроллера, и преобразовывающий его в выходной параметр, отправляемый обратно контроллеру. Чтобы такой метод обработки можно было реализовать, необходимо создать вспомогательные структуры input, которая будет содержать входящий параметр в виде нажатой пользователем ячейки, и output, которая будет содержать выходной параметр, возвращающий структуру DataSource и новый список секций. Внутри модели определим заголовки для ячеек: «Расписание преподавателей», «Расписание студентов». Для каждой ячейки добавим соответствующее изображение. Также определим метод createSection, который будет заполнять ячейки таблицы необходимыми значениями, и метод sectionSelected для инициализации выбранной ячейки и перехода на следующий экран.

Контроллер будет содержать отображение экрана для пользователя и отслеживать действия пользователя. Добавим необходимые элементы верстки на экран: фон, таблица, отступы. Добавим элемент таблицы как переменную класса и свяжем ее с соответствующим элементом на экране верстки. Так-

же внутри контроллера создадим переменные для входной и output структур, модели, а в selectedSection будем сохранять нажатую ячейку.

Переопределим родительский метод viewDidLoad, который будет загружать представление экрана. Внутри него будем вызывать дополнительные функции. Метод setupTableView будет настраивать отображение таблицы, метод createOutput будет получать входящий параметр и преобразовывать его в выходной параметр с помощью функции модели — bindOutput, метод bindDataSource будет связывать структуру DataSource и выходной параметр, а метод bindModelSelecting будет связывать выбранную ячейку таблицы и переменную selectedSection.

Запустим приложение через симулятор, и нажмем кнопку меню в нижней панели вкладок. Кнопка меню изменит цвет с черного на синий, затем откроется экран меню с двумя пунктами «Расписание студентов» и «Расписание преподавателей».

2.4 Создание экрана «Выбор факультета и курса»

Перейдем к экрану выбора факультета и курса, который открывается при выборе на экране меню пункта «Расписание студентов». Данный экран должен состоять из коллекции, в которой будут отображаться номера курсов от 1 до 6, таблицы, которая будет отображать список факультетов, кнопки для перехода на следующий экран и двух заголовков: «Курсы» и «Факультеты». Кнопка перехода на следующий экран должна появляться в тот момент, когда будут выбраны курс и факультет, и отображать выбранные значения этих полей.

Начнем с создания ячеек. Поскольку у нас будет два списка на экране, то необходимо создать ячейку для каждого из них. Ячейка для курса будут иметь тип CollectionViewCell, поскольку список курсов на экране будет являться коллекцией. Добавим на ячейку основной элемент фона, затем добавим элемент для заголовка и зададим для него центрирование по вертикали и горизонтали. Создадим переменные для фона и заголовка и привяжем их к элементам верстки. Добавим функцию конфигурации ячейки, которая будет изменять ее внешний вид в зависимости от входного параметра isSelecting.

Ячейка для факультета будет иметь тип TableViewCell, поскольку будет отображать ячейки таблицы. Добавим основной элемент фона View, затем добавим элемент для заголовка ячейки. Создадим переменные для фона и

заголовка привяжем к элементам верстки, а затем настроим параметры отображения этих элементов с помощью кода. Добавим в класс функцию конфигурации, в которой если ячейка выбрана, то будем выделять ее, а иначе убирать выделение.

DataSource экрана разделен на Course, содержащий файлы настроек для коллекции курсов, и Faculty, содержащий файлы настроек для таблицы с факультетами. Такое разделение необходимо, поскольку их конфигурации отличаются. Начнем с раздела Course, для связи ячеек и данных создадим перечисление CourseSectionType, которое будет содержать номер курса строкового типа и статус логического типа. Перейдем к разделу Faculty, для связи ячеек и данных создадим перечисление FacultySectionType, которое будет содержать факультет типа Department и статус логического типа.

Внутри раздела ViewModel будет хранится логика обработки данных, пришедших от контроллера. Для входного параметра, хранящего передаваемые от контроллера данные, создадим структуру CourseFacultyViewModelInput, в которой определим переменную для задания функций выезжающей кнопки, переменную для хранения выбранной ячейки таблицы с факультетами, и переменную для хранения выбранной ячейки коллекции с курсами. Зададим класс для выходного параметра. Для курсов и факультетов зададим DataSource и список секций, а также передадим текст, который будет отображаться на кнопке. Для модели добавим переменную departmentService, которая будет отвечать за отправку запросов на сервер, поскольку перед тем как вывести список факультетов его необходимо получить. Выбранные курс и факультет будем записывать в переменные selectedCourse и selectedDepartment, соответственно. Затем создаем инициализатор, в который будет передаваться сервис и будут создаваться секции для курсов и факультетов. Создадим метод createCourseSection, в котором будем заполнять ячейки коллекции с курсами. Зададим список курсов вручную строками от 1 до 6. Полученный массив запишем в объекты класса CourseSectionType и отфильтруем пустые значения. В результате вернем список секций для коллекции курсов. Создадим также метод createFacultySection, в котором будем получать с помощью сервиса список факультетов и записывать полученные данные в ячейки таблицы.

Для модели переопределим метод протокола — bindOutput, в котором из входного параметра получим выбранный факультет, изменим значение пере-

менной класса selectingDepartment на полученный факультет и пересоздадим секции для факультетов. Затем проверим не нужно ли выехать из под нижней панели вкладок кнопке перехода на следующий экран. Далее из входного параметра получим выбранный курс и изменим значение переменной selectingCourse на полученное значение, пересоздадим ячейки коллекции и снова проверим не нужно ли показать кнопку. На следующем шаге получим из входного параметра действие пользователя — нажатие кнопки. Если выбраны оба параметра (курс и факультет), то возвращаем переход на следующий экран с выбором номера группы, а иначе возвращаем выходной параметр.

Перейдем к верстке основного экрана. Добавим в верхнюю часть экрана элемент–контейнер View, зададим для него отступы. На View добавим два заголовка, которые будем использовать для отображения строк «Курсы» и «Факультеты», а между ними поместим элемент для отображения коллекции курсов — CollectionView. В нижнюю часть экрана добавим элемент для отображения таблицы с факультетами — TableView, а также кнопку — Button.

Зададим переменные для элементов экрана, привяжем их к элементам верстки и настроим с помощью кода. Добавим переменные для входного и выходного параметров, позволяющие производить обработку действий пользователя, а также disposeBag для сохранения переходов между экранами. Для выбранного факультета и курса создадим переменные selectingDepartment и selectingCourse, а для нажатия кнопки добавим переменную continueTapped.

Переопределим функцию viewDidLoad. Создадим функции, которые будем вызывать внутри данного метода. Для настройки отображения таблицы факультетов будем вызывать функци setupTableView, а для настройки коллекции курсов будем вызывать setupCollectionview. Внутри функции createOutput преобразуем входной параметр в выходной параметр с помощью метода модели bindOutput. Затем проверим, выбраны ли у нас оба параметра (курс и факультет), если да, то вызывем функцию displayContinueButtonWithAnimation, которая помогает кнопке выезжать и становится активной.

Запустим наше приложения для устройства iPhone 12 Pro Max. Список факультетов и курсов отображается корректно, при пролистывании ячейки заходят под верхнюю панель. Выберем 3 курс и факультет КНиИТ, ячейки

изменят цвет. Также заметим, что после выбора курса и факультета появилась красная кнопка с номеров выбранного курса и аббревиатурой выбранного факультета.

2.5 Создание экрана «Выбор номера группы»

Перейдем к экрану выбора номера группы, который будет открываться после экрана выбора курса и факультета при нажатии на выплывающую кнопку. Экран должен состоять из заголовка с информацией о выбранных значениях курса и факультета, а также коллекции номеров групп.

Как и ранее, начнем с верстки ячейки для номера группы. На основу добавим элемент–контейнер — View, для которого выставим отступы, также добавим элемент с заголовком — Label, для которого зададим центрирование по горизонтали и вертикали, чтобы заголовок отображался по центру.

Создадим переменные для фона и заголовка. В функции конфигурации будем изменять текст заголовка, используя данные с предыдущего экрана. Привяжем созданные переменные к элементам верстки.

Перейдем к разделу DataSource, который будет хранить тип ячейки — параметр number строкового типа для хранения номера группы. Для общения контроллера и модели создадим вспомогательные структуры:
StudentGroupViewModelInput, которая будет хранить выбранную группу, и StudentGroupViewModelOutput, которая будет возвращать DataSource, а также номер группы и факультет, которые необходимы на следующем экране для получения расписания.

Внутри класса модели объявим переменную для сервиса, которая будет отвечать за получение списка групп. Также объявим переменную department, в которой будем передавать факультет на следующий экран. В функции инициализации присвоим значения этим переменным, а также определим текст заголовка. Далее вызовем функцию bindStudentGroup которая отправит запрос, получит список групп и создаст из него коллекцию.

Перейдем к верстке экрана. Добавим в верхнюю часть экрана элемент–контейнер, на котором расположим элемент Label для отображения заголовка с аббревиатурой факультета и номером курса, зададим для них необходимые отступы. В нижнюю часть экрана добавим элемент для коллекции номеров групп — CollectionView.

Создадим переменные для элементов экрана верстки и произведем до-

полнительную настройку с помощью кода. Добавим в контроллер переменные для входных и выходных параметров, а также для модели, чтобы экран мог обрабатывать действия пользователя. Для сохранения номера выбранной группы создадим переменную selectedGroup.

Рассмотрим функцию загрузки экрана — viewDidLoad. Для начала изменим заголовок и цвет фона верхней панели View, а затем вызовем вспомогательные функции. Внутри вспомогательной функции setupUI вызовем функцию настройки коллекции — setupCollectionView, а в функции createOutput входные параметры преобразуем в выходные параметры с помощью реализованного в модели метода протокола bindOutput.

Запустим приложение для устройства iPhone 12 Pro Max. На экране выбора курса и факультета выберем 4 курс и факультет КНиИТ и нажмем на кнопку перехода на экран выбора группы, увидим созданный ранее экран. На экране отображается заголовок с правильными данными, а коллекция отображает все группы для заданного курса и факультета.

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, которая позволяет выбирать и вводить параметры для поиска расписания студентов. Экраны с поиском преподавателя и выводом расписания 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.