**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Выпускная квалификационная работа**

на тему «Разработка iOS - приложения BSTU App»

по направлению подготовки 09.03.04 «Программная инженерия»

|  |  |
| --- | --- |
| Руководитель работы:  Заикин Александр Олегович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата | Выполнил:  Студент группы ПВ-41  Таран Алексей Николаевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата |

**Белгород 2021**

**Содержание**

[Введение 3](#_Toc69468704)

[Глава 1. Возможности информационных технологии в реорганизации образования 6](#_Toc69468705)

[1.1. Тенденции в современном образовании 6](#_Toc69468706)

[1.2. Роль мобильных платформ в современном мире 8](#_Toc69468707)

[**1.2.1. Краткая история развития мобильных ОС 8**](#_Toc69468708)

[**1.2.2. Факты о платформе iOS 10**](#_Toc69468709)

[**1.2.3. Распространение версий iOS среди пользователей 11**](#_Toc69468710)

[**1.2.4. Магазин приложений App Store и поддержка Apple в сфере образования 12**](#_Toc69468711)

[1.3. Обзор существующих решений для ВУЗов 14](#_Toc69468712)

[1.4. Требования к системе. 16](#_Toc69468713)

[Глава 2. Архитектурные особенности приложения и принцип работы 18](#_Toc69468714)

[2.1. Структура работы приложения 18](#_Toc69468715)

[2.2. Принципы функционирования модулей 23](#_Toc69468716)

[2.3. Архитектура приложения 30](#_Toc69468717)

[2.4. Проектирование графического интерфейса. Apple HIG. 32](#_Toc69468718)

[Глава 3. Описание программной реализации iOS-приложения 36](#_Toc69468719)

[3.1. Используемые технологии 36](#_Toc69468720)

[3.2. Разработка модуля «Расписание» 36](#_Toc69468721)

[3.3. Программирование раздела «Навигация» и интеграция с базой данных 40](#_Toc69468722)

[3.4. Реализация расписания занятий для преподавателей 48](#_Toc69468723)

[3.5. Создание раздела «Пункты питания» 49](#_Toc69468724)

[3.6. Реализация раздела «Личный кабинет» 50](#_Toc69468725)

[3.7. Тестирование 54](#_Toc69468726)

[Заключение 56](#_Toc69468727)

[Список использованных источников 57](#_Toc69468728)

# Введение

По данным отечественных аналитических компаний, в России количество людей, пользующихся смартфонами, составляет около 95 млн человек. Важным аспектом в росте популярности умных гаджетов является постоянное совершенствование качества написанного ПО (удобства и функциональности), а также развитие интернет-сетей, обеспечивающих стабильный и быстрый интернет [1].

На сегодняшний день, телефон можно назвать полноценным карманным компьютером. В 2020 году, Apple представила собственный процессор Apple M1 на архитектуре Apple Silicon с основой ARM64 для компьютеров Mac, что, в будущем, должно оказать влияние на разработку ПО. Все производители мобильных процессоров на сегодняшний день: Qualcomm Snapdragon, Samsung Exynos, MediaTek, Apple A series используют ARM-архитектуру. Благодаря выходу компьютеров с M1 на ARM архитектуре, появляется возможность одновременной разработки приложений, предназначенных как на iOS, так и на macOS, что позволяет пользоваться одними и теми же приложениями на компьютере, планшете, телефоне. Этот факт, может мотивировать других производителей компьютерного оборудования вести собственные разработки, направленные на постепенный отказ от intel x86-64 в пользу ARM. Поэтому, индустрия мобильный приложений c годами продолжает свое развитие.

На сегодняшний день, в любом смартфоне, есть встроенный браузер, который позволяет осуществлять доступ к необходимым пользователю сервисам. К тому же, часто разработчики сайтов делают мобильные версии своих веб-приложений, которые удобно отображаются на телефоне. Гораздо больше возможностей, для эффективной коммуникации с пользователем предоставляет мобильное приложение. Оно позволяет:

• Объединить самые необходимые сервисы,

• Реализовать функционал, недоступный для мобильных веб - сайтов. Например, отправка уведомлений, добавление виджетов на телефоне.

• Приложение гораздо проще продвигать, благодаря, таким мощным платформам как Google Play, App Store, Marketplace. Приложение может попасть в список Топ и будет предложено потенциальным пользователям, которым оно может быть интересно.

• Некоторый функционал мобильного приложения может быть доступен в офлайн режиме и не требует доступа к интернету.

Множество процессов, присущих каждому университету, может быть автоматизировано. Телефон, посредством мобильного приложения, может способствовать удобному доступу к необходимым сервисам, а также реализовать удобный интерфейс и функционал, характерный конкретно для мобильного устройства. Сократится время доступа к нужной информации, в сравнении с веб-сайтом. Поэтому, в разработке приложения заинтересованы, как сотрудники и преподаватели университета, так и студенты.

Сформулируем основные потребности, необходимые для быстрого доступа подавляющего числа посетителей университета ежедневно:

• Расписание для групп. Каждый студент должен получить доступ к расписанию своей группы с указанием времени начала и конца занятий, списком преподавателей, аудиториями, в которых проходят занятия, а также типом занятия. Должно быть включено также и экзаменационное расписание.

• Навигация. Человек, оказывающийся в новом незнакомом помещении, часто не имеет информации, как пройти к тому или иному кабинету. Особенно, эта проблема актуальна для новых людей в институте, которые тратят свои перерывы или опаздывают на занятия из-за проблем поиска. Также вопрос навигации по университету актуален и во время дней открытых дверей, а также экскурсий. Ее качественная реализация позволяет гостям наглядно ознакомиться с расположением аудиторий в телефоне и пройти к необходимой.

• Расписание для преподавателей. В приложении удобно организовать интерфейс для пофамильного поиска для преподавателя и отобразить необходимое расписание с учетом типов недели, а также экзаменов и расписания для заочных групп. Нет необходимости для просмотра расписания пользоваться веб-версией, использовать личный кабинет, делать пометки в записных книжках, устанавливать напоминания в календарь.

• Личный кабинет. Сотрудник может получить информацию о себе в контексте института. В случае студента, это могут быть: текущие оценки, результаты сессий, расписание для свой группы. Тогда приложение может помочь избежать длинной цепочки стандартных действий на веб-версии сайта.

• Пункты питания. Множество людей на территории вуза пользуются пунктами питания. Мобильное приложение может позволить в режиме live оценить список меню той или иной точки питания, и определить подходящее для себя.

Многие крупнейшие российские и иностранные ВУЗы на сегодняшний день имеют свои приложения, автоматизирующие некоторые из перечисленных выше процессов для использования студентами и преподавателями в повседневной жизни. В БГТУ им. Шухова обучается около 25 тыс. студентов. С каждым годом в вузе проходят обучение все большее число иностранцев со всего мира. Немаловажным фактором, соответствующим современным критерием для вуза, является также и грамотное цифровое обеспечение.

На сегодняшний день, для улучшения качества обучения, актуально создание мобильного приложения для студентов, преподавателей и гостей вуза с удобным интерфейсом и реализации необходимых функций.

**Постановка задачи**

Целью дипломной работы является реализация мобильного приложения на базе iOS, позволяющего осуществить удобный доступ к сервисам БГТУ им. Шухова. Необходимо решить следующие задачи:

• Разработка и реализация графического интерфейса для расписаний групп институтов студентов с использованием существующего API.

• Создание навигации для корпусов университета. Разработка интерфейса, стандартизация карт, логика построения кратчайшего пути, рисование пути, выбор пунктов на карте, поиск аудитории. Разработка базы данных, а также ее заполнение информацией о корпусах, аудиториях.

• Реализация пофамильного поиска преподавателей c расписанием по числителю и знаменателю, а также с информацией об экзаменационном расписании и расписании заочных групп. Заполнение информацией расписания преподавателя данными из API.

• Создание интерфейса для отображения меню пунктов питания, подключение API.

• Реализация личного кабинета в мобильном приложении. Авторизация в приложении. Получение через API информации о сотруднике. Для студента – создание окон аттестации и успеваемости, где есть информация о текущих оценках и о оценках сессии, полученных за время обучения в ВУЗе.

# Глава 1. Возможности информационных технологии в реорганизации образования

## 1.1. Тенденции в современном образовании

Внедрение новых технологий в какой-либо области всегда следует за открытиями в других областях, а также же за потребностями общества. Появление современных средств коммуникации, в том числе, компьютера, планшета, смартфона оказало сильное влияние на развитие многих областей, включая образование [2].

При этом, информационные технологии, как символ прогресса, должны стать не дополнительным используемым средством, а полноценной частью образовательного процесса, которая повышает его эффективность.

В качестве возможностей ИТ для образования можно выделить: совершенствование и организации обеспечения образовательного процесса (внедрение электронных материалов для учащихся, виртуальные школы, лаборатории); совершенствование самого программного обеспечении в соответствии с современными тенденциями получения информации.

Информационные технологии в образовании достаточно разнообразны. Каждые из них являются частью целого комплекса для полноценного и наиболее всеобъемлющего обучения, а также призвано сделать процесс взаимодействия удобным для всех сторон.

Примерами могут являться: повышение доступности образования в связи с введением различных форм получения знаний, внедрение индивидуальных траекторий для обучения, совершенствование методических комплексов, автоматизация существующих решений и внедрение новых из других сфер [3].

Например, автоматизация, изначально, начиналась на заводе и только затем перешла в офис, а, затем, и в другие сферы, в частности, в образование. Специалистов в области автоматизации в учебном процессе, как и в других сферах, привлекает возможность повышения производительности своего труда, путем автоматизации рутинной работы.

Автоматизация процессов организации должна в себя включать создание и поддержку коммуникационных процессов не только внутри организации, например, университета, но и со внешней средой, например, через Интернет.

Автоматизированные технологии помогают повышать производительность работников, а также экономят время на новые типы работ.

В настоящее время, в России, как и в других странах мира, ИТ активно внедряются в учебный процесс. Таким образом, обучение приходит в соответствие с современным уровнем науки, повышается эффективность учебной работы, учащиеся проходят подготовку к деятельности в информационном обществе после обучения. В связи с этим, информационные технологии становятся неотъемлемым компонентом в обучении, повышая эффективность процесса обучения и развивают учащегося.

Cуществует несколько направлений применения ИТ в учебной сфере:

•  Разработка вспомогательных интерактивных средств для педагогов

•  Разработка Web-порталов с необходимой информацией

•  Разработка электронных пособий и дидактических материалов

•  Внедрение ботов для оперативного решения частых проблем

• Организация и моделирование физических и химических экспериментов, обработка их результатов

• Работа с большими массивами данных

•  Организация интеллектуального досуга для учащихся

• Организация удобного доступа к сервисам учебного заведения. Повышение эффективности взаимодействия человека с учебным заведением.

Видеоконференции, новые учебные пособия и материалы позволяют сделать современное образование интересным.

Важнейшими потенциальными выгодами от применения информационных технологий является удобство и продуктивность, а также экономия времени. Именно поэтому, быстрый доступ к ресурсам, обеспечение современными учебными платформами и сервисами, сегодня наиболее важно для обучаемых.

Согласно данным компании AppAnnie, занимающейся аналитикой в области рынка мобильных приложений, мобильные приложения генерируют в 7 раз больше интернет-трафика, нежели мобильные версии веб-сайтов. Это говорит о том, что люди предпочитают удобные приложения сайтам. Мобильные приложения способствуют оптимальному получению информации – в минимальные сроки и с минимальными усилиями и, в контексте учебного заведения, должны быть направлены на повышение качества обучения.

## 1.2. Роль мобильных платформ в современном мире

### 1.2.1. Краткая история развития мобильных ОС

Мобильные устройства прошли путь от инструмента для совершения звонков и отправки сообщений до полноценного помощника, обладающего интуитивно понятным управлением и являющегося незаменимой частью повседневной жизни, в том числе, и в сфере образования.

Первые мобильные операционные системы, начали появляться в 1988 году, когда компания Psion разработала для карманных персональных компьютеров 16- разрядную операционную систему EPOC-16. В ее состав входили: калькулятор, записная книжка, текстовый редактор, файловый менеджер и тд. Позднее, та же самая компания начнет разработку 32-разрядной ОС и назовет ее Symbian, долгое время являющейся самой популярной в мире среди пользователей [4].

ОС, разработанные некоторыми компаниями, были плохо востребованными, однако, содержали технологии, впоследствии внедренными в уже в новые ОС. Например, в 1993 году Apple представила Newton OS, в которой была реализована система распознавания слитного рукописного текста. Символы записывались на экран специальным устройством – стилусом, а затем, автоматически переводились в текст после ввода целого слова. В том числе, поддерживалось написание и на русском языке.

Несколько лет спустя, в 1996 году на свет появилась OC Palm. Особенностью устройства, помимо поддержки рукописного ввода, являлось приложение «Security», в котором была возможность защищать данные, хранящиеся на телефоне от несанкционированного доступа, например, паролем, только после ввода которого были доступны заметки, календари, список дел. Со временем, теряла популярность, а в 2012 году стала свободным ПО. На сегодняшний день, одна из разновидностей Palm, называемая Web OS используется в умных телевизорах.

Также, в 1996 году Microsoft вышла на рынок мобильных ОС, представив Microsoft Windows CE. Она являлась упрощенной версией настольной версии Windows 95, оптимизированной для процессоров с архитектурой MIPS, x86, ARM и Power PC. Позднее, была трансформирована в Microsoft Pocket PC и Windows Mobile (а затем и в Windows Phone). Тогда, карманные гаджеты стали оснащаться оптимизированными под размер экрана версиями компьютерного ПО. Например, в состав ОС от Microsoft входили такие популярные приложения как, Word, Excel, Internet Explorer.

В 1999 году в Канаде, компания Blackberry представила операционную систему для коммуникаторов, которая поддерживала передачу зашифрованных писем электронной почты и мгновенные сообщения, используя собственный сервер, а также технологию Peer-to-Peer. В США, Канаде и Западной Европе Blackberry получил хорошую репутацию в бизнесе, как надежный помощник. Считалось, что сообщения были надежно защищены и невозможны для перехвата даже спецслужбами.

В 2007 году началась новая эра в развитии мобильного ПО, связанного с выходом iPhone. Мобильная платформа iPhone OS, впоследствии, переименованная в iOS, обязана свой популярностью новому интерфейсу, в котором отказались от использования в телефонах стилуса, а также выпустили технологию «мультитач», при которой устройство управлялось при помощи пальцев руки. Впоследствии, iOS был объединен в одну экосистему с macOS, с watchOS, позволяя эффективно взаимодействовать с другой компьютерной техникой Apple.

Год спустя, в 2008 году, усилиями Google, на рынок вышла ОС Android. На сегодняшний день, является самой популярной в мире операционной системой, установленной на телефонах, планшетах, роботах, смарт-часах и браслетах. Ее популярность обусловлена широкими возможностями индивидуализации под нужды пользователя, полноценная многозадачность, разнообразным ценовым диапазоном устройств. Открытый исходный код дает возможность сторонним разработчикам периодически вносить поправки, исправлять баги, улучшать интерфейс.

Главную роль в развитии мобильных платформ сыграло наличие огромного числа приложений под разные виды задач в специализированных магазинах AppStore для iOS и Google Play для Android. Ранее, процесс установки приложений, в частности для Symbian представлял собой сложную процедуру – скачивания приложения на компьютер, подключения кабеля к компьютеру, копирования приложения на смартфон, установка на телефоне, путем запуска инсталлятора. В настоящий момент, эта процедура сильно упростилась, так как установка приложения происходит путем нажатия одной кнопки установки из магазина.

По данным аналитических компаний, в настоящий момент, число загрузок мобильных приложений в мире на 2 самых популярных платформах iOS, Android превышает 100 млрд. в год. Также растет доходность компаний от продажи своих приложений через магазины. Для iOS она составляет более $60 млрд, для Android – около $30. С каждым годом, эти цифры растут в связи с развитием мобильных технологий, например c появлением оплат через банковские карты, эмуляцией электронных пропусков, различными сервисами, образовательных порталов и тд.

### 1.2.2. Факты о платформе iOS

iOS базируется на модификации ОС Darwin, являющийся набором Unix-подобных компонентов. Центральной частью Darwin является гибридное ядро XNU, состоящее из микроядра Mach и компонентах ядра FreeBSD (планировщик процессов, файловая система, сетевой стек, POSIX Api) [5]. Также, в Darwin используется API для драйверов, названный I/O Kit, позволяющий писать драйвера в объектно-ориентированном стиле на С++. Уровнем выше ядра, в Darwin располагается слой UNIX/BSD, который включает в себя набор библиотек языка Си, shell и различные демоны (например, ssl).

Далее, часть Darwin закачивается, начинается слой абстракций, который образуют то, что считается iOS. Выделяется 4 компонента:

*Cocoa Touch* – самый высокоуровневый компонент. Это фреймворк для создания приложений под iPhone, iPad и iPod touch. Основан на классах фраймворка Cocoa, используемого в Mac OS X. Предоставляет библиотеку для отображения GUI на устройстве – UIKit, библиотеку для взаимодействия с картами MapKit, библиотеку GameKit для взаимодействия с Game Center и другие. Также в Cocoa Touch есть специальные возможности для распознавания multitouch-жестов, управления уведомлениями (Notification Center), реализация мультизадачности.

*Media Layer –* управление графикой, аудио и видео. Например, Assets Library – позволяет получить доступ к медиатеке фотографий на устройстве. Core Audio – предоставляет платформу программного обеспечения, разработанную для обработки аудио в приложениях. AV Foundation – используется для управления записью аудио и контроль над процессом воспроизведения звука. Поддержка обработки векторной графики при помощи OpenGL и Metal, а также СoreGraphics для создания двухмерных и трехмерных изображений.

*Core Services –* управление основными системными сервисами. Главные компоненты – управление системными очередями (Grand Central Dispatch), управление встроенными покупках в приложении (In-AppPurchase), управление хранилищем iCloud. На этом уровне также реализован механизм управления памяти для приложений – автоматический подсчет ссылок – Automatic Reference Counting. Core Foundation – платформа, предоставляющая удобные структуры данных для разработки – массивы, словари, даты и тд.

*Core OS –* низкоуровневая реализация компонентов 4 вышестоящих слоев. Реализация решения вопросов безопасности, внешнего доступа, работой с внешними устройствами. Управление работой акселерометром. Реализует многие интерфейсы, к которым приложение не имеет доступ.

Каждое из приложений изолируется песочницей (Sandbox), которая не позволяет приложениям осуществить прямой доступ к железу и системным функциям за пределами собственного каталога. Эти ограничения существуют только для приложений, написанными сторонними пользователями. Преимущества использования песочниц – защита от негативного воздействия на систему. При этом, приложения должны быть подписаны цифровым ключом App Store. Многие из этих ограничений дают возможность iOS быть одной из самых защищенных операционных систем.

iOS любят за высокое качество интерфейса и оптимизации предоставляемых приложений, длительную поддержку старых устройств, частый выход обновлений, удобный переход со предыдущей моделей телефонов, возможности простого восстановления.

### 1.2.3. Распространение версий iOS среди пользователей

По данным Apple Insider, если принимать во внимание все устройства с iOS, то на конец 2020 – начало 2021 года самой популярной версией является iOS 14 со степенью распространения около 72%, 18% пользователей остаются с iOS 13, 10% пользователей используют более ранние версии системы.

Для разработки приложений на новой iOS 14, интересным моментом является новый тип виджетов, которые могут располагаться на главном экране и могут содержать потенциально важную информацию для пользователя. Появились AppClips, позволяющие использовать небольшие части приложений, минуя AppStore. В частности, выходят минорные апдейты, позволяющие использовать новые технологии в области камеры и фотографий, а также звука. iOS 14 доступна для устройств, использующих чип Apple A9 и новее.

На 2020 год в России, в том числе, среди студентов, по-прежнему популярны устройства, которым более 5 лет, например, iPhone 5S/6. Особенно популярным является iPhone SE 1-го поколения, который входит в число самых распространенных среди всех, даже новых устройств.

Представленная в 2018 году iOS 12, начиная с конца 2019 года поддерживает 12 версию, выпуская апдейты, повышающих производительность и безопасность для старых устройств (начиная с процессора A7). Для приложений информационного характера нет необходимости в использовании SDK для новых iOS, так написанное приложение на старой версии iOS будет работать на новой версии и отсутствие нововведений, например связанных с отображением графики или новых анимаций, не является критичным. Зато, появляется возможность охватить большее число потенциальных пользователей.

Тем более, Apple представляет возможность для поддержки версионирования кода в приложениях, позволяя, со временем, добавлять новый функционал для новых версий iOS, не нарушая работу приложений на старых версиях.

Таким образом, на 2021 год для информационных приложений компромиссным вариантом с точки зрения охвата аудитории и функционала – является минимальная поддержка начиная с iOS 12.0, то есть, для устройств iPhone 5s и новее, iPod Touch 6-го поколения и новее.

### 1.2.4. Магазин приложений App Store и поддержка Apple в сфере образования

В 2008 году Apple, с выходом iPhone OS 2.0 на свет появился магазин приложений App Store для предложений, разработанных с использованием Apple Software Development Kit. Эти приложения могут быть загружены на iPhone, iPad, смарт-часы Apple Watch. Первоначально, магазин содержал 500 приложений. Пика достиг в 2017 году, когда число приложений составило примерно 2.2 миллиона. По состоянию на конец 2020 года, в результате удаления старых, не соответствующих требованиям приложений, в магазине представлено около 1.8 млн. приложений.

За время создания App Store, разработчикам, со стороны Apple было выплачено более $155 млрд. На сегодняшний день, пользователями загружено более 150 млрд. приложений.

Для публикации приложений в App Store, разработчики должны иметь специальный аккаунт разработчика, который стоит $99 в месяц. В США, Германии, Франции и некоторых других странах для некоммерческих, образовательных организаций и правительств существует льгота, позволяющая не платить данный взнос. России в этом списке нет.

Существует несколько моделей распространения приложений в App Store. Они комбинируют варианты платной/бесплатной загрузки и платные/бесплатные дополнительные опции внутри приложения.

Apple Store имеет раздел приложений, посвящённый образованию. Из поддерживаемых Apple проектов можно выделить: Pages – для создания документов – бесплатный аналог Microsoft Word, Numbers – приложение для создания таблиц, Keynote – для создания презентаций, Swift Playgrounds – интерактивная книга для обучения программированию на языке программирования Swift.

Крупные образовательные платформы, такие как Coursera, Stepik, Sololearn также имеют свои клиенты на iOS с удобным интерфейсом для дистанционного обучения на телефоне.

Многие популярные на других платформах приложения, например, Microsoft Office, Todoist для записи дел и задач, Dropbox – облачное хранилище файлов, PDF Expert - для чтения книг и пособий, uMake 3D CAD – для создания дизайна, Matlab – для математических вычислений также портированы на iOS и предлагают скидки на лицензию для студентов.

Существуют отдельные программы, реализованные только на iOS:

Например, приложение Photomath, позволяющее пошагово решать математические задачи, например квадратные уравнения.

Complete Anatomy использует 3D – модели и возможности дополненной реальности в области человеческого тела. В приложении студенты могут изучить строение тела и экспериментировать, не используя ничего, кроме iPad.

Приложение Logic Remote позволяет записывать музыкальные дорожки, а также научиться редактировать треки при помощи сенсорных микшеров.

Приложение Explain Everything Whiteboard – предоставляет интерактивную доску для удобного рисования диаграмм, записи текста и незаменимо в лекциях.

Hudl Sportscode – позволяет ведущим факультетам физической культуры анализровать качество тренировок для спортсменов, и, опираясь на данные, помочь игроку добиться лучших результатов.

Помимо этого, Apple предоставляет скидки для учащихся учебных заведений, а также для сотрудников. Льготы затрагивают как ПО, так и технику.

## 1.3. Обзор существующих решений для ВУЗов

Идея существования информационного приложения для ВУЗа появилась довольно давно и уже продолжительное время многие учебные заведения и сторонние разработчики занимаются разработкой решений для организации удобного доступа к сервисам ВУЗов. Многие из этих решений достаточно гибкие и применимы в других учреждениях, помимо тех, для которых они разработаны. Некоторые из них предоставляют лишь доступ к нескольким сервисам, что может быть частью большого объединяющего приложения. Рассмотрим аналоги приложений среди российских и западных ВУЗов.

*Физтех.Расписание* – содержит актуальное расписание для групп университета. Карточка расписания пары содержит: название пары, время продолжительности пары, преподавателя, аудиторию. Занятие может быть локально отредактировано на телефоне, для отображения информации о нем по собственному желанию. В приложении поддерживается светлая и черная тема.

Плюсы – Физтех.Расписание – одно из немногих приложений, не требующих авторизацию в приложении. Решения многих вузов требуют вход в личный кабинет при запуске приложения. Таким образом, нет возможности получить даже общедоступную информацию, которая, например, есть на веб-ресурсах ВУЗа. Также плюсом является возможность добавления виджетов о расписании на главный экран. Есть возможность настройки уведомлений о парах. Отсутствие рекламы.

Минусы – ориентировано на студентов. Нет расписания для преподавателей. Работает только на новых устройствах iOS 14+.

*Вышка.Навигация* – приложение для поиска аудиторий в корпусах НИУ ВШЭ. Приложение поддерживает поиск аудиторий, выбор кабинетов на карте. Строит маршрут между аудиториями. Позволяет выбрать необходимый этаж для загрузки нужной карты. Карта поддерживает масштабирование. Есть окно для смены корпуса.

Плюсы – хорошая детализация карт – отмечены аудитории, лифты, коридоры, туалеты. Удобный пользовательский интерфейс. Высокая доступность – необходимая версия iOS – 9.0 или новее.

Минусы – аспект реализации. Лаги при масштабировании карты, местами – неочевидные решения для пользователя – например, скролящийся индикатор этажей для пользователя. После выбора аудитории – часть полезной части экрана пустует, при том, что изначально там находилось карта. По причине ошибок в верстке, главный экран, местами, разного цвета.

*ТПУ Абитуриент* – приложение Томского политехнического университета для абитуриентов. Предоставляет возможность оценить свои силы при поступлении в институт. Содержит информацию о конкретном студенте (его баллах, индивидуальных достижениях). В приложении реализована группировка специальностей по институтам. Для каждой специальности указывается – форма обучения, количество мест, число поданных заявлений, текущий конкурс, проходной балл.

Плюсы – удобная реализация анализа конкурсной ситуации для мобильных устройств. В приложении имеется вся необходимая информация о поступлении. Доступно с iOS 7.0, то есть приложение будет работать от iPhone 4S и новее.

Минусы – обязательная регистрация, что дает невозможным просмотр даже той информации, которая есть в открытом доступе на сайте. С 2021 года изменяется структура рейтингов поступающих [6]. Узнать собственную позицию в конкурсных списках может будет только используя СНИЛС или специальный код. Нет информации о том, как будут выглядеть новые списки в приложении и как будет осуществлен к ним доступ.

*Trinity College App (Ирландия)* – приложение информационного характера о событиях в Тринити Колледже. Содержит новости, календарь событий, информацию о спортивных секциях, контакты учебного заведения, а также онлайн магазин с атрибутикой с символикой колледжа. Работает в связке с приложением SEQTA, предоставляющего личный кабинет для обучающихся. Содержит: расписание, оценки, внутреннюю социальную сеть, курсы обучающегося.

Плюсы – Большой объем полезной информации. Удобный плиточный интерфейс в главном меню. Многие из функций не требуют авторизации. Для личного кабинета, полностью пропадает смысл в использовании веб-версии.

Минусы – в приложении стилистически не выдержано однообразие интерфейсов для различных разделов. За большим объемом информации скрывается, местами, перенасыщение информации, которую не всегда удобно просматривать на мобильном устройстве. Особенно, это касается части SEQTA, реализующую студенческий личный кабинет.

*MIT Mobile (США)* – приложение Массачусетского технологического института. Содержит новости учебного заведения, схемы проезда к кампусам из разных частей Бостонской агломерации, в которой он располагается. Cодержит подробную карту и навигацию по студенческому городку с поиском необходимых помещений и описанию этого помещения. Информация о библиотеках, точках питания.

Плюсы – большое количество сервисов, ориентированных как на обучающегося, так и на гостей кампуса. Удобный плиточный интерфейс главного экрана, подобно Тринити Колледжу. Возможность обратной связи.

Минусы – Не все из сервисов рабочие. Некоторые из интерфейсов устарели (например, карт) и, в дальнейшем, должны быть доработаны с учетом Apple HIG. Некоторые разделы приложения напоминают веб-сайты, реализованные в рамках приложения.

Как вывод, для удобства использования, разрабатываемые приложения для ВУЗов должны сочетать перечисленные аспекты, обладать необходимым функционалом, минимизировать недостатки существующих решений.

## 1.4. Требования к системе.

Существует множество решений от ВУЗов России и мира, реализующие отдельные компоненты взаимодействия с учебным заведением. На сегодняшний день, не так много вариантов, собравших в себя все самые удобные сервисы в одном месте. В БГТУ им. Шухова нет мобильного приложения для быстрого доступа к необходимой информации. В рамках дипломной работы представлена система, предполагающая доступ к следующим разделам:

•  Расписание для групп институтов университета;

•  Навигация внутри корпусов ВУЗа;

•  Поиск преподавателей и их актуальное расписание;

•  Пункты питания на территории университета;

•  Личный кабинет для студента/сотрудника.

Функциональные возможности разработанной системы:

1. Использование в качестве устройства iPhone на базе iOS 12.0 и новее. Список поддерживаемых устройств: iPhone 5S и новее, iPod Touch 6-го поколения и новее.

2. Для студенческого расписания: Поиск необходимого института, группировка учебных групп по номеру курса, экран группы – выбор типа недели (числитель / знаменатель), просмотр карточек с информацией о занятиях, реализация экзаменационного расписания.

3. Реализация навигации по корпусам университета. Выбор аудитории в качестве точки отправления/прибытия на карте этажа или в экране поиска. Функция поиска аудитории. Построение кратчайшего пути, а также рисование маршрута на карте. Выбор необходимого этажа. Выбор корпуса. Реализация масштабирования карты, изображение маркеров пунктов на карте. Реализация базы данных для хранения информации об аудиториях.

4. Поиск преподавателей университета. Экран расписания для каждого преподавателя с возможностью выбора типа недели, экзаменационного расписания, заочного расписания. Выгрузка данных из API.

5. Личный кабинет для учащихся и сотрудников. Авторизация в приложении, запоминание информации о пользователе для последующих входов в приложение. Выход из аккаунта. Получение информации о сотруднике. Для студента, реализация модуля для просмотра оценок аттестации в семестре, просмотр сессионных оценок, расписание для собственной группы.

**Выводы по главе.** Информационные технологии в образовании, в частности, в мобильных устройствах, каждым годом все сильнее становятся ее неотъемлемой частью. Главными выгодами от их использования является продуктивность и удобство использования, благодаря чему происходит экономия времени. Важно, чтобы разработанные продукты обеспечивали быстрый доступ к необходимым ресурсам, сервисам и платформам. Apple, на сегодняшний день, является разработчиком одной из двух самых популярных мобильных платформ. Благодаря магазину приложений App Store, самые популярные образовательные порталы сегодня доступны на телефоне и планшете. Также развиваются совершенно новые проекты с учетом современных тенденций на развитие дополненной реальности и 3D – графики, что полезно в сфере медицины. Многие российские вузы занимаются разработкой собственных приложений, например, автоматизирующие некоторые процессы для использования студентами и преподавателями в повседневной жизни. В связи с этим, для БГТУ им. Шухова актуальна разработка мобильного приложения с учетом необходимых сервисов для использования сотрудниками института для платформы iOS с последующей публикаций в App Store для скачивания [7].

# Глава 2. Архитектурные особенности приложения и принцип работы

## 2.1. Структура работы приложения

Рассмотрим варианты использования приложения. Для разработки программного обеспечения, вне зависимости от используемых методологий и технологий, на первом этапе происходит формулировка требований к продукту. Для реализации формализации функциональных требований хорошо подходят созданные в языке UML диаграммы использования (Use Case Model) [8]. Она состоит из графической диаграммы, в которой описываются действующие лица, взаимодействующие с системой (актеры), прецеденты – варианты использования или процессы, выполняющиеся пользователями, которые дают пользователю ощутимый результат, отношения между элементами диаграммы, а также комментарии. Прецеденты следует дополнить словесным описанием, где необходимо, в числе прочего, указать, какая последовательность действий является главной, а какая – альтернативной.

Между прецедентами используются 4 отношения: расширения (extend), включения (include), обобщения (generalization) и ассоциации (association) отображаемые на диаграммах.

Отношение ассоциации – линия (стрелка – если инициируется необходимо указать направление связи) между актером и вариантом использования, отражающая их связь.

Отношение включения указывает, что поведение одного прецедента включается в другой прецедент в качестве составного компонента (от базового к включаемому направлена пунктирная стрелка «include»). Является обязательным поведением системы.

Отношение расширения означает присоединение одного варианта использования к другому базовому (к нему направлена пунктирная стрелка «extend») в некоторой точке и выполняется лишь при определенных условиях. Является необязательным поведением системы.

Отношение обобщения – прецедент A является потомком или предком по отношению к прецеденту B. Потомок наследует поведение своего родителя и может быть дополнен.

Продемонстрируем диаграммы использования UML частей разрабатываемого приложения.

Раздел «Расписание»

****

Рис. 1 – Use – Case Model раздела «Расписание»

При входе в приложение, для пользователя загружается раздел «Расписание», как самая популярная и повседневно необходимая часть. Подробнее о переходе между разделами будет рассказано при проектировании графического интерфейса.

Раздел «Расписание», первоначально, должен предоставлять выбор необходимого института. Этот выбор влияет на дальнейшую загрузку групп, которые соответствуют выбранному институту. Группы сгруппированы по курсу обучения. Также, для ускоренного доступа необходимо реализовать поиск, позволяющий в режиме реального времени отображать необходимые группы, согласно информации в строке поиска. И для получения списка институтов, и для списка групп используется существующее API, возвращающее на клиент (телефон) информацию, подлежащую отображению.

В процессе выбора института или группы может случиться ситуация, когда потеряно соединение с интернетом на телефоне или сервер не отвечает по запросам. Тогда необходимо отобразить пользователю соответствующее сообщение на экран с возможностью повторной попытки загрузки.

После входа в экран расписания, приложение определяет текущую дату на сегодня, загружает данные о занятиях на ближайшие 2 недели с сервера. В числе прочего, с сервера приходит информация о типе недели числителе или знаменателе. На экране появляется расписание на текущую дату, название группы, тип недели, активируется кнопка, дающая доступ к экзаменам. При нажатии на «Тип недели», должен быть осуществлен способ выбора следующей недели. Возле названия текущей недели появляется постфикс – текущая неделя. Например «Числитель – текущая неделя».

Расписание пар представляет собой карточки, на каждой из которых есть информация о: названии пары, типе занятия (лекция, лабораторная, практика, зачет, экзамен), времени, аудиторий, преподавателях.

Карточки экзаменов структурно выглядят аналогичным образом, за исключением того, что поле времени также содержит информацию о дате.

Раздел «Навигация»

****

Рис. 2 – Use – Case Model раздела «Навигация»

При доступе пользователь в раздел «Навигация», автоматически открывается карта 1-го этажа Главного учебного корпуса. Глобально, на экране имеется информация о текущем выбранном корпусе, доступ к окну поиска аудиторий, возможность смены корпуса, а также способ использовать смену этажа для загрузки другой карты.

Как только карта загружена, пользователь может изменять ее маcштаб, а также перемещать при помощи пальцев. На карте имеются маркеры, соответствующие помещениям на карте. По мере отдаления или приближения карты, некоторые маркеры не отображаются, чтобы не было перекрытий со стороны других маркеров, и карта оставалась читаемой.

Маркер помещения может быть выбран нажатием. В таком случае – он является приоритетным для рисования и виден при любом масштабе. Нажатому маркеру соответствует окно помещения, в котором есть возможность выбрать его либо в качестве пункта отправления, либо в качестве пункта прибытия. Окно помещения может быть закрыто, если его потянуть вниз, либо по нажатию на экран.

Окно поиска аудиторий позволяет в поиске написать название необходимого кабинета, а затем, приложение загрузит карту этажа необходимой аудитории и перенаправит камеру на нее.

После выбора двух кабинетов – отправления и прибытия – приложение строит кратчайший маршрут между ними. Если аудитории располагаются на разных этажах, то на экране появляется вспомогательная кнопка, позволяющая в один клик переходить между этажами, через которые будет отображаться необходимый маршрут.

Раздел «Преподаватели»

****

Рис. 3 – Use – Case Model раздела «Преподаватели»

Первичный экран с разделом «Преподаватели» должен содержать список первых букв фамилий, что ускорит поиск. В сгруппированном по первой букве фамилии списке можно найти необходимого преподавателя вручную, а можно воспользоваться поиском, который будет выдавать нужную информацию в зависимости от введенного текста. Экран расписания преподавателя не имеет больших отличий, в сравнении с расписанием для студентов. При загрузке появляется расписание на текущий день, есть возможность выбора типа недели. Расписание занятий для очных групп, аналогично, состоит из карточек. Карточки, вместо информации о преподавателях, содержат информацию о группах.

В отдельном окне, с активностью в определенные даты (консультации и экзамены у очных групп), дополнительно содержится информацию о парах заочных групп.

Раздел «Пункты питания»

Дает возможность отобразить меню пунктов питания, работающих на территории ВУЗа. Меню сгруппировано по разделам и содержит карточки, в которых содержится информация о: названии блюда, его весе, энергетической ценности, а также стоимости в рублях.

Раздел «Личный кабинет»

****

Рис. 4 – Use – Case Model раздела «Личный кабинет»

Для доступа к функциям личного кабинета необходима авторизация. Существует 2 варианта:

1 - В приложении нет сохраненных данных о пользователе

2 - Пользователь уже входил в систему и его учетные данные сохранены на телефоне

В первом случае, на экране должно появиться окно, где пользователю будет предложено ввести логин и пароль для входа в систему. Также, в окне будут расположены вспомогательные кнопки в случае проблем входа.

После ввода данных, приложением будет предпринята попытка входа в систему. Результат может быть следующим: ошибка подключения к интернету, неверные данные учетной записи или успешная авторизация. В случае первых двух вариантов, пользователь получает соответствующую информацию на экране и имеет возможность повторить попытку. В случае успешной авторизации, открывается главная страница личного кабинета.

В случае, если приложение хранит данные пользователя, им будет предпринята попытка входа с имеющимися данными. В случае отсутствия подключения к интернету будет предложена возможность повторить попытку. В случае, если имеющиеся данные пользователя невалидны (что возможно, например, при смене пароля), произойдет сброс данных из приложения и переход на страницу входа. Дальнейшие действия аналогичны первому способу авторизации.

В главном экране личного кабинета: сотрудник может найти информацию о себе в контексте института: свой институт, статус. В случае студента, дополнительно отображается - группа, номер зачетной книжки, форму обучения, а также информацию об оплате. Существует также раздел личной информации, в которой содержится: дата рождения, гражданство, телефон, почта, номер Снилс.

Студент имеет возможность просмотреть информацию о своих оценках в разделах «Успеваемость» и «Аттестация». «Успеваемость» представляет собой информацию о сданных студентом сессиях. Информация в них сгруппирована по очередности сессий. «Аттестация» позволяет узнать текущие оценки промежуточной аттестации, 2 раза за семестр.

Раздел «Расписание», доступный и для преподавателей, и для студентов, описан ранее. В случае студента, для получения расписания используется название относящейся к нему группы, а в случае преподавателя – его уникальный идентификатор (id).

## 2.2. Принципы функционирования модулей

*Раздел «Расписание»*

Для загрузки расписания группы необходимо последовательно совершить 3 запроса: 1 – для загрузки списка институтов, 2 – для загрузки списка групп для выбранного института, 3 – загрузка расписания группы. Опишем последовательность подробнее:

1) Загрузка списка институтов.

Способ: GET- запрос на [*http://info.bstu.ru/index.php/*](http://info.bstu.ru/index.php/)

Ответ: HTML

Дальнейшими действиями является парсинг полученных данных и вывод на экран. Сохранение ссылок (href) на список групп.

2) Загрузка списка групп.

Способ: Для каждого института - GET- запрос на *http://info.bstu.ru\{ссылка из пункта 1}*

Ответ: HTML

Далее, происходит парсинг полученных данных и вывод на экран в соответствующих формах.

3) Загрузка расписания группы.

Способ: Названия групп, полученные при запросе 2, используем для совершения GET- запроса: *http://cabinet.bstu.ru/api/1.0/timetable?role=student&groupName=\(groupName)&wholeWeek=true*

Ответ: JSON.Определение типа недели, заполнение данных по типу недели (для очных групп). Группировка экзаменов в отдельную категорию. Вывод данных на экран после подстановки в необходимые формы.

*Раздел «Навигация»*

Для организации навигации и поиска аудиторий, для начала необходимо организовать способ их хранения. Разработаем БД.

Diagram

Description automatically generated

Рис. 5 – Схема БД раздела «Навигация»

Информация о картах содержится в MapDB. В ней описываются характеристики конкретной карты. Она содержит информацию о корпусе, а номер этажа, изображение карты в формате Blob, а также матрицу кратчайших путей (массив деревьев кратчайших путей) и матрицу кратчайших расстояний. На двух последних пунктах остановимся подробнее позднее.

Каждое помещение (PremiseDB) относится к определенной карте, соответствующей конкретному корпусу и этажу. Помещение имеет определенный тип (PremiseTypeDB), указывающий на ее предназначение, например: кабинет, буфет, гардероб, туалет и тд. Эта информация может быть полезна в окне выбора аудитории. Каждое помещение имеет свое название, отображающееся на карте, а также описание.

Каждому помещению соответствует маркер на карте (MarkerDB). Ему соответствуют координаты x, y, которые говорят о его положении на карте.

Информация о точках дорог, через которые проходит путь между помещениями, содержится в MapRoadDB. Точка дороги соответствует определенной карте и соединена с помещением посредством полей: idPremise, audienceX, audienceY.

Когда аудитория выбирается в качестве пункта отправления или назначения, ее идентификатор (id) запоминается в качестве уникального значения, впоследствии, использующейся для построения маршрута.

Поиск кратчайшего маршрута между несколькими помещениями, как математическая задача, может быть решен Алгоритмом Флойда - Уоршелла с модификациями.

Идеей базового алгоритма Флойда – Уоршелла является постепенное улучшение оценки кратчайшего пути между парой вершин до тех пор, пока оценка не станет оптимальной. Для пар вершин (i, j) может существовать такая промежуточная вершина k, при которой суммарный путь i -> k -> j, окажется короче пути i -> j напрямую. В алгоритме, каждая вершина графа выбирается как k и через нее происходит попытка улучшения пути i, j.

Опишем базовый алгоритм Флойда – Уоршелла в качестве псевдокода:

for each edge (u, v) do

dist[u][v] ← w(u, v) // Вес ребра (u, v)

for each vertex v do

dist[v][v] ← 0

for k from 1 to |V|

for i from 1 to |V|

for j from 1 to |V|

if dist[i][j] > dist[i][k] + dist[k][j]

dist[i][j] ← dist[i][k] + dist[k][j]

end if

Данный алгоритм помогает найти матрицу кратчайших расстояний. Так как нам необходимо иметь информацию о самих маршрутах, то для каждого пункта введем дерево кратчайших путей, которое будем модифицировать в процессе нахождения кратчайшего расстояния в алгоритме Флойда – Уоршалла.

Матрица деревьев кратчайших путей инициализируется следующим образом:

Если сообщение i -> j существует, то кратчайший путь – напрямую, то есть path[i][j] = j.

Если i == j, то это одна и та же вершина, и их кратчайший маршрут соответствует той же самой вершине: path[i][j] = i.

Добавим в алгоритм Флойда - Уоршалла пункт модификации матрицы деревьев кратчайших путей.

if dist[i][j] > dist[i][k] + dist[k][j]

dist[i][j] ← dist[i][k] + dist[k][j]

path[i][j] ← path[i][k]

end if

Тогда, для восстановления пути, необходимо, используя индекс вершины на карте, поочередно посещать вершины, пока не будет достигнута конечная.

// v1 – начальная вершина

// v2 – конечная вершина

if bestWaysMatrix[v1][v2] == null

return []

end if

pathArray.append(v1)

while v1 != v2

v1 = bestWaysMatrix[v1][v2]

pathArray.append(v1)

end while

Для разгрузки клиента в плане скорости вычисления кратчайшего пути используем следующие улучшения:

1) Необходимо заранее, в БД загрузить уже готовую матрицу кратчайших путей и матрицу кратчайших расстояний.

2) Алгоритм Флойда – Уоршалла применяется к помещениям определённого этажа. Для этого, необходимо создать скрипты, выполняющие вычисления и сохранить их результат в БД.

3) Клиент (телефон), при помощи регулярных выражений парсит данные матриц, сохраняет, и использует по назначению.

Когда необходимо строить путь между пунктами, находящимися на разных этажах, для корпусов, имеющих 2 входа, (например, Главный корпус) в действительности, строится 2 пути, поскольку этаж имеет 2 входа. Среди построенных путей, при помощи матрицы кратчайших расстояний, выбирается наименьший по расстоянию и называется кратчайшим путем.

При масштабировании карты, не все из маркеров могут быть отображены. При минимальном масштабе, рисуются маркеры обладающие высоким приоритетом (выделенные пользователем) и наибольшие по размеру. Для отсутствия перекрытий между маркерами, используется следующий принцип:

1) При увеличении масштаба, среди всех неторисованных маркеров, отсортированных в порядке приоритета и размера, выбираются те, которые не перекрывают уже нарисованные и отрисовываются.

2) При уменьшении масштаба, среди всех отрисованных маркеров, отсортированных в порядке приоритета и размера, выбираются те, которые перекрывают уже нарисованные и не отрисовываются.

*Раздел «Преподаватели»*

Для получения расписания преподавателя, необходимо совершить 2 последовательных запроса:

1) Получение списка преподавателей по первой букве фамилии:

Способ – GET – запрос на [*https://www.bstu.ru/about/useful/schedule/staff/\(suffixUrl)*](https://www.bstu.ru/about/useful/schedule/staff/\(suffixUrl))*,* где suffixURL – первая буква фамилии. В качестве заголовка необходимо также указать *"X-Requested-With": "XMLHttpRequest"* как эмуляция запроса без перезагрузки страницы, аналогично тому, как это происходит в веб – версии сайта.

Ответ – HTML

Ответ содержит, помимо данных о сотрудниках, ссылки на расписание конкретного преподавателя.

2) Получение расписания конкретного преподавателя:

Способ – GET запрос по ссылке из пункта 1. Образец: [*http://www.bstu.ru/static/themes/bstu/schedule/index.php?tid=1111*](http://www.bstu.ru/static/themes/bstu/schedule/index.php?tid=1111)

Ответ – HTML

Определение типа недели, заполнение данных по типу недели (для очных групп). Группировка экзаменов и заочного расписания в отдельную категорию. Вывод данных на экран после подстановки в необходимые формы.

*Раздел «Пункты питания»*

Получение информации о меню пунктов питания осуществляется в 2 шага:

1) Запрос для получения названий и идентификаторов пунктов питания

Способ – GET – запрос на *http://menu.bstu.ru/*

Ответ - HTML

Получение необходимой информации для следующего запроса. Названия пунктов служат идентификацией конкретного заведения для пользователя.

2) Получение меню для всех точек питания

Способ – POST – запрос на [*http://menu.bstu.ru/*](http://menu.bstu.ru/)c указанием в теле запроса [point: {id точек, полученные на предыдущем шаге}]

Ответ - HTML

Формирование информации для передачи контроллеру, который формирует вывод данных на экран.

*Раздел «Личный кабинет»*

Как было сказано ранее, для использования личного кабинета, необходимо выполнить вход в систему.

Способ – POST – запрос на [*http://cabinet.bstu.ru/auth/login*](http://cabinet.bstu.ru/auth/login)с указанием в теле запроса email и password.

Ответ – HTML, содержащий данные об успешности входа. Заголовок документа содержащий текст «Вход в систему» или сообщение об ошибке будет признан ошибочным. При успешном входе, в заголовке документа содержится тэг script, внутри которого, c префиксом define содержится json c информацией об авторизированном пользователе. Часть из этой информации (например, институт, группа, форма обучения и тд.) используется для отображения информации о пользователе. Таким образом, выполнен вход в систему. Сохраняются данные о пользователе для повторного входа.

Функциональность личного кабинета для студента включает в себя получение информации об аттестации и успеваемости, а также расписание группы.

Получение информации об аттестации:

Способ – GET – запрос на *"http://cabinet.bstu.ru/api/1.0/certification?dataId=\(idUser)"*

Ответ – JSON, содержащий данные об оценках промежуточных аттестаций каждого студента.

Получение информации об успеваемости:

Способ – GET – запрос на *"http://cabinet.bstu.ru/api/1.0/progress?dataId=\(idUser)"*

Ответ – JSON, содержащий данные об оценках, полученных студентом за периоды сессий.

Загрузка расписания группы:

Способ: Название группы, полученной при авторизации студента, используем для совершения GET- запроса на: *http://cabinet.bstu.ru/api/1.0/timetable?role=student&groupName=\(groupName)&wholeWeek=true*

Ответ: JSON. Определение типа недели, заполнение данных по типу недели (для очных групп). Группировка экзаменов в отдельную категорию. Вывод данных на экран после подстановки в необходимые формы.

Для выхода из аккаунта необходимо:

Отправить при авторизированной сессии GET – запрос на: <http://cabinet.bstu.ru/auth/logout>.

Ответ: HTML. При анализе полученного заголовка документа, в тэге script, внутри которого, c префиксом define содержится json c информацией об авторизированном пользователе, id – пользователя должен быть равен «undefined». Иначе, выход из аккаунта успешным не является. После выхода из системы, из приложения удаляется личная информация пользователя для быстрого входа.

## 2.3. Архитектура приложения

На основе описанной информации о функциональности, необходимо выбрать архитектурный паттерн проектирования для модулей [9], удовлетворяющий требованиям.

Apple рекомендует в общем случае использовать паттерн Cocoa MVC [10], который основан на стандартном MVC, но имеет свои особенности. Например, View – отвечающий в стандартном MVC за представление (это все, что связано с графическим интерфейсе) и Controller – отвечающий изменения в модели, реагируя на действия пользователя и обновляя представления объединены в одном классе UIViewController.

Diagram

Description automatically generated

Рис. 6 – Cocoa MVC

Имеем ситуацию, что View и Controller фактически являются одной сущностью. В итоге, View Controller, содержащий и код для отображения графических элементов, и код для реакции на нажатия, свайпы, и, в то же время, код для запросов в модель, парсинг полученных данных и тд. Размер ViewController будет увеличиваться с увеличением сложности графического интерфейса и объемом функциональности, связанной с данными. Возрастет также и сложность тестирования. В нашем случае, раздел «Навигация» даже при выделении отдельных сущностей под компоненты графического интерфейса, будет обладать достаточно большим размером, чтобы дополнительно добавлять логику. В связи с этим, есть смысл рассмотреть другие архитектурные паттерны.

Если выделяется отдельная сущность Presenter – посредник между Представлением и Моделью, который не имеет отношения к жизненному циклу View, а лишь обновляет его в соответствии с новыми данными и состоянием, то такой паттерн называется MVP. Такое разделение полномочий позволяет протестировать бизнес – логику в Presenter, отдельно от графического интерфейса.

Как развитие MVP, рассматривается паттерн MVVM, который использует в качестве посредника представления и данных ViewModel, вместо Presenter. Их отличие в том, что ViewModel, в отличии от Presenter, сам не обновляет представление, а связан с ним посредством биндинга. Когда ViewModel обновляется (например, когда изменился Model), то View, будучи «привязанным» также обновляется. Автоматическое распространение потоков данных соответствует парадигме реактивного программирования. Ее плюсом является возможность написания кода в функциональном стиле, минусом – большой стек вызовов сильно затрудняет отладку приложения. Архитектура, по – прежнему, хорошо тестируема, имеет распределение обязанностей. В сравнении с MVP, View имеет больше полномочий так как должен самостоятельно выполнять свое обновление.

Diagram

Description automatically generated

Рис. 7 – MVVM

Из рассмотренных архитектур, в соответствии с нашими задачами, выбираем MVVM для реализации разделов, в связи c хорошим разделением полномочий, возможностями для тестирования, возможностями переиспользовать ViewModel, так как он не зависит от конкретных представлений, сравнительно небольшим числом реализации интерфейсов (допустим, если бы использовали VIPER с еще большим разделением полномочий). Переходы между разделами осуществляются посредством корневого навигационного контроллера приложения.

## 2.4. Проектирование графического интерфейса. Apple HIG.

Для создание качественных дизайнов приложений необходимо соответствовать разработанному Apple руководству по дизайну Human Interface Guideline. Из кратких его положений, поведение приложений должно интегрироваться с его функцией, содержать стандартные элементы управления, знакомые пользователю, а также обеспечивать предсказуемое поведение.

Через прямое манипулирование устройством, например жестами, свайпами, прямыми перемещениями объектов на экране, прокручиваниями достигается соответствие предметам реального мира и более удобно для пользователя. Например, это касается переходов к предыдущим экранам, закрытием подсказок и тд.

Результаты действий должны проясняться пользователю. К примеру, при загрузке данных, на экране должна появиться индикаторы с анимацией загрузке. Разнообразные уведомления также должны сопровождаться звуком, специальными модальными окнами.

Для проектирование графического интерфейса экранов приложения Apple предлагает использовать Storyboard, а для небольших элементов интерфейса, которые могут использоваться повторно (например, определенный дизайн строк таблиц, кастомные переключатели и тд.) – Xibs.

Storyboard и Xib представляют собой XML – файлы, которые генерируются при заполнении их графическими элементами. При проектировании Storyboard и Xib удобно использовать AutoLayout, который при установке ограничений в размерах и взаиморасположении графических элементов, автоматически, сформирует верное отображение на различных размерах экранов. Преимуществами использования также является возможность мгновенно увидеть результаты своих действий, а также не писать шаблонный код для UI. Однако некоторые действия, например анимации, очевидно, описываются программно. Также, минусом использования является сложность в исправлении конфликтов ограничений графических компонентов.

Так как приложение имеет несколько разделов, то доступ к ним удобно осуществить при помощи боковом меню (Hamburger menu). Оно должно вызываться либо по нажатию кнопки в шапке любого из разделов, либо по свайпу слева – направо из представлений всех разделов (кроме «Навигации»).

Одним из основных компонентов при отображении списков являются таблицы. Таблицы используются: при отображении списка институтов, списка групп, списка преподавателей. Для более удобной работы с таблицей в ее заголовке должна быть расположена строка поиска для ввода информации, по данным который, фильтруется контент в таблице и отображается необходимый.

При проектировании страницы расписания необходимо в ее верхней части разместить информацию о сегодняшнем дне неделе, группе (или фамилии преподавателя), типе недели (если выбран преподаватель или группа очная). Ниже, располагается панель с датами и днями недели. По нажатию на которую происходит анимация переходу к расписанию нового дня. Расписание занятий представляет собой скролящееся вниз представление, содержащее карточки – представление view с информацией о занятиях. Один из нижних слоев карточек имеет тень, для более четкого выделения на экране. Помимо доступа к расписанию занятий, на верхней панели в правой части размещается кнопка доступа к экзаменам, по нажатии на которую открывается окно, содержащее карточки для экзаменов. Выход из окна доступен либо по нажатии на крест вверху представления, либо по нажатию вне представления.

При проектировании навигационной страницы необходимо в верхней части разместить информацию о текущем выбранном корпусе, а также формы для ввода пунктов отправления и прибытия. По нажатии на эти формы, открывается таблица, предлагающая выбрать пункт в соответствии с набранным текстом. Также, в правой стороне верхней части располагается кнопка выбора корпуса. По ее нажатию, снизу экрана, всплывает окно, в котором выбирается необходимый корпус.

Карта расположена на элементе, поддерживающим скроллинг во все 4 направления, а также зум посредством пальцев. На карте расположены маркеры, не масштабирующиеся при изменении размеров карты. При выборе маркера, снизу, при помощи анимации, появляется окно описания аудитории. Рисование маршрута представляет собой отображения с использованием анимации полоски оранжевого цвета от пункта отправления до пункта назначения.

В нижней левой части экрана расположен переключатель этажей. Если необходимо нарисовать маршрут, то в нижней правой части экрана по необходимости появляется кнопка переключения этажа, а переключатель этажей блокируется.

При реализации интерфейса меню пунктов питания удобно использовать выпадающее меню для выбора заведения, а в таблице ниже отображать необходимую информацию по меню.

Интерфейс входа в Личный кабинет представляет собой 2 формы для ввода логина и пароля, а также кнопку входа и различную вспомогательную информацию. Главная страница Личного кабинета содержит view для отображения основной информации, список с личной информацией с возможностью скрытия, а также таблицу с разделами – успеваемость, аттестация, расписания. Так как все из разделов требуют доступа к интернету то пользователь информируется в виде специального представления с крутящимся колесом о процессе загрузки. Экраны «Успеваемость» и «Аттестация» представляют собой таблицы, с возможностью анимированного открытия и закрытия нужных ячеек для отображения необходимых оценок.

**Выводы по главе:** в данной главе была подробно рассмотрена структура работы приложения в виде use-case диаграмм. Было рассмотрено приложение с точки зрения принципа функционирования, использования внешних источников данных, алгоритмов. Рассмотрены различные архитектурные варианты реализации разделов и выбран MVVM в качестве основного. Рассмотрели принципы проектирования графического интерфейса и указали на основные графические элементы.

# Глава 3. Описание программной реализации iOS-приложения

## 3.1. Используемые технологии

Для реализации приложения используем нативный язык для разработки iOS – приложений Swift 5. В качестве среды разработки используем Xcode 11. Инструмент для контроля версиями – Git. В качестве менеджера зависимостей используется Cocoa Pods.

Библиотеки для реализации архитектуры MVVM при помощи реактивного программирования – RxSwift [11][12] и RxCocoa.

Веб запросы выполняются при помощи Alamofire. Для реализации парсинга JSON используется библиотека SwiftyJSON [13], для парсинга HTML – SwiftSoup [14].

Для работы с базой данных используется библиотека SQLite.Swift [15]. Заполнение БД при помощи Python и ORM для работы с БД peewee.

Для написания Unit – тестов используется фреймворк XCTest, а также Mock – сервер Hippolyte.

## 3.2. Разработка модуля «Расписание»

Рассмотрим основные моменты, связанные с программированием.

Как было сказано ранее, разработка графических интерфейсов ведется при помощи Storyboard и Xib. Тогда, в Xcode необходимо в окне разработки графического элемента указать класс, с которым он связан:

Graphical user interface, text, application

Description automatically generated

Рис. 8 – Связывание класса InstitutionsViewController и Storyboard в Xcode

Далее, графические элементы, присутствующие в Storyboard (или xib) как-либо изменяющие свои свойства в процессе эксплуатации приложения, должны быть задекларированы в разрабатываемом классе и быть помеченными как @IBOutlet.

Пример загрузки Storyboard в коде:

****

После подключения всех графических элементов, необходимо писать логику для обработки нажатий, свайпов и тд. Раздел «Расписание» состоит из 3 последовательных окон: Список институтов, список групп, расписание группы.

Для многих графических элементов имеются привязки к RxSwift, позволяющие декларативно, при помощи замыканий описывать реакцию на действия пользователя. Если нет привязки к RxSwift, то необходимо указать текущий класс – делегатом элемента, подписаться на протокол – делегат (в понимании других языков программирования – интерфейс) и реализовать необходимый метод. Типичным таким графическим элементом является кнопка. Тогда, действие на ее нажатие описывается следующим образом.

**self**.menuButton.rx

.tapGesture()

.when(.recognized)

.subscribe(onNext: { [**weak** **self**] **\_** **in**

**// Реакция**

}).disposed(by: disposeBag)

В таком случае, замыкание становится подписчиком нажатия на кнопку и асинхронно вызывается по необходимости. Если внутри замыкания используется доступ к полям экземпляра (требующих) self, тогда необходимо указать ссылку на self как слабую. Иначе, эта ссылка рассматривается механизмом ARC (Automatic Reference Counter) как сильная и в случае удаления текущего контроллера представления не вызовет его деструктор, поскольку замыкание все еще указывает на self сильной ссылкой. Если бы указывало слабой, то планировщик посчитал бы, что ссылок, указывающих на self нет и необходимо вызывать деструктор. Повторный вызов DisposeBag способствует отмене подписок.

Аналогичная парадигма используется и при привязке данных к таблице UITableView. Фактически таблица является подписчиком данных. Эти данные могут быть постоянными, а могут изменяться. В первом случае, сделать данные наблюдаемыми можно при помощи следующей конструкции:

**let** data = Observable.just(institutions)

Тогда, типичная конфигурация таблицы выглядит следующим образом:

Graphical user interface, text, application, email

Description automatically generated

В случае постоянно обновляющихся данных, например, при поиске необходимой группы, необходимо использовать BehaviorRelay, который, при получении новых значений при помощи метода accept, асинхронно рассылает их своим подписчикам.

Пример:

1. Привязка данных и таблицы

Graphical user interface, text, application, email

Description automatically generated

2. Рассылка подписчику новых данных

****

Так как в архитектуре MVVM ViewModel, отвечающий за обработку запросов пользователя и работу с Model не имеет непосредственной ссылки на View и коммуникация между ними должна осуществляться посредством биндингов [16], то RxSwift хорошо подходит для реализации данной задачи.

Представление (ViewController) содержит в себе ссылку на ViewModel и при его обновлении, обновляется сам. Пример:

Graphical user interface, text

Description automatically generated

Методы из ViewModel совершают запросы в сеть, используя Alamofire и парсят полученные данных при помощи SwiftSoup для HTML и SwiftyJSON для JSON.

Разработанные экраны:

Расписание может быть выбрано в боковом меню:

Graphical user interface, application

Description automatically generated

Рис. 9 – Боковое меню

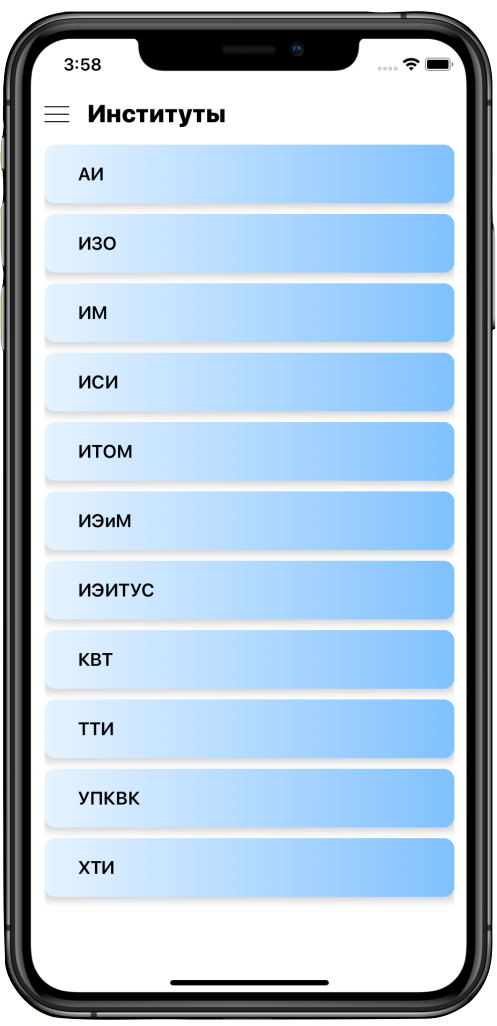
 

Рис. 10 – Поиск института Рис. 11 – Выбор группы

Для разработки экрана расписаний используем UIView для верхней панели, содержащей день недели, название группы и тип недели. Через RxSwift кнопка типа недели привязана к открытию/закрытию таблицы UITableView со статическими данными (Observable.just). Дни и числа недели реализованы при помощи UIStackView. Расписание пар и экзаменов представляет собой UIScrollView, содержащий карточки UIView. Жесты для перехода между днями недели реализованы при помощи RxGesture.

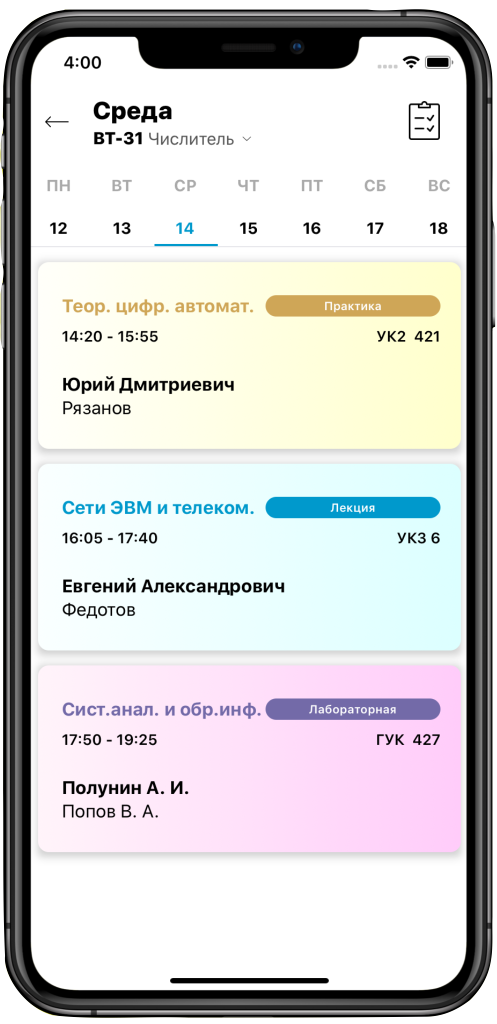
****

Рис. 12 – Расписание группы

## 3.3. Программирование раздела «Навигация» и интеграция с базой данных

Для реализации навигации, необходимо хранить информацию о помещениях в БД. В прошлых главах было рассказано, какова схема БД. Заполнить БД SQLite можно при помощи Python и библиотеки peewee. Тогда, осуществим подключение к БД:

A screenshot of a computer

Description automatically generated with low confidence

Общая структура выглядит следующим образом:

Graphical user interface

Description automatically generated with low confidence

Рис. 13 – Структура модуля для управления БД

Как было сказано ранее, подсчет матрицы кратчайших расстояний и кратчайших путей вычисляется заранее и хранится в БД. Поэтому, необходимо в скрипте формирующем строку базы для карты (Map) выполнить алгоритм Флойда – Уоршелла. Тогда, метод добавления карты в БД выглядит следующим образом:

Graphical user interface, text

Description automatically generated

А запись базы выглядит следующим образом:



Рис. 14 – Пример записи карты в БД

Базу данных необходимо скопировать в приложение, а затем подключиться к ней при помощи SQLite.Swift:

Text

Description automatically generated

В реализации MVVM для раздела – таблицы, соответствующие таковым из базы данных, являются Model. Покажем структуру.

Graphical user interface

Description automatically generated with medium confidence

Рис. 15 – Таблицы Model

Также реализованы TableMappers для совершения запросов в БД, используя таблицы из Model.

Text

Description automatically generated with medium confidence

Рис. 16 – Классы TableMappers

При парсинге матриц, считанных из БД в приложении, используются регулярные выражения, которые поддерживаются в Swift, посредством NSRegularExpression и описываются, как и вся обработка данных, во ViewModel.

Центральное управление графическим интерфейсом осуществляется из NavigationViewController. Он в себе содержит: верхнюю панель: TopBarNavigation, нижнюю панель: BottomBarNavigation, карту: Map, окно для поиска помещений SearchPremise и окно для выбора корпуса UniversityCorps. Подробнее рассмотрим каждый из компонентов.



Рис. 17 – Стартовое окно раздела «Навигация»

Для обеспечения общения между компонентами можно использовать встроенный механизм NotificationCenter, который в NavigationViewController принимает вызовы и выполняет дальнейшие действия. Например, перенаправляет информацию одному из дочерних View (которые описаны выше).

Верхняя панель содержит ячейки пунктов отправления и прибытия. Ее контроллер, TopBarNavigation, обрабатывает через подписки RxSwift нажатия и обновления текста в ячейках поиска пунктов, обновляя информацию в таблице поиска (BehaviorRelay) и отправляя информацию в главный NavigationViewController с idPremise выбранного пункта. Открытие экрана выбора корпусов может быть осуществлено по кнопке «More» на верхней панели. Выход из нее, по нажатии на экран, либо выбор корпуса.

Рис. 18 – Поиск аудитории Рис. 19 – Выбор корпуса

Карта представляет собой модифицированный UIScrollView, внутри которого содержатся маркеры Marker, а также StoreySwitcherView, управляющий перемещениями между этажами корпуса.

Внутри ScrollView загружается карта в формате png. Программно, максимальное масштабирование карты установим на уровне 1.0 от натурального размера и минимальное на уровне 0.3 (начальный показатель). При изменении масштабов карты, масштаб маркеров на ней тоже должен измениться. Тогда, по вызову метода UIScrollView – didZoom, необходимо отмасштабировать маркеры:



Благодаря методу LayoutSubviews, который вызывается каждый раз при действиях с картой UIScrollView (например, изменение размеров карты), изображение карты центрируется (при уменьшении карты) [17] и происходит добавление / удаление маркеров с экрана, в соответствии с текущим состоянием перекрытий (описано в 2.2. Принцип функционирования модулей).

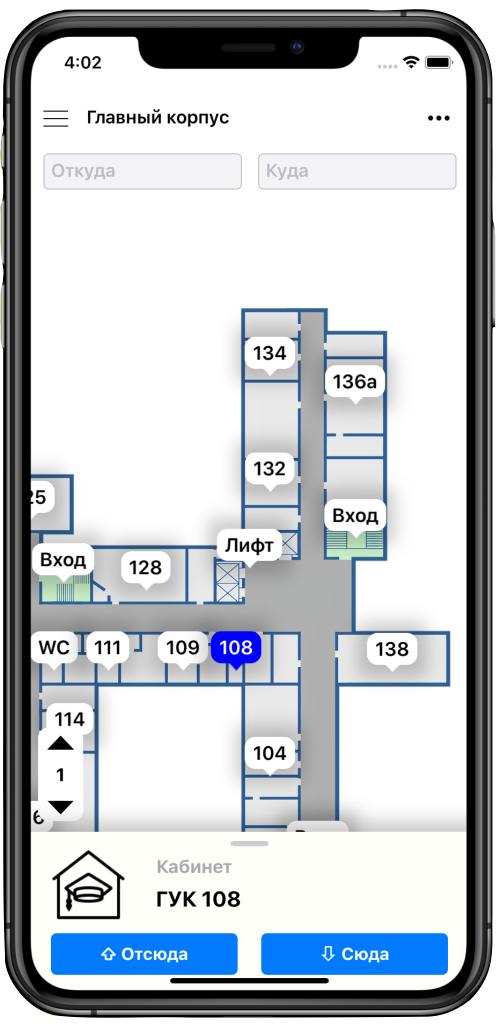
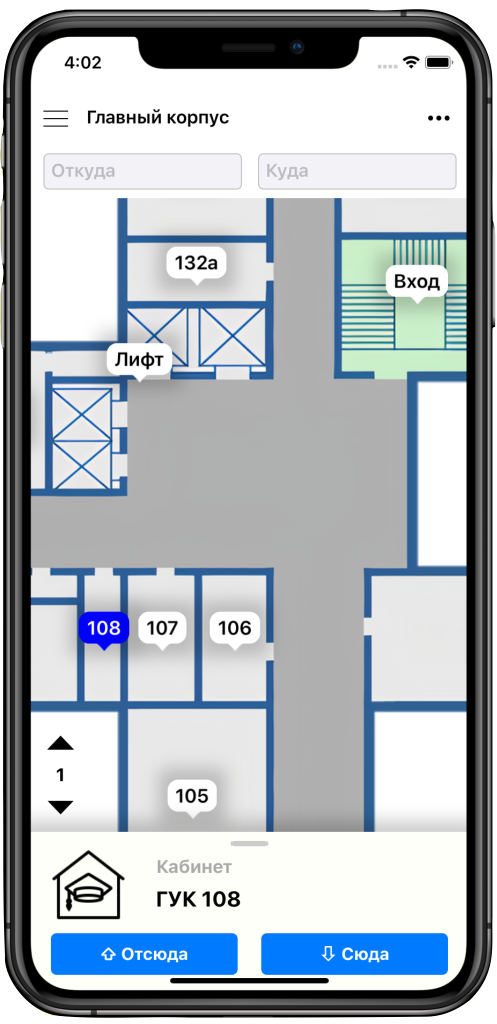
 

Рис. 20 – Масштаб карты - 0.3 Рис. 21 – Масштаб карты – 0.7

BottomBarNavigation позволяет выбрать помещение в качестве пункта отправления или прибытия, для чего реализованы соответствующие кнопки и реакции на которые реализованы в качестве замыканий RxSwift. После выбора помещения, его idPremisе сохраняется в NavigationViewController для реализации последующего проложения маршрута. Реализация интерактивности движения пальцем нижнего бара BottomBarNavigation реализуется при помощи UIPropertyViewAnimator, который позволяет ослеживать состояние прогресса анимации в зависимости от начальной, конечной и текущей точки графического элемента и продолжать выполнение анимации либо в начало, либо в конец. Приведем создания такого объекта и совершение анимации:

Graphical user interface, text, application, email

Description automatically generated

При построении маршрута, для обоих пунктов через idPremise осуществляется поиск необходимого помещения, анализируется поле idMap, затем у Map анализируется поле storey. На основе этой информации, принимается решение, являются ли пункты принадлежащими одному этажу. В случае несовпадения, в необходимо строить маршрут по 2 этажам через входы лестниц. Если лестниц несколько, то выбрать путь по кратчайшему расстоянию между лестницами.

Рисование пути осуществляется при помощи помещения пути Безье (состоящего из линейных кривых Безье B(t) = (1 - t)\*P0 + t\*P1, t = [0, 1]) на холст CAShapeLayer, анимация которого происходит при помощи CABasicAnimation, описанного в Core Animation, входящего в состав Foundation.

Text

Description automatically generated



Рис. 22 – Построение пути на одном этаже

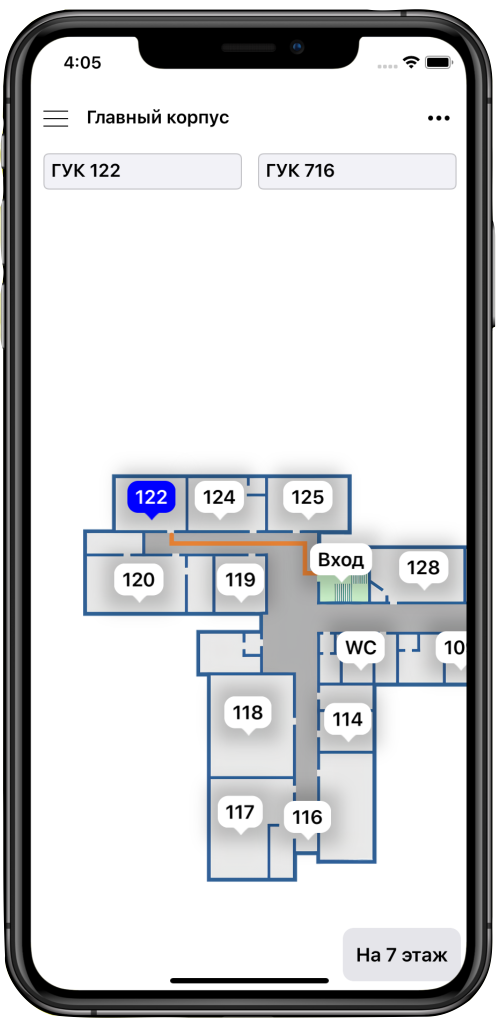
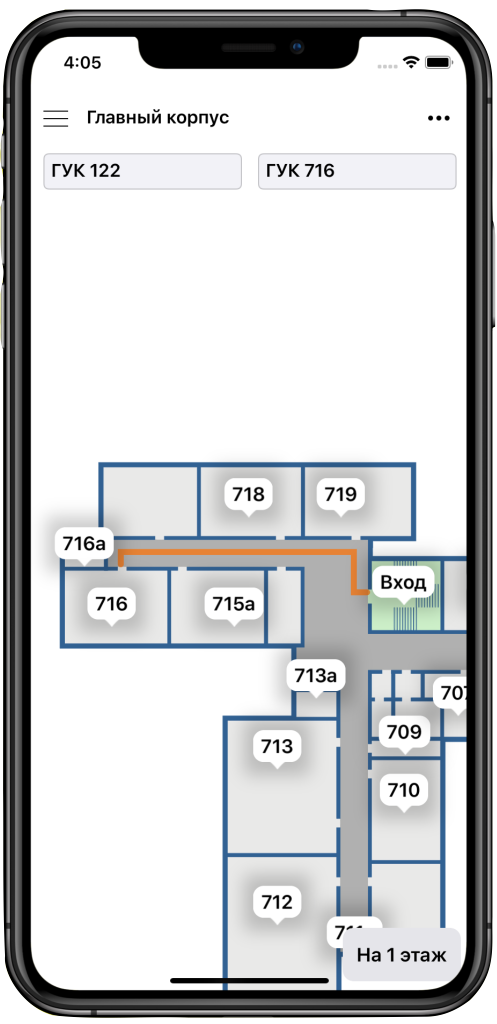
 

Рис. 23, 24 – Путь на нескольких этажах

## 3.4. Реализация расписания занятий для преподавателей

Для экрана выбора первой буквы фамилии используется UICollectionView с ячейками, содержащими первую букву.

Контент таблицы с преподавателями при поиске постоянно обновляется, значит необходимо использовать BehaviorRelay и принимать через Accept необходимые данные.

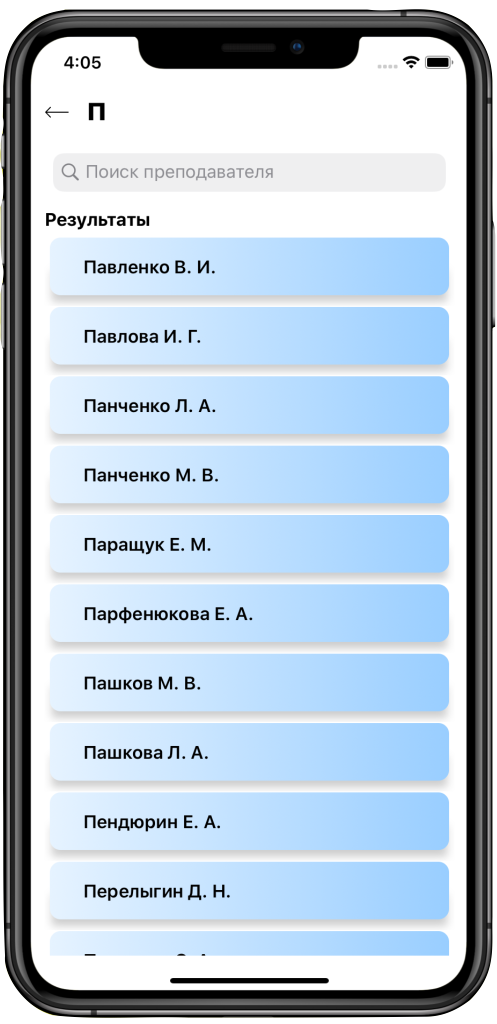
 

Рис. 25 – Первые буквы фамилий Рис. 26 – Список преподавателей

Реализация экрана с расписанием не отличается от таковой для студенческого, за исключением того, что вместо UILabel c группой, отображены ФИО преподавателя. Веб запрос, по-прежнему, совершается при помощи Alamofire, а HTML ответ анализируется при помощи SwiftSoup.

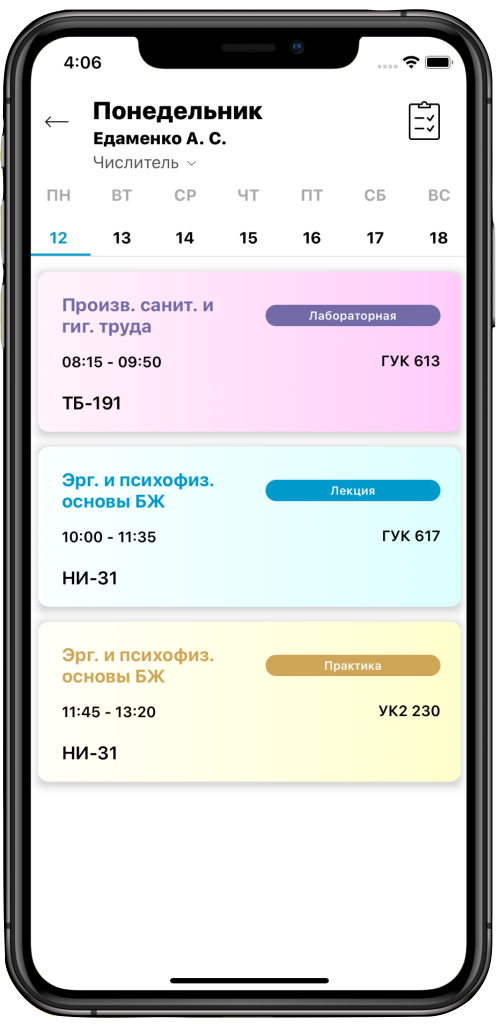
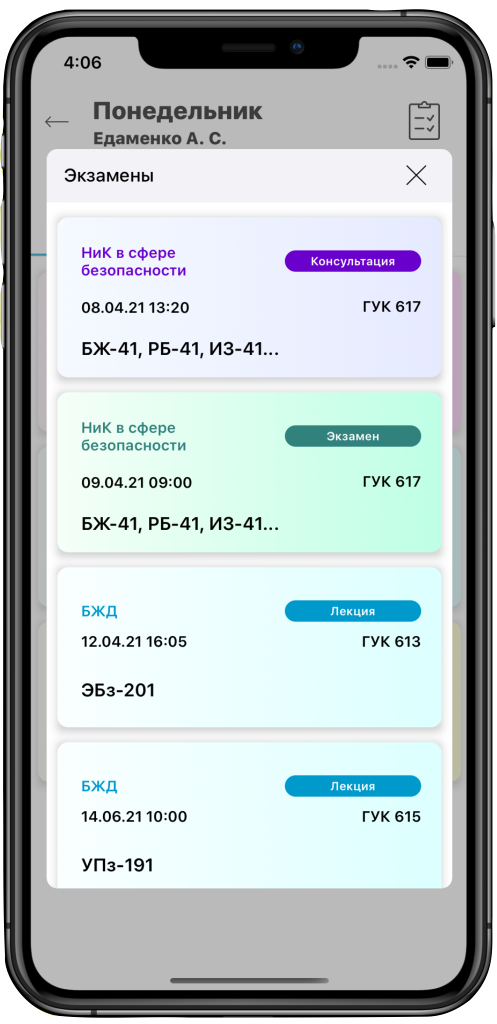
** **

Рис. 27 – Расписание занятий Рис. 28 – Расписание экзаменов

## 3.5. Создание раздела «Пункты питания»

ViewController состоит из выпадающего списка с заведениями, и таблицы с блюдами. Список заведений представляет собой UITableView, данные в котором не перегружаемы, а значит, могут быть в терминологии Rx «привязаны» к таблице, являясь Observavle.just. Через viewModel при помощи Alamofire совершается веб-запрос, из HTML получаем данные о заведениях. Для данных заведений, загружается списки меню. Меню помещается в таблицу с блюдами. Соответственно, данная таблица должна быть обновляемой, а значит данные должны быть BehaviorRelay и при выборе нового пункта получать новые данные через BehaviorRelay.accept(…).

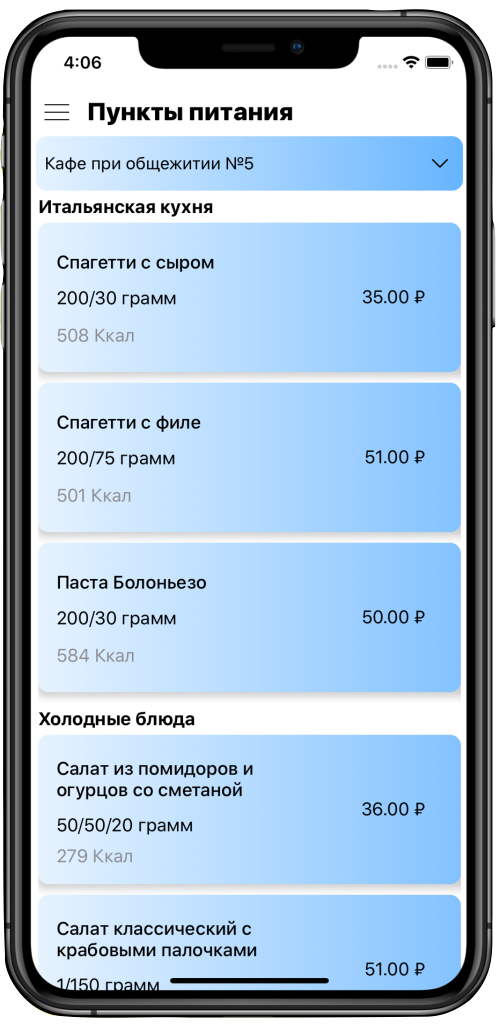


Рис. 29 – Пункты питания

## 3.6. Реализация раздела «Личный кабинет»

Замыкание, реагирующее на нажатие кнопки входа в систему, отправляет через Alamofire запрос со считанными личными данными с целью создать сессию доступа к сервисам личного кабинета. Анализируя ответ, определяем успешность входа в систему.

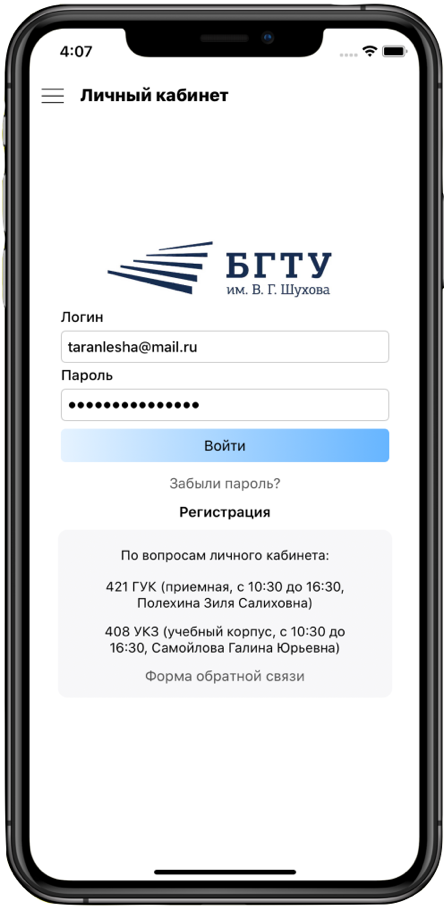
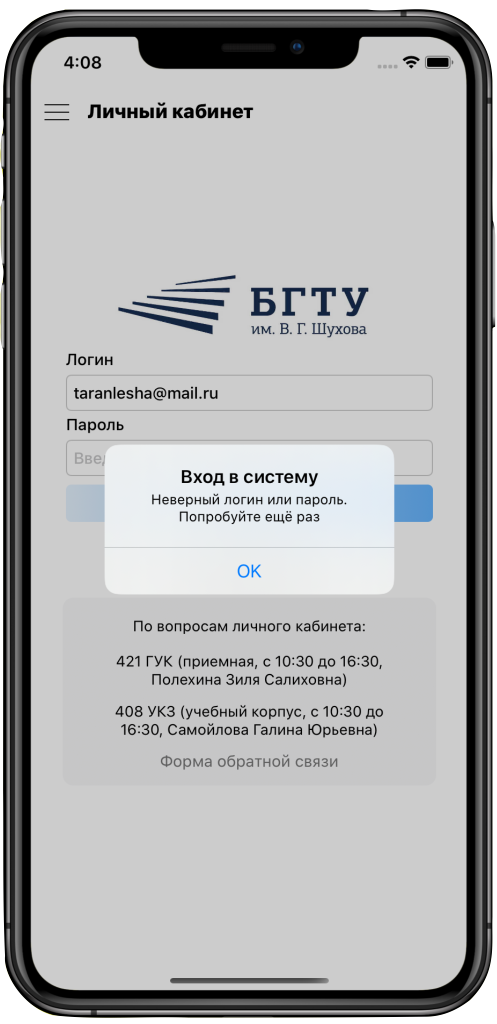
 

Рис. 30 – Вход в ЛК Рис. 31 – Неудачная попытка входа

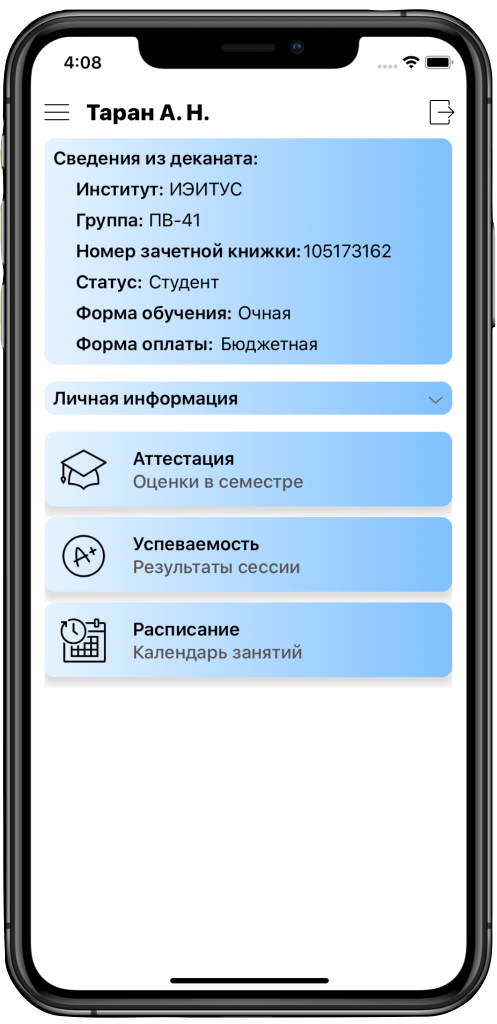
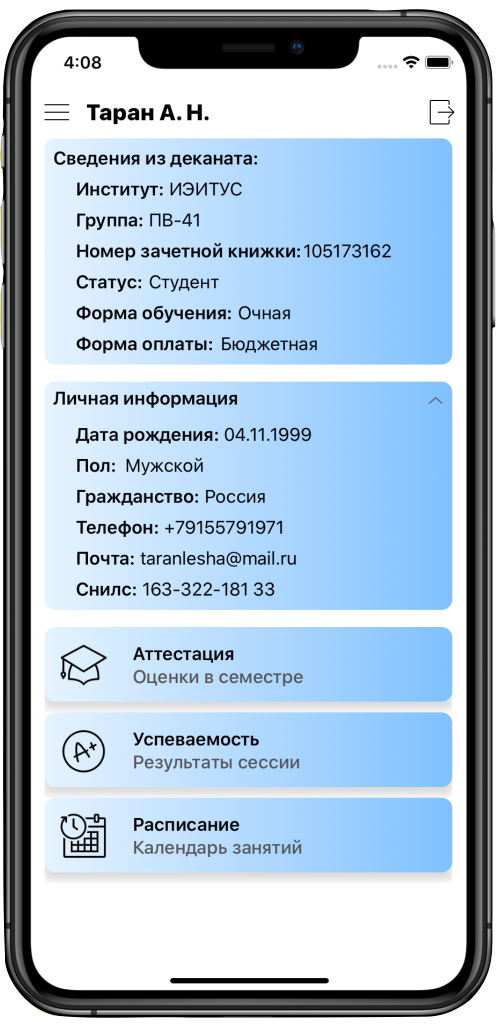
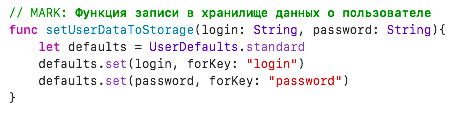
 

Рис. 32 – Главная страница ЛК Рис. 33 – Личная информация

Хранить данные в приложении можно при помощи встроенного механизма в Foundation: UserDefaults, а затем использовать их для повторного входа.



Удаление данных из UserDefaults (при выходе из системы, обнаружении ошибочных данных при повторном входе)

Text

Description automatically generated

Для реализации страниц аттестаций и успеваемости необходимо реализовать так сконфигурировать таблицу, чтобы ее ячейки могли обладать различным размером в высоту. Как было сказано ранее, многие из методов делегата UITableView имеют реализацию в RxSwift, однако, метода для установки размеров ячейки таблицы нет. Тогда, необходимо указать, что помимо Rx методов UITableView может принимать методы стандартного делегата таблицы:

Text

Description automatically generated with low confidence

Graphical user interface, text

Description automatically generated

После приема данных через Alamofire и анализа JSON, «привязываем» таблицу к данным. Ее данные не будут перегружены, значит для обертки данных достаточно Observable.Just.

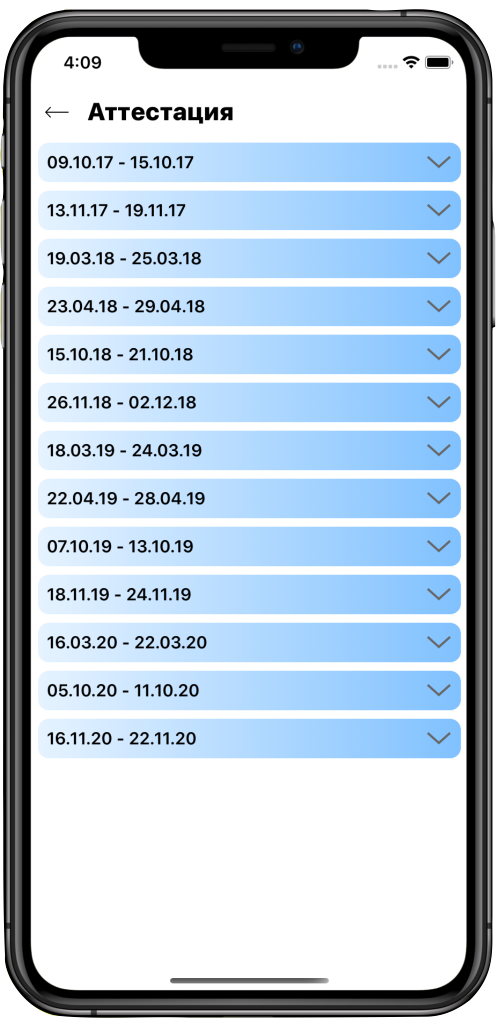
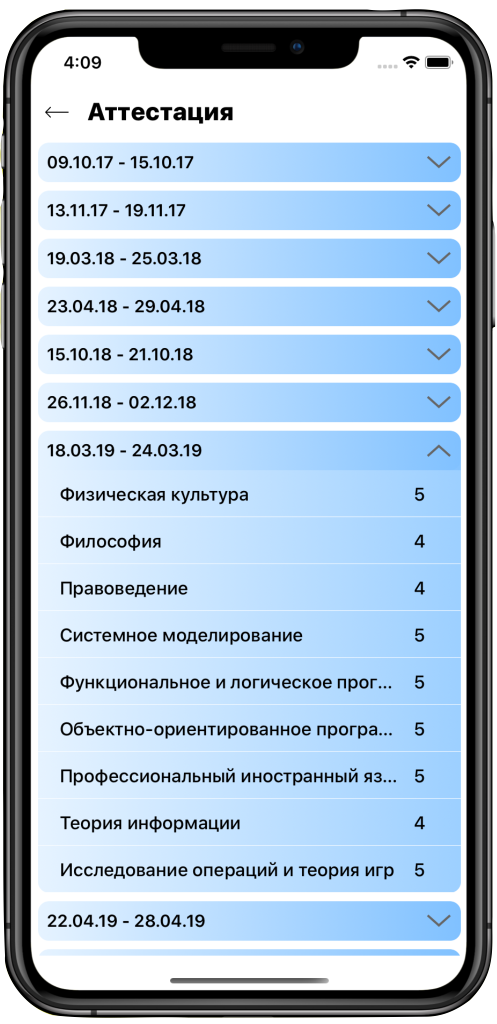
 

Рис. 34, 35 – Аттестация

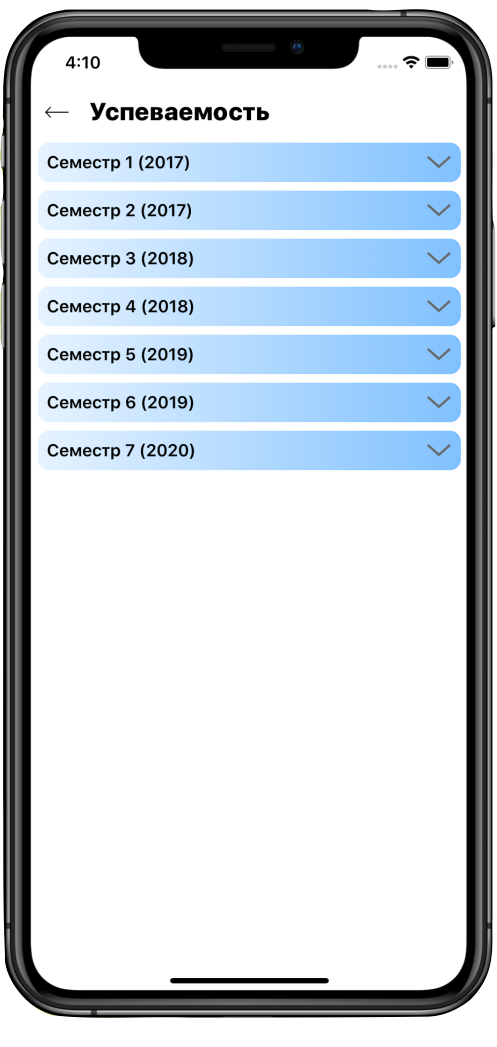
 

Рис. 36, 37 – Успеваемость

Расписание конструируется, исходя из названия группы.

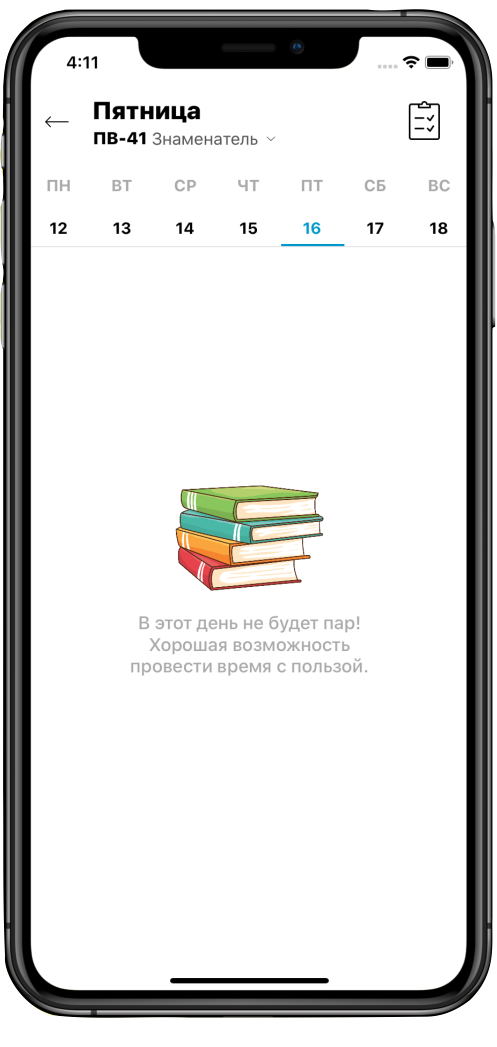


Рис. 38 – Расписание группы

## 3.7. Тестирование

Для тестирования при помощи XCTest [19], необходимо создать в XCode новую схему («BSTU App Tests»), где внутри файлов, в которых будут содержаться тесты, необходимо вместе с описанием импорта библиотек указать @testable import BSTU\_App, что позволит использовать классы, разработанные для приложения для тестирования.

Класс, содержащий тесты должен быть наследником XCTestCase и иметь префикс test.

Для веб – запросов есть возможность запустить тестовый Mock - сервер Hippolyte [20], который будет выдавать в зависимости от полученных запросов необходимые ответы.

При архитектуре MVVM, основная логика по обработке данных осуществляется во ViewModel, которая является самостоятельной независимой частью, а значит, может быть достаточно просто протестирована в изоляции.

Проведенные тесты для каждого из разделов приведены на изображении ниже:

Table

Description automatically generated

Рис. 39 – Результаты тестирования в Xcode

**Выводы по главе:** были описаны технологии, использованные при разработке приложения, рассмотрена разработка разделов c точки зрения программирования, а также продемонстрированы скриншоты приложения. Произведено тестирование разработанных разделов.

# Заключение

По итогам данной дипломной работы было разработано и протестировано мобильное приложение на платформе iOS для БГТУ им. Шухова. Разработан и реализован графический интерфейс для расписаний студентов и преподавателей, подключен существующий API. Создана навигация для корпусов университета. Разработана база данных, содержащая информацию о помещениях, этажах, корпусах. Создан интерфейса для отображения меню пунктов питания, подключение API. Реализован личный кабинет в мобильном приложении. Через сессии Alamofire реализована авторизация в приложении. Получение через API информации о сотруднике. Для студента – создание окон аттестации и успеваемости, где есть информация о текущих оценках и о оценках сессии, полученных за время обучения в ВУЗе.

Варианты развития приложения:

• Для раздела «Навигация» – реализация определения местоположения в помещении

• Создание разнообразных уведомлений (Today View, «Напоминания») о предстоящих занятиях для авторизированных пользователей

• Реализация раздела «Абитуриенту» для анализа текущей конкурсной ситуации и поиска абитуриента в списках

• Оплата услуг университета в приложении (Apple Pay).

• Реализация эмуляции университетского пропуска MiFare в телефоне.

# Список использованных источников

1) Статистика пользователей смартфонов в России. [Электронный ресурс]// URL: <https://www.gazeta.ru/tech/news/2019/12/23/n_13848662.shtml> (Дата обращения: 11.02.2021, режим доступа: свободный).

2) Информационные технологии в образовании и науке [Электронный ресурс]// URL: <https://www.top-technologies.ru/ru/article/view?id=25948> (Дата обращения: 16.02.2021, режим доступа: свободный).

3) Информационные технологии в образовании [Электронный ресурс]// URL: <https://nvsu.ru/ru/Intellekt/1135/Pashchenko%20O.I.%20Informatsionnie%20tehnologii%20v%20obrazovanii%20-%20Uch-met%20posobie%20-%202013.pdf> (Дата обращения: 28.02.2021, режим доступа: свободный).

4) Эволюция мобильных ОС [Электронный ресурс]// URL: <https://www.if24.ru/evolyutsiya-mobilnyh-os/> (Дата обращения: 21.02.2021, режим доступа: свободный).

5) Как устроена ОС [Электронный ресурс]// URL: <https://xakep.ru/2014/10/08/kau-ustroena-ios/> (Дата обращения: 02.03.2021, режим доступа: свободный).

6) Информация о конкурсных списках 2021 [Электронный ресурс]// URL: <http://adm-krom.ru/news/s-2021-goda-v-konkursnykh-spiskakh-vuzov-bolshe-ne-budet-imyon> (Дата обращения: 06.03.2021, режим доступа: свободный).

7) Процесс загрузки приложения в AppStore [Электронный ресурс]// URL: <https://livetyping.com/ru/blog/kak-vylozhit-prilozhenie-v-app-store> (Дата обращения: 09.03.2021, режим доступа: свободный).

8) Диаграммы вариантов использования [Электронный ресурс]// URL: <http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4> (Дата обращения: 12.03.2021, режим доступа: свободный).

9) Архитектурные паттерны в iOS [Электронный ресурс]// URL: <https://habr.com/ru/company/badoo/blog/281162/> (Дата обращения: 16.03.2021, режим доступа: свободный).

10) Cocoa MVC [Электронный ресурс]// URL: [h](https://habr.com/ru/company/badoo/blog/281162/)[[https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html](https://habr.com/ru/company/badoo/blog/281162/)](https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html) (Дата обращения: 18.03.2021, режим доступа: свободный).

11) RxSwift Часть 1 [Электронный ресурс]// URL: <https://habr.com/ru/post/423603/> (Дата обращения: 22.03.2021, режим доступа: свободный).

12) ReactiveX/RxSwift [Электронный ресурс]// URL: <https://github.com/ReactiveX/RxSwift> (Дата обращения: 24.03.2021, режим доступа: свободный).

13) SwiftyJSON [Электронный ресурс]// URL: <https://github.com/SwiftyJSON/SwiftyJSON> (Дата обращения: 25.03.2021, режим доступа: свободный).

14) SwiftSoup [Электронный ресурс]// URL: <https://github.com/scinfu/SwiftSoup> (Дата обращения: 26.03.2021, режим доступа: свободный).

15) SQLite Swift [Электронный ресурс]// URL: https://github.com/stephencelis/SQLite.swift (Дата обращения: 28.03.2021, режим доступа: свободный).

16) MVP/MVC/MVVM [Электронный ресурс]// URL: <https://www.pvsm.ru/ios/117313> (Дата обращения: 01.04.2021, режим доступа: свободный).

17) Модификации UIScrollView [Электронный ресурс]// URL: <https://github.com/IvanAkulov/iOS-Demos/blob/master/100.%20iOS%2013%20Xcode%2011%20ScrollingApp-master.zip> (Дата обращения: 04.04.2021, режим доступа: свободный).

18) Screenshots for AppStore [Электронный ресурс]// URL: <https://appscreens.com/user/screen/cqtc3w9on4x5dukFHlUw> (Дата обращения: 08.04.2021, режим доступа: свободный).

19) XCTest [Электронный ресурс]// URL: <https://developer.apple.com/documentation/xctest> (Дата обращения: 16.04.2021, режим доступа: свободный).

20) Mock – сервер Hippolyte [Электронный ресурс]// URL: <https://github.com/JanGorman/Hippolyte> (Дата обращения: 13.04.2021, режим доступа: свободный).