МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Рубцовский институт (филиал) федерального государственного бюджетного образовательного учреждения высшего образования
«Алтайский государственный университет»

Кафедра Математики и прикладной информатики

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (бакалаврская работа)

Тема: Разработка мобильного приложения для организации доступа к ресурсам информационно-образовательного портала (на примере Рубцовского института (филиала) АлтГУ)

Выпускную квалификационную работу (бакалаврскую работу) выполнил студент
3 курса, группы 1255У
Шалда С.В.
_____
(подпись)

Научный руководитель:
канд. техн. наук, доцент
Анисимова Е.А.
_____
(подпись)

Допустить к защите
Зав. кафедрой
канд. техн. наук, доцент
Жданова Е.А.
_____
(подпись)

«18» июня 2018 г.

Выпускная квалификационная работа (бакалаврская работа) защищена

«26» июня 2018 г.
Оценка отлично

Председатель ГЭК
д-р техн. наук, профессор
Пятковский О.И.
_____
(подпись)

Рубцовск 2018

РЕФЕРАТ

Дипломная работа: 105 страниц, 39 рисунков, 4 таблицы, 25 источников, 1 приложение.

СИСТЕМНЫЙ АНАЛИЗ, ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНЫЙ ПОРТАЛ, МОБИЛЬНЫЙ ИНТЕРНЕТ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПЛАТФОРМА ANDROID, ПЛАТФОРМА XAMARIN, ЛИЧНЫЙ КАБИНЕТ, РАСПИСАНИЕ, НОВОСТОЙ КОНТЕНТ.

Цель работы – разработка мобильного приложения для организации доступа к ресурсам информационно-образовательного портала Рубцовского института (филиала) Алтайского государственного университета (для платформы Android).

Объект исследования – информационно-образовательный портал Рубцовского института (филиала) АлтГУ.

Предмет исследования – организация доступа мобильных устройств к разделам информационно-образовательного портала.

Методы решения поставленных задач: системный анализ, функционально-ориентированная методология описания систем, оригинальное проектирование, CASE и RAD технологии.

Результатом работы является создание мобильного приложения для платформы Android, которое позволяет пользователям мобильных устройств получать доступ к актуальной информации информационно-образовательного портала Рубцовского института (филиала) АлтГУ.

Практическая значимость работы заключается в разработке нового прикладного программного обеспечения для удовлетворения информационных потребностей пользователей. Разработанное приложение рекомендовано для внедрения через свободное распространение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время неуклонно растет доля пользователей сети Интернет, которые для выхода в сеть используют мобильные устройства, такие как смартфоны, планшеты и коммуникаторы. Современные мобильные устройства используют различные операционные системы и по производительности и функциональности почти не уступают персональным компьютерам. Самыми распространёнными операционными системами на мобильных устройствах являются iOS, Android и Windows Phone.

С развитием данного сегмента доступа в сеть Интернет изменился и подход к получению необходимой информации. Так, в последнее время одним из самых распространенных подходов является получение информации при помощи мобильных приложений. Основным отличительным преимуществом этого подхода является возможность его настройки под нужды пользователей.

По данным исследовательской компании eMarketer с каждым годом пользователи всё больше используют мобильные приложения и меньше – мобильные браузеры и персональные компьютеры. В настоящее время уже у многих учебных заведений существует мобильное приложение, которое позволяет обеспечить удобный доступ к необходимой информации.

В Рубцовском институте (филиале) федерального государственного образовательного учреждения высшего образования «Алтайский государственный университет» (далее – Рубцовский институт (филиал) АлтГУ или институт) функционирует развитый информационно-образовательный портал, который обеспечивает доступ пользователей к информационным ресурсам в ключевых областях, таких как образование, наука, внеучебная деятельность, регламентирующие документы, библиотечные ресурсы, новостной контент и другие. Использование мобильного приложения для получения информации из разделов портала позволит, например, обеспечить пользователей постоянным и актуальным

расписанием занятий, сведениями об успеваемости студентов, возможностью оперативного заказа различного рода справок и другое.

Основными достоинствами мобильных приложений, по сравнению с мобильными сайтами, являются:

– повышенная степень взаимодействия;

– интуитивный и понятный пользовательский интерфейс;

– работа в автономном режиме – без активного подключения к сети интернет;

– уменьшение количества расходуемого мобильного трафика;

– автоматизация требуемых действий.

На текущий момент в Рубцовском институте (филиале) АлтГУ имеется мобильное приложение для информирования студентов о текущих занятиях, но в этом приложении имеется существенный ряд недостатков, таких как:

– некорректная работа на новых версиях операционной системы Android (выше версии 6.0);

– невозможность узнать сведения о текущей успеваемости, не заходя на информационно-образовательный портал института;

– невозможность оперативного заказа требуемых справок;

– отсутствие новостной ленты Рубцовского института (филиала) АлтГУ.

Наличие этих недостатков в имеющемся мобильном приложении на платформе Android делает выбранную тему актуальной.

Новизна работы заключается в применении платформы разработки, на которой будет разработано мобильное приложение – Xamarin. Эта платформа предназначена для создания нативных приложений для iOS, Android и Windows из общего кода C# или .NET, которая позволяет многократно использовать между платформами почти 95% кода. Данная платформа позволит в дальнейшем продолжить разработку и развитие приложения для всех существующих платформ.

Таким образом, объектом исследования данного проекта является информационно-образовательный портал Рубцовского института (филиала) АлтГУ.

Предметом исследования является организация доступа мобильных устройств к разделам информационно-образовательного портала, таким как:

- расписание занятий;
- личный кабинет;
- новостной контент.

Целью работы является разработка взаимодействующего с API мобильного приложения, предназначенного для работы на смартфонах и планшетах под управлением операционной системы Android, которое позволит обеспечить пользователям доступ к ресурсам информационно-образовательного портала Рубцовского института (филиала) Алтайского государственного университета. Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области;
- выявить недостатки существующей системы предоставления доступа пользователей к информации;
- описать проектные решения по функциональному обеспечению разрабатываемого программного приложения;
- выработать проектные решения по обеспечивающим подсистемам разрабатываемого приложения;
- разработать функциональное мобильное приложение;
- оценить эффективность от внедрения мобильного приложения.

Исходными данными для выполнения работы является учебная и научная литература, а также интернет-источники по проектированию информационных систем, локальная нормативно-справочная документация Рубцовского института (филиала) АлтГУ.

Для выполнения работы использовались следующие методы и средства: технико-экономический анализ предметной области, структурно-

функциональное описание систем с использованием CASE-средств (CA BPwin), оригинальное проектирование, прототипирование и быстрая разработка приложений с помощью RAD-средств (Xamarin).

1 Аналитическая часть

1.1 Технико-экономическая характеристика Рубцовского института (филиала) АлтГУ

Рубцовский институт (филиал) АлтГУ – обособленное структурное подразделение Алтайского государственного университета, расположенное в городе Рубцовске Алтайского края и осуществляющее постоянно его функции в рамках выданной институту лицензии на образовательную деятельность.

Институт является государственной, некоммерческой организацией, имеет право на ведение образовательной деятельности и на льготы, установленные законодательством РФ, в соответствии с лицензией, выданной Министерством образования РФ.

Институт реализует образовательные программы среднего профессионального и высшего (бакалавриат) образования. Также институт обладает правом реализации образовательных программ по хозрасчетным договорам в рамках установленного перечня специальностей и направлений.

Институт был создан для удовлетворения потребности в высококвалифицированных специалистах и для повышения уровня образования в регионе. В соответствии с заданными целями, предприятие вправе осуществлять следующие виды деятельности:

- предоставление услуг высшего образования;
- проведение курсов повышения квалификации;
- оказание платных библиотечных услуг;
- услуги передачи данных, информационные услуги, в том числе на основе организуемых компьютерных классов открытого доступа;
- услуги по разработке, производству и поставке программных продуктов, информационных систем, программных и методических средств;

– сервисных работ и услуг по техническому обслуживанию средств вычислительной техники, оргтехники;

– организация и проведение физкультурно-оздоровительных и спортивных мероприятий, создание спортивных секций;

– организация и предоставление культурно-бытовых услуг.

Финансирование деятельности института осуществляется в установленном порядке за счет средств:

– федерального бюджета;

– полученных от выполнения хозяйственных договоров;

– полученных от подготовки студентов по хозрасчетным договорам;

– спонсоров, добровольных взносов, пожертвований граждан, организаций;

– из других законных источников.

Организационная структура Рубцовского института (филиала) АлтГУ представляет собой строгую иерархию, которая представлена на рисунке 1.1.

Институт располагается в двух учебных корпусах. Численный состав сотрудников филиала составляет 105 человек. Директору института непосредственно подчиняются:

– заместитель по учебной работе;

– заместитель по научной работе;

– отдел по связям с общественностью;

– юридический отдел;

– отдел по работе с персоналом;

– финансово-экономический отдел;

– отдел технического и программного обеспечения;

– ведущий документовед;

– ведущий инженер;

– комендант;

– магазин.

Рисунок 1.1 – Структура управления Рубцовского института (филиала) АлтГУ

Полномочия для управления конкретными сферами деятельности рассредоточиваются между заместителями директора. В целом рационального отношения организационной структуры и информационного аспекта управления трудно достичь без оптимального распределения системы управленческого аппарата, которое наглядно представлено в АлтГУ.

Параметры основных структурных элементов объекта

Каждый отдел и структурное подразделение в институте имеет свою функциональную особенность.

Отдел по работе с персоналом – проводит работу с персоналом, заключает договора, поддерживает социальную обеспеченность работника со стороны работодателя, численность отдела составляет 2 человека, на 15 мая 2018 года.

Отдел программного и технического обеспечения – обеспечивает

работоспособность компьютерных сетей в Рубцовском институте, ремонт и наладку оборудования, создание и освоение современного программного обеспечения, модификацию и поддержку информационно-образовательного портала. Численность отдела составляет 9 человек, на 15 мая 2018 года.

Отдел по работе со студентами – осуществляет работу со студентами, формирование групп, составление приказов, составление отчётов успеваемости. Численность отдела составляет 5 человек, на 15 мая 2018 года.

Отдел по связям с общественностью – осуществляет приём абитуриентов и проведение вступительных экзаменов. Численность центра составляет 5 человек, на 15 мая 2018 года. Штатный состав – 2 человека на май 2018 года, на время работы приемной комиссии – 5 человек, по данным на август 2017 года.

Бюро расписаний – осуществляет составление план-графиков учебных процессов, расписания и распределение нагрузок среди преподавателей. Численность отдела составляет 2 человека, на 15 мая 2018 года.

Библиотека – осуществляет выдачу учебной литературы студентам и сотрудникам института, поддерживает библиотечный фонд. Численность отдела составляет 2 человека, на 15 мая 2018 года.

Финансово-экономический отдел – осуществляет ведение бухгалтерского учёта и анализа предприятия и начисляет заработную плату сотрудникам. Численность отдела составляет 5 человек, на 15 мая 2018 года.

Кафедры – осуществляют организацию и проведение образовательного процесса согласно общегосударственным стандартам. Численность преподавательского состава составляет 40 человек, на 15 мая 2018 года.

Служба коменданта и служба ведущего инженера, численность составляет 21 человек, на 15 мая 2018 года.

В парке средств вычислительной техники Рубцовского института (филиала) АлтГУ имеются 193 единиц ноутбуков и персональных компьютеров. Из них:

– 15 персональных компьютеров с процессорами Intel Core i3;

– 15 персональных компьютеров с процессорами Intel Pentium G2010;

– 34 персональных компьютеров с процессорами Intel Pentium E5700;

– 32 персональных компьютера с процессорами Intel Pentium G630;

– 18 персональных компьютеров с процессорами Intel Celeron D 315;

– 18 персональных компьютеров с процессорами Intel Celeron 420;

– 16 ноутбуков с процессорами Intel Celeron T3100;

– 17 ноутбуков с процессорами Intel Pentium B970;

– 14 ноутбуков с процессорами Intel Duo Core T6570;

– 14 ноутбуков с процессорами Intel Celeron 550.

Совместно с данным оборудованием, в учебном и производственном процессах используется 13 мультимедийных проекторов, 43 единицы лазерных принтеров (3 цветных), 22 единицы планшетных сканеров, 14 единиц копировально-множительной техники.

Для осуществления образовательного процесса в институте имеется 11 компьютерных классов, из них один класс на 15 машин используются в режиме свободного доступа студентов.

Еще одна аудитория на 8 компьютеров используется для проведения лабораторных занятий по дисциплинам специальностей и направлений кафедры «Математики и прикладной информатики». Мобильные классы включают в себя 4 компьютерных кабинета с ноутбуками, и активно используются в учебно-образовательной деятельности, как для тематических занятий, так и для организации доступа к ресурсам корпоративной сети и Internet на территории института.

Все административные подразделения, кафедры, компьютерные классы и лаборатории объединены в единую локальную вычислительную сеть, построенную на основе традиционных структурированных кабельных

системах (СКС), суммарной протяженностью около 12 км, со скоростью передачи данных в сетях подразделений 1Гбит/с.

Кроме этого, имеется 2 сегмента магистральных волоконно-оптических каналов связи, объединяющих между собой оба корпуса института в пределах кампуса. Общая протяженность данной линии 1,58 км с пропускной способностью кабельной системы 1Гбит/с и возможностью расширения до 10 Гбит/с, за счет модернизации активного сетевого оборудования. Один оптический сегмент используется как опорный канал для доступа пользователей к ресурсам локальной сети института и внешним сетям, а другой – для организации внутренней цифровой IP-телефонии на основе голосовых шлюзов VoIP.

Необходимо отметить, что доступ к ресурсам ЛВС на территории Рубцовского института (филиала) АлтГУ может осуществляться как по традиционным проводным технологиям, так и через беспроводные системы Wireless WiFi (до 300 Мбит/с).

В настоящее время в институте существует следующая конфигурация серверов:

1. Файловый сервер RFAGU. Выполняет функции главного контроллера домена RBDOM, обеспечивающего централизацию управления ЛВС института, файлового сервера личных и общедоступных ресурсов, СПС и Консультант +. Технические характеристики: 2xP4Xeon 2,8GHz, 4x2Gb, IDE-SATA 500Gb, 1Tb, RAID5 SATA 3x1Tb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

2. Сервер WWW. Выполняет функции базового WEB-сервера со следующим набором сервисов: HTTP, FTP, DNS, MAIL, PROXY, APACHE, MySQL. Технические характеристики: 2xP4Xeon, 3,0GHz, 4x2Gb, RAID1 SATA 2x500Gb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

3. Сервер GTW. Выполняет функции шлюза, обеспечивающего интеграцию сети института в глобальные сети Internet. Здесь установлен

корпоративный межсетевой экран. Технические характеристики: Pentium-IV 2,4GHz, IDE-SATA 80Gb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

4. Сервер EDUSERVER. Выполняет функции сервера БД для используемых в учебном процессе банковских информационных систем «RS-Bank 5.1». На нем установлен web-сервер Apache с модулем PHP 4.0, обеспечивающий онлайн доступ к расписанию занятий, система управления курсами Moodle, СКУД обеспечивающая автоматическое управления входом и выходом людей в здание организации, модуль FastReport, обеспечивающий формирование отчетов печати для приложений. Технические характеристики: Intel-I7 (4 core) 3,00GHz, 8Gb, IDE-SATA 500Gb, Gigabit Ethernet. Установленное ПО – MS Windows 2012 AS.

5. Сервер ORACLE. Выполняет функции сервера БД для хранения информации для портала и информации, необходимой для учебного процесса. Технические характеристики: 2xP4Xeon, 2,8GHz, 24Gb, RAID5 SATA 4x200Gb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

Характеристика информационных потоков института

Информация об абитуриенте попадает в базу данных по абитуриентам, это в первую очередь входная анкетная информация (фамилия, имя, отчество, адрес, условия поступления, специальность и т.д.), после успешной сдачи вступительных экзаменов, информация о поступивших студентах передаётся в базу данных по студентам. В ней отражается состояние успеваемости студента и академические долги.

Информация о количестве студентов в группах, количестве групп, попадает в учебно-методический сервер, для составления учебного расписания.

Готовая информация попадает на кафедры в виде составленных расписаний, как для студентов, так и для преподавателей.

База данных по студентам. Это программный продукт, разработанный сотрудниками института, выполненный на основе СУБД Oracle.

Входящей информацией для неё является информация о поступлении студентов, передаваемая из базы данных по абитуриентам. В базе по студентам ведётся:

- медицинский учёт, осуществляемый медиком;
- воинский учёт, осуществляемый отделом по работе с персоналом.

Отдел по работе со студентами вносит данные о студентах, группах, дисциплинах, успеваемости, академических задолженностях.

Выходной информацией считается:

- списки групп;
- списки задолжников;
- приказы об отчислении;
- приказы о переводе на другой курс;
- списки специальностей.

База по сотрудникам, также собственная разработка информационной системы ведения учёта работающего персонала. Входной информацией являются анкетные данные всех служащих.

Ведение бухгалтерского учёта, начисления зарплаты, производится в системе «1С: Предприятие», используя данные из базы по сотрудникам, что позволяет отказаться от двойного ввода информации. Также, «1С: Предприятие», используя данные из базы по студентам, производит учёт оплаты за обучение студентов.

Учебно-методический сервер, используя информацию из базы по студентам и данные об учебных планах и рабочих программах, составляет учебное расписание, графики учебных процессов.

В структурном подразделении финансово-экономический отдел существуют информационные потоки, такие как: информация о поступивших товарах, постановка их на учёт и счёт-фактуры. Выходной информацией являются акты о списании расходных материалов, накладные, отчёты о

задолженностях студентов по оплате. Отчёты в инспекцию по налогам и сборам, в пенсионный фонд, так же считаются выходной информацией.

Все информационные ресурсы, доступные для общего пользования, можно получить практически с любого компьютера в институте, благодаря использованию высокоскоростной локально-вычислительной сети.

Также в обоих корпусах института установлены для общего пользования отдельные компьютеры с сенсорным экраном, с помощью которых любой желающий может найти информацию о группах, студентах, их задолженностях, успеваемости, о преподавателях, а также узнать расписание.

Использование электронного библиотечного каталога в большой мере облегчает работу студентам с литературой.

Электронный библиотечный каталог позволяет найти нужную информацию о каком-либо издании, находящимся в библиотеке института. Библиографическая информация об имеющихся книгах и периодических изданиях заносится в базу данных работниками библиотеки.

Официальный информационно-образовательный портал Рубцовского института (филиала) АлтГУ (www.rb.asu.ru) призван решать следующие задачи:

– обеспечение построения единого информационного пространства института на основе интеграции информационных ресурсов и сервисов структурных подразделений Рубцовского института (филиала) АлтГУ;

– обеспечение повышения эффективности образовательной деятельности института за счет использования дистанционных форм обучения;

– способствовать обеспечению формирования целостного позитивного образа Рубцовского института (филиала) АлтГУ в стране и мире, представление информации о деятельности института, его учебном и научном потенциале;

– объективно и оперативно информировать российское и мировое сообщества о наиболее значимых событиях, происходящих в Рубцовском институте (филиале) АлтГУ;

– способствовать развитию научных и учебных связей с ВУЗами России, ближнего и дальнего зарубежья;

– обеспечивать доступ студентов, сотрудников и преподавателей к различным по представлению информационным ресурсам, нормативно-методическим материалам, системам промежуточного и выходного контроля знаний, а также системам управления качества образования, для повышения эффективности образовательной деятельности ВУЗа;

– оперативно предоставлять достоверную информацию абитуриентам института.

Интернет сервер института позволяет получить информацию о текущих новостях института, информацию о расписании учебного процесса, провести голосование, оставить сообщение на форуме.

Система тестирования позволяет провести дистанционное тестирование студентов по различным дисциплинам.

1.2 Анализ функционирования объекта исследования

В Рубцовском институте (филиале) АлтГУ функционирует информационно-образовательный портал, который предоставляет студентам и преподавателям необходимую информацию. На портале существует множество сервисов, которые обычным пользователем используются крайне редко. Для доступа к порталу используется его основная версия.

Главная функция портала делится на 7 основных подфункций:

– доступ к библиотеке;

– доступ к информации для абитуриентов;

– доступ к расписаниям;

– доступ к информации об образовании;

– доступ к новостной ленте;

– доступ к персональной странице;

– доступ к формируемым отчетам.

На рисунке 1.2 отображена контекстная диаграмма процесса «Организации доступа к ресурсам информационно-образовательного портала Рубцовского института (филиала) АлтГУ» в представлении «Как есть», выполненная в нотации IDEF0.

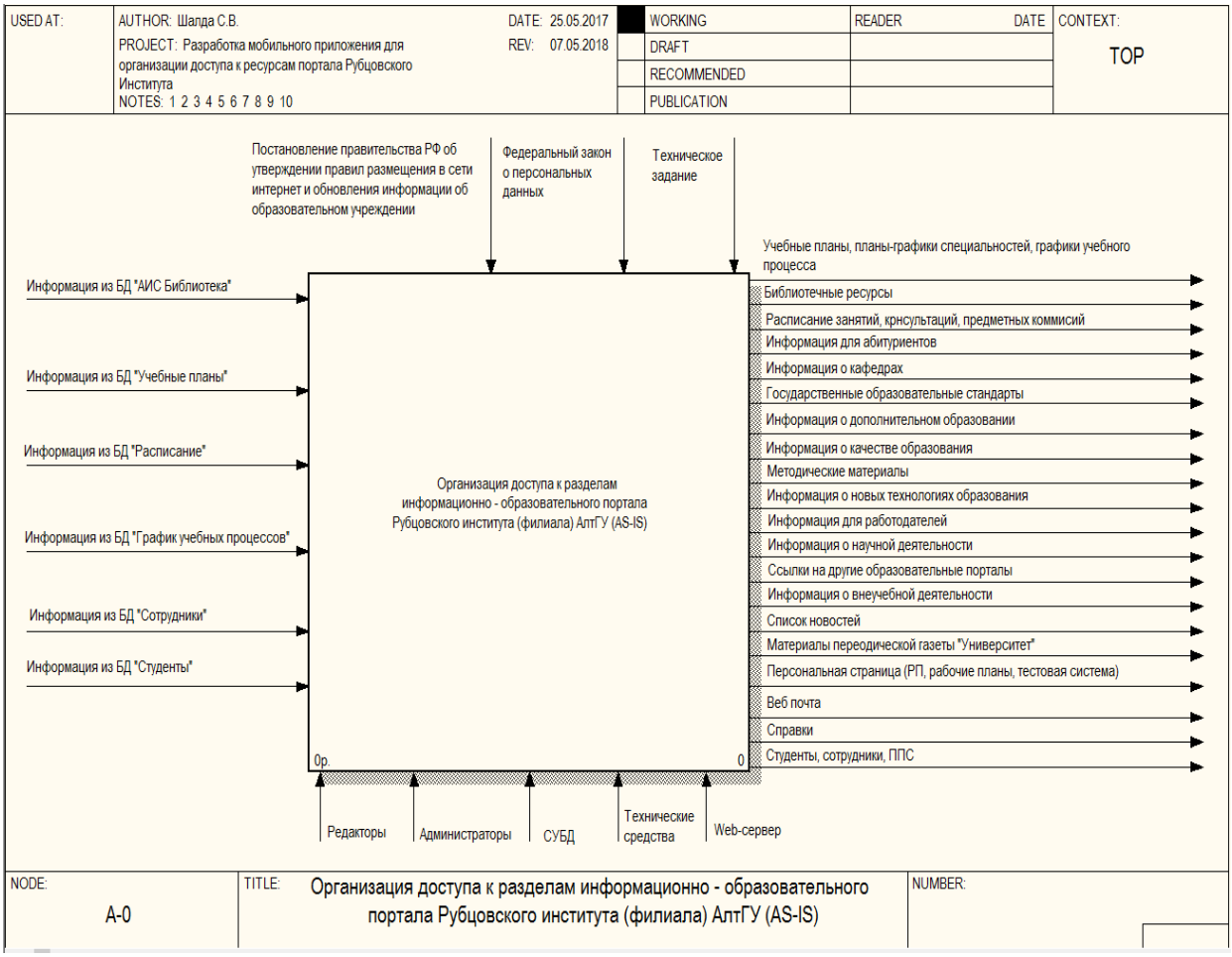

Рисунок 1.2 – Контекстная диаграмма процесса «Организации доступа к разделам информационно-образовательного портала Рубцовского института (филиала) АлтГУ» «Как есть»

Декомпозиция диаграммы представлена на рисунке 1.3.

Диаграмма потоков данных рассматриваемого процесса представлена на рисунке 1.4.

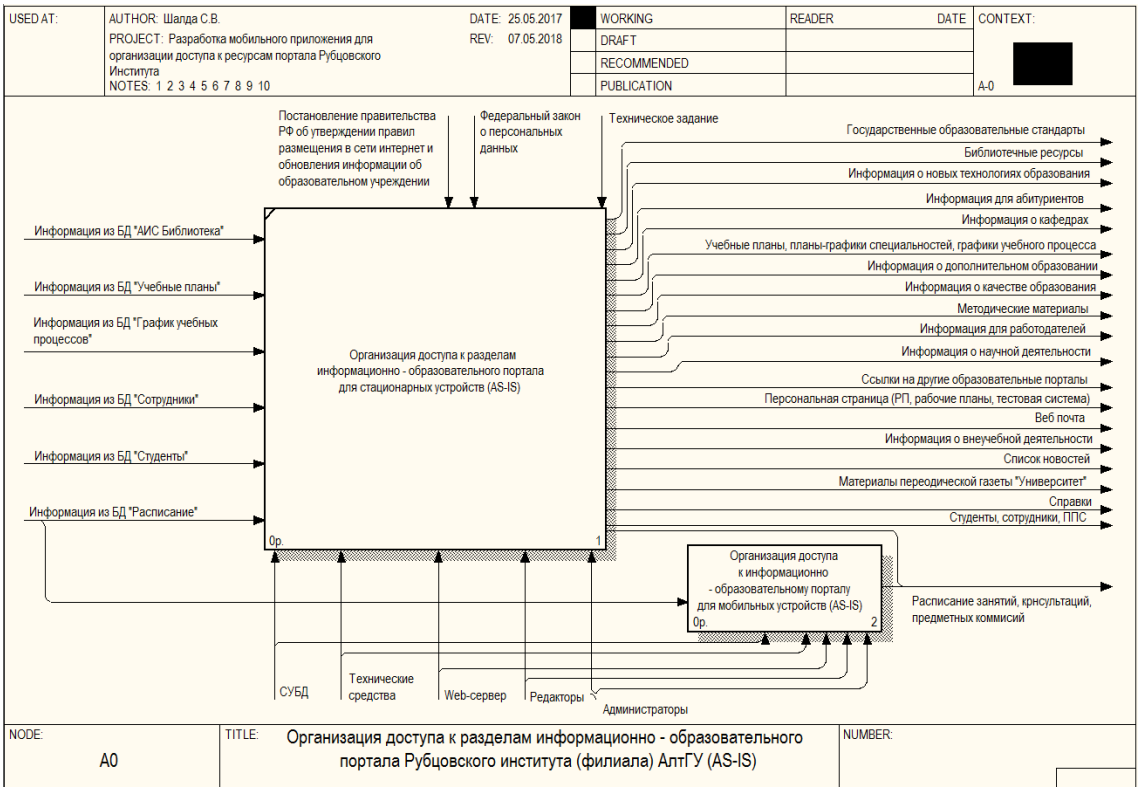

Рисунок 1.3 – Декомпозиция контекстной диаграммы процесса «Организации доступа к разделам информационно-образовательного портала» «Как есть»

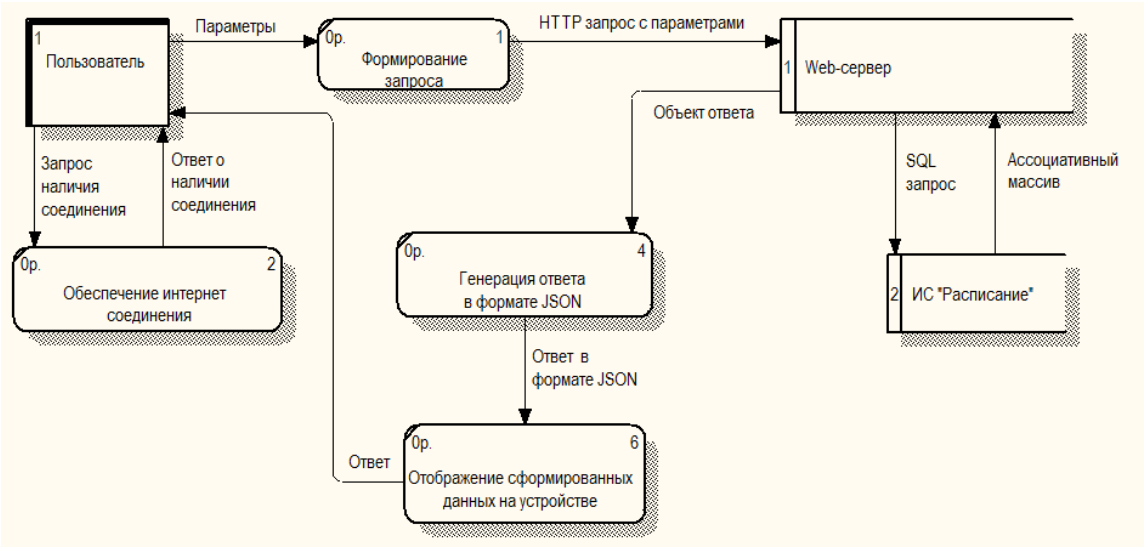

Рисунок 1.4 – Диаграмма потоков данных процесса «Организации доступа к разделам информационно-образовательного портала для мобильных устройств» «Как есть»

Способ получения информационных ресурсов мобильным устройством

пользователя посредством веб-портала имеет ряд недостатков:

– отсутствие способа для хранения полученной информации в памяти устройства;

– отсутствие возможности работать в автономном режиме – без интернета;

– отсутствие интерактивного взаимодействия с пользователем;

– отсутствие возможности кастомизации.

Еще одним способом доступа к ресурсам информационно-образовательного портала является использование мобильного приложения, данный способ немаловажен для низкоскоростных соединений и дорогого мобильного трафика. Мобильные приложения могут работать в режиме офлайн – без наличия интернета.

Исходя из этого, было принято решение о разработке мобильного приложения.

1.3 Определение цели и задач проектирования мобильного приложения

Мобильное приложение – последовательность инструкций, предназначенных для исполнения операционной системой мобильного устройства. Способ получения информации посредством мобильного приложения позволяет использовать настраиваемые параметры. Такими параметрами могут быть: автоматическое скачивание, получение интерактивных уведомлений, задание пользовательских настроек и другое.

Данные факторы обеспечивают удобство от использования мобильного приложения. В связи с этим можно определить цель работы: разработать мобильное приложение, которое бы обеспечивало пользователей удобным и постоянным доступом к необходимой им информации, а именно к своему расписанию, расписанию консультаций, предметных комиссий, личному

кабинету, заказу справок, новостям.

Исходя из цели, можно выделить задачи, решаемые мобильным приложением:

– получение расписания занятий, консультаций, предметных комиссий;

– корректное отображение информации на любых экранах мобильных устройств;

– обновление списков групп, преподавателей и аудиторий;

– возможность сохранения пользовательских настроек;

– автоматическая фоновая загрузка выбранного расписания;

– оповещение об изменениях состояния расписания;

– доступ к личному кабинету для просмотра текущей успеваемости;

– заказ необходимых справок;

– просмотр актуальных новостей информационного портала;

– экономия мобильного трафика.

«Мобильное приложение РИ АлтГУ» будет более удобным способом доступу к разделам расписаний, личному кабинету и новостной ленте информационно-образовательного портала.

1.4 Обзор и анализ существующих разработок, выбор технологии проектирования

Проектирование мобильного приложения по степени типизации будет оригинальным, по степени автоматизации – компьютерным. Поэтому необходимо провести анализ и выбор средств разработки приложения.

На данный момент существуют разнообразные технологии для разработки мобильных приложений – от создания HTML-приложения до

нативного мультиплатформенного приложения. Самый простой способ создания мобильных приложений – создание HTML-приложений.

HTML-приложения, иными словами – сайт, упакованный специальным образом. Данная технология использует системный веб-браузер для работы приложения. Для создания приложения разработчику необходимо знать язык гипертекстовой разметки – HTML, JavaScript и умение работать с фреймворками.

Фреймворки – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Самыми популярными фреймворками для создания мобильных HTML-приложений являются Cordova и Mobile Angular UI.

Cordova Cordova – это платформа, позволяющая разрабатывать мобильные приложения на разные платформы, путем встраивания браузера в мобильное приложение. Таким образом, приложение является, по сути, мини-браузером, который показывает один единственный сайт – приложение. Все ресурсы можно помещать в распространяемый пакет приложения для ускорения загрузки, а можно загружать с сервера при необходимости, предоставив программистам более гибкое API. Фреймворк позволяет применять законченный единый визуальный дизайн темы для мобильных сайтов и приложений.

По умолчанию Cordova предоставляет только базовые возможности браузера, которые есть на данном мобильном устройстве, но позволяет расширять набор функций, доступных в браузере путем использования плагинов. Каждый плагин предоставляет унифицированный интерфейс, который можно использовать из браузера на разных платформах.

И если поддерживаемые функции CSS/JS отличаются в каждом браузере от версии операционной системы или платформы, то функционал Cordova старается предоставлять унифицированный функционал для всех поддерживаемых версий мобильных ОС.

Несмотря на то, что данный подход имеет свои ограничения в виде необходимости учитывать различия в мобильных браузерах, все же он позволяет создавать качественные кросс-платформенные приложения. В отличие от фреймворка Cordova, Angular UI, который часто называют MVW (Model-View-Whatever), отличается тем, что имеет такие выгоды как: быстрое написание кода, быстрое тестирование любой части приложения и двухсторонняя привязка данных (изменения в бэкэнде сразу же отражаются на пользовательском интерфейсе). Сейчас фреймворк Angular UI заслуженно называют наиболее используемым JS фреймворком для разработки одностраничных приложений (SPA Single-Page-Applications) и он имеет крупнейшее сообщество разработчиков.

Преимуществами создания приложений по HTML технологии является кроссплатформенность. Недостатками таких приложений являются:

– основные возможности – возможности веб-браузера;

– слабая производительность.

Такое сочетание достоинств и недостатков не подходит для создания удобного и производительного приложения.

Цель же данного проекта – разработка программного продукта, который удовлетворяет запросам и возможностям пользователя. Гораздо выгоднее разработать приложение, которое будет поддерживать все заявленные функциональные возможности мобильной операционной системы, а не использовать лишь определенные, уже существующие, возможности обычных веб-браузеров.

Еще одной технологией разработки мобильного приложения является язык программирования – Java, который является нативным языком для разработки приложений на платформе Android.

Преимуществами такого выбора являются:

– скорость работы приложения;

– использование самых новых технологий в области разработки.

Предварительно было произведено сравнение интегрированных сред для разработки мобильного приложения по стоимости, гибкости и возможностям. На данный момент для разработки Android приложений существует 4 основных программных продукта – Android Studio, Eclipse (с плагином ADT), Embarcadero RAD Studio 10.2 Tokyo и Microsoft Visual Studio 2017 Community.

Eclipse – это интегрированная среда разработки (IDE), используемая в компьютерном программировании, и являющейся наиболее широко используемой Java IDE. Он содержит базовое рабочее пространство и расширяемую подключаемую систему для настройки среды.

Eclipse написан в основном на Java, и его основное использование предназначено для разработки Java-приложений, но оно также может использоваться для разработки приложений на других языках программирования через плагины, включая Ada , ABAP , C , C ++ , C # , COBOL, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog , Python , R, Ruby, Rust, Scala, Clojure, Groovy, Scheme и Erlang. Он также может использоваться для разработки документов с помощью LaTeX и пакетов для программного обеспечения Mathematica. Среда разработки включает в себя инструменты разработки Java Eclipse (JDT) для Java и Scala, Eclipse CDT для C / C ++ и Eclipse PDT для PHP и другие.

Android Development Tools (ADT) – это лучший плагин, предоставляемый Google для Eclipse IDE, который предназначен для создания приложений под Android.

ADT расширяет возможности Eclipse, позволяя разработчикам создавать новые проекты для Android, создавать пользовательский интерфейс приложений, добавлять пакеты на основе Android Framework API, отлаживать свои приложения с помощью инструментов Android SDK и экспортировать подписанные (или неподписанные) .apk-файлы в порядке распространять свои приложения. IDE для Android была официальной до выхода Android Studio (на основе IntelliJ IDEA Community Edition) и была

заменена. ADT официально устарел с конца 2015 года, и теперь Google ориентирован на Android Studio как официальную платформу Android IDE. Eclipse с плагином ADT распространяется бесплатно.

Android Studio на данный момент является официальной интегрированной средой разработки (IDE) от Google и разработан специально для разработки приложений под Android. Он доступен для загрузки в операционных системах Windows , MacOS и Linux. Данная среда разработки создана для замены Eclipse Android Development Tools (ADT) как основной среды разработки для разработки приложений для Android.

В текущей стабильной версии представлены следующие функции:

– поддержка сборки на базе Gradle;

– рефакторинг и быстрые исправления для Android;

– инструменты Lint для отслеживания производительности, удобства использования, совместимости версий и решения других проблем;

– возможность интеграции и использования приложений ProGuard;

– шаблонные мастер для создания общих конструкций и компонентов приложений Android;

– богатый редактор макетов, который позволяет пользователям перетаскивать компоненты пользовательского интерфейса, возможность предварительного просмотра макетов на разных конфигурациях экрана;

– поддержка создания приложений Android Wear;

– встроенная поддержка облачной платформы Google, позволяющая интегрироваться с облачными сообщениями Firebase и Google App Engine;

– android virtual device (эмулятор AVD) для запуска и отладки приложений в студии Android.

Android Studio поддерживает все те же языки программирования IntelliJ Python и Kotlin. Android Studio распространяется бесплатно.

Embarcadero RAD Studio – это интегрированная среда разработки (IDE) для быстрой разработки приложений настольного, мобильного, сетевого и

консольного программного обеспечения, разработанного Embarcadero Technologies.

Embarcadero RAD Studio использует язык Delphi, управляемый событиями. Компиляторы Delphi используют собственный диалект Object Pascal и генерируют собственный код для Microsoft Windows, MacOS, iOS, Android. С 2016 года выпускаются новые версии Delphi каждые шесть месяцев.

В текущей стабильной версии представлены следующие функции:

– наличие компилятора Delphi Android ARM для устройств и эмулятора;

– платформа FM platform для создания «нативных» приложений под Android;

– элемент управления Time Picker для Android, iOS, Windows;

– компонент Notification Center для Android и iOS;

– встроенная поддержка поисковой фильтрации для TListView на Android, iOS и Windows;

– жест «смахивания» для удаления на Android и iOS;

– поддержка «Share sheet» на Android и iOS;

– оптимизация производительности платформы FMDelphi RTL для Android;

– deployment manager для Android;

– поддержка отладки и запуска на встроенном эмуляторе;

– запуск приложений на устройствах под Android и IOS;

– удалённая отладка для Android.

Стоимость Embarcadero RAD Studio 10.2 Tokyo – 298 399 р.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также

веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework, Silverlight, Android и IOS .

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода.

Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

Visual Studio 2017 имеет встроенное расширение Xamarin для создания мультиплатформенных приложений.

Xamarin – это платформа разработки мобильных приложений для создания нативных приложений iOS, Android и Windows из общего кода C# или .NET, которая позволяет многократно использовать между платформами от 75 % до почти 100 % кода. Приложения, написанные с помощью Xamarin и C#, имеют полный доступ к интерфейсам API базовой платформы и возможность создавать нативные пользовательские интерфейсы, а также компилировать код в машинный, поэтому влияние на производительность во время выполнения является незначительным.

Разработчики, знакомые с C#, .NET и Visual Studio, могут рассчитывать на такие же возможности и производительность при работе с Xamarin для мобильных приложений, включая удаленную отладку на устройствах

Android, iOS и Windows, без необходимости изучать нативные языки, например Objective-C или Java. Удивительно, но много высокопроизводительных приложений с красивыми пользовательскими интерфейсами – например, NASCAR, Aviva и MixRadio – созданы с помощью Xamarin [6, с. 44-72].

Microsoft Visual Studio 2017 Community распространяется бесплатно.

Сравнение стоимости рассмотренных программных средств представлено в таблице 1.1.

Таблица 1.1 – Стоимость программных средств

Наименование	Стоимость
Embarcadero RAD Studio 10.2 Tokyo	298 399 руб.
Cordova или Angular UI (HTML)	Бесплатно
Android Studio	Бесплатно
Eclipse + Android Development Tools	Бесплатно
Microsoft Visual Studio 2017 Community	Бесплатно

Microsoft Visual Studio оказалась наиболее оптимальным выбором, так как она более гибкая для расширения функциональных возможностей приложения, по сравнению с остальными рассмотренными выше IDE, а так же является отличным решением для будущих разработок мобильных приложений для всех основных существующих операционных систем на мобильных устройствах и операционной системы Windows.

1.5 Выбор и обоснование проектных решений

Проектируемое приложение «Мобильное приложение РИ АлтГУ» будет предоставлять оперативную информацию о расписании, успеваемости и новостях Рубцовского института (филиала) Алтайского государственного университета.

Цель создания данного приложения – повышение эффективности и удобства работы с расписанием, что повлечет к сокращению случаев, связанных с отсутствием расписания и опозданиями студентов, успеваемостью, что позволит студентам всегда быть в курсе своей успеваемости, а так же новостям, знать, что происходит в жизни института.

Проектируемое приложение будет иметь клиент-серверную архитектуру.

Серверная часть имеет набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (API).

Клиентская часть будет разрабатываться с помощью интегрированной среды разработки Microsoft Visual Studio 2017 Community на языке C# и XML.

Чтобы реализовать приложение, необходимо указать все ограничивающие факторы, которые присутствовали во время проектирования. Эти факторы можно разделить на три категории:

– информация (оперативное отображение всех изменений в объектах управления);

– организация (использованные программные и аппаратные средства);

– экономика (сбережение в реализации проектов).

Информационная поддержка (ИП) включает такие системы как:

– классификации и кодирования;

– унифицированной документации, используемой в ИП;

– информационной базы, то есть информация из всех информационных массивов.

Существует три способа организации информационной базы данных (ИБ):

– файловая организация информационной базы;

– интегрированная информационная база,

– смешанная организация информационной базы.

Наиболее подходящей организацией информационной базы является интегрированная т.к. она представляет собой набор взаимосвязанных и сохраненных данных с такой минимальной избыточности, которая позволяет оптимально пользоваться ею в любых приложениях, обеспечивая при этом независимость и актуализацию данных при которой используется общий способ управления [14, с. 45-54].

Существуют три классические модели логической структуры базы данных (путем установления отношений между данными):

– иерархическая;
– сетевая;
– реляционная.

Реляционная модель основана на математической логике и является самой простой и наиболее знакомой формой представления данных в виде таблицы, поэтому в проекте реализуется этот метод организации хранения данных [12, с. 105-107].

Преимущества использования реляционных баз данных заключаются в том, что в реляционной модели данных существует только одна информационная структура, которая формализует табличное представление данных знакомое для пользователя.

Теоретическое обоснование – наличие теоретически обоснованных методов нормализации отношений позволяет получить базы данных с предопределенными свойствами (в основном, с гарантией минимальной избыточности представления данных).

Независимость данных – когда необходимо изменить структуру реляционной базы данных, это приводит к минимальным изменениям в программном продукте [12, с. 116].

Таким образом, в качестве информационного обеспечения разрабатываемого приложения будет использована реляционная база данных.

Программное обеспечение (ПО) – набор программ системы для обработки данных и программных документов, необходимых для работы

данных программ. Программное обеспечение предназначено для предоставления вычислительной системе определенных свойств, связанных с повышением производительности, повышением надежности полученных результатов, повышением надежности и повышением опыта работы пользователя.

Существует следующая классификация методов разработки программного обеспечения:

- метод «сверху вниз»;
- метод «снизу вверх»;
- метод структурного проектирования;
- метод структурного программирования;
- метод группы главного программиста;
- метод модульной конструкции.

В работе будет использован метод проектирования «сверху-вниз». Этот метод включает определение задачи в общих терминах, а затем постепенное уточнение структуры путем введения более мелких деталей. На каждом последующем этапе этого метода необходимо определять основные функции, которые нужно выполнить, то есть задача разбивается на несколько подзадач, пока они не станут настолько простыми, что каждая из них будет иметь один модуль, который может быть написан на алгоритмическом языке с помощью фразы. Далее описываются данные, основные структуры и вид обработки данных. Каждый модуль должен иметь тестовые данные, описанные с модулем, которые будут использоваться позже на этапе тестирования [13, с. 115-117].

Системным ПО в работе будет использоваться операционная система Windows. В комплекс программ функционального (прикладного, специального) ПО будут входить уникальные программы и функциональные пакеты прикладных программ (ППП). В работе будут использована прикладная программа, предназначенная для проектирования и разработки

программных средств – свободная интегрированная среда разработки приложений Microsoft Visual Studio 2017 Community.

Операции сбора и регистрации данных будут осуществляться с помощью различных средств, отображающих реальное состояние объекта на носители информации.

Заключительным этапом обработки информации является выдача результатной информации пользователю. Она будет осуществляться непосредственно на экран мобильного устройства [1, с. 187].

Обоснование выбора технического обеспечения (ТО).

В дальнейшем программное обеспечение мобильного решения может быть доработано требующейся функциональностью. В качестве операционной системы для разработки приложения была выбрана система Android, т.к. она является самой распространенной операционной системой на мобильных устройствах.

2 Проектная часть

2.1 Разработка функционального обеспечения

Выявленные для пользователей мобильных устройств недостатки в работе действующей мобильной версии мобильного приложения «РИ АлтГУ – Расписание» приводят к необходимости описания функционирования предметной области в представлении «Как должно быть». Задачей описания такой модели является нахождение мер блокирования отрицательного влияния неудовлетворительных бизнес-факторов, найденных при анализе.

Полученная модель представлена на рисунке 2.1.

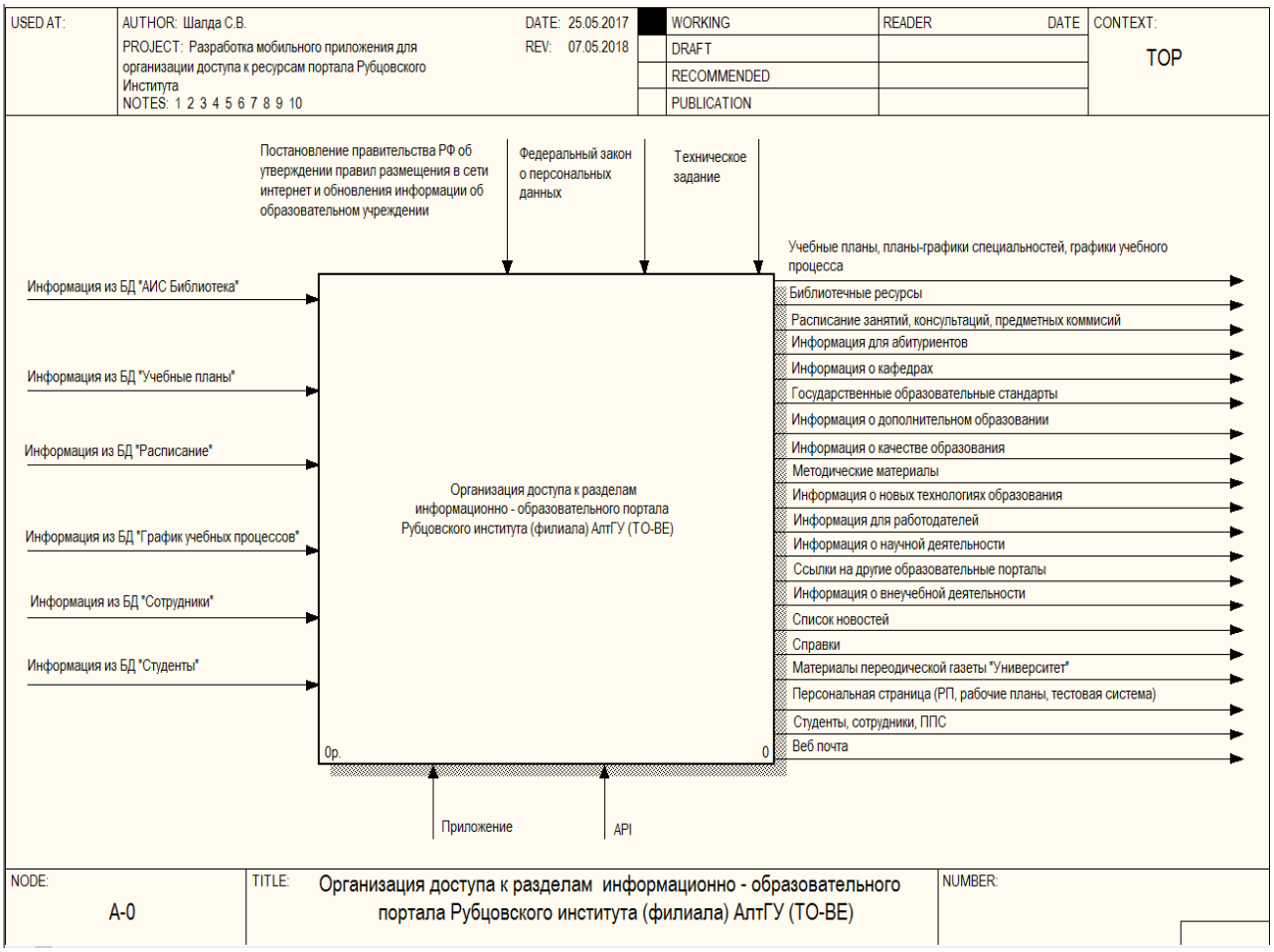

Рисунок 2.1 – Контекстная диаграмма процесса «Организации доступа к разделам информационно-образовательного портала Рубцовского института (филиала) АлтГУ» «Как должно быть»

На данной модели видно, что мобильное приложение будет содержать функционал, предоставляющий доступ к расписанию предметов, консультаций, предметных комиссий, личному кабинету и новостям.

Для данного решения это добавит удобство работы с необходимой информацией. На рисунке 2.2 представлена декомпозиция диаграммы «Как должно быть». На диаграмме видно, как изменится организация доступа к информации портала. Изменения коснулись входных, выходных данных и механизма управления.

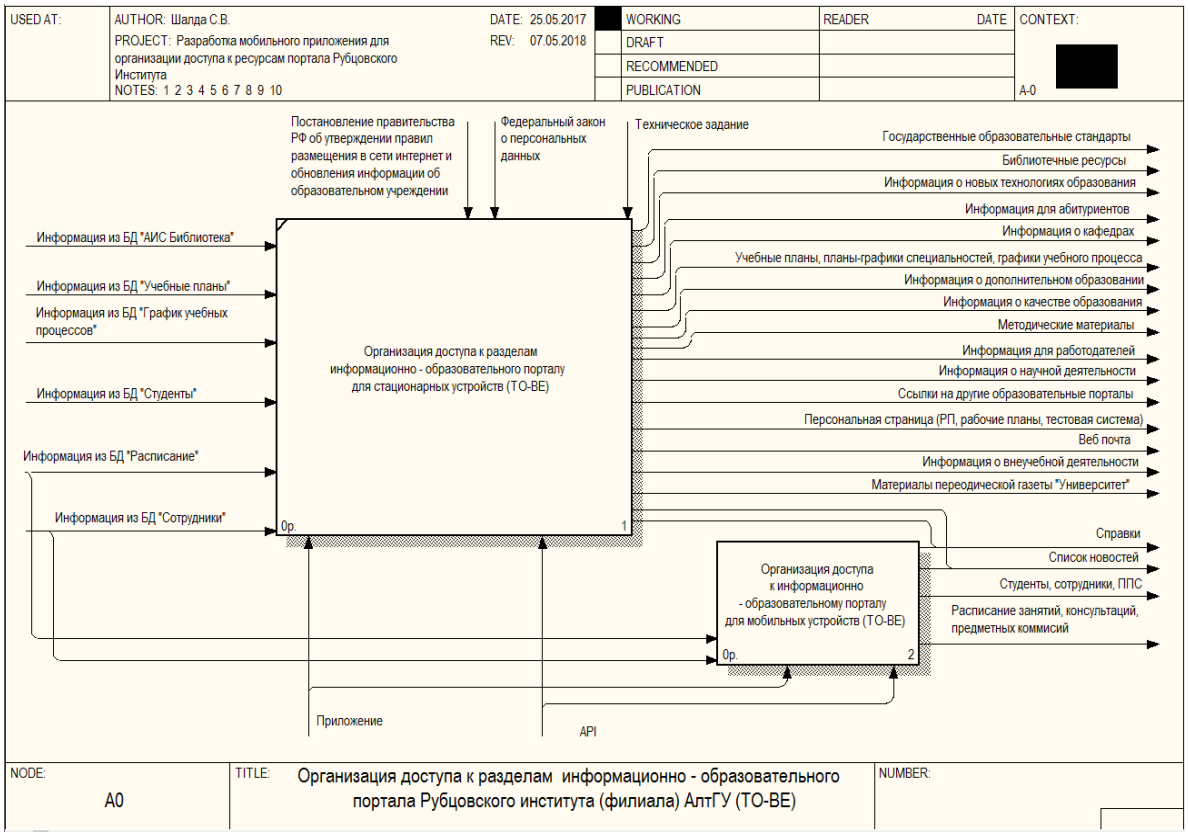

Рисунок 2.2 – Декомпозиция контекстной диаграммы процесса «Организации доступа к разделам информационно-образовательного портала» «Как должно быть»

На рисунке 2.3 представлена диаграмма потоков данных. Из полученной диаграмм были удалены лишние функциональные блоки, такие как:

– доступ к информации об образовании;

– доступ к формируемым отчетам;

– доступ к библиотеке;

– доступ к информации для абитуриентов.

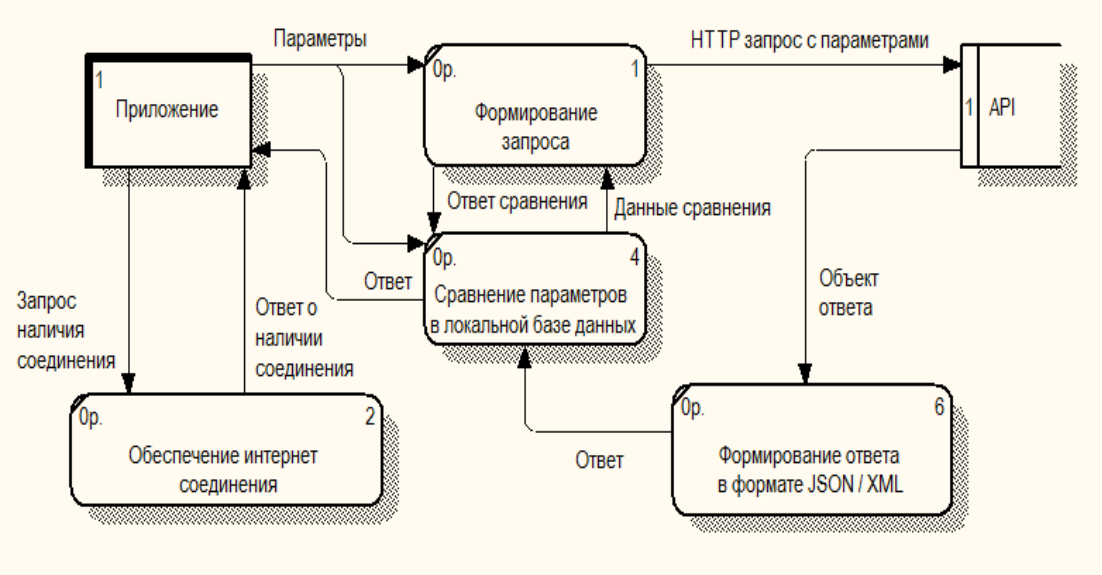

Рисунок 2.3 – Диаграмма потоков данных процесса «Организации доступа к разделам информационно-образовательного портала для мобильных устройств» «Как должно быть»

2.2 Разработка информационного обеспечения

2.2.1 Используемые классификаторы и системы кодирования

При реализации проекта для обеспечения уникальности объектов в пределах класса применяется порядковая система кодирования объектов.

При этом все множество объектов предварительно не упорядочивается, в результате чего всю работу по обеспечению максимальной скорости поиска объекта и определения его принадлежности во время сложной логической обработки с использованием имеющихся данных берет на себя СУБД.

Таким образом, некоторые таблицы имеют в своем составе автоинкрементное поле – идентификатор. Это позволяет не контролировать процессы генерации и присвоения уникальных кодов для записей программно, а полностью предоставить этот процесс под контроль СУБД.

Все даты представляются в стандартном для России формате «ДД.ММ.ГГГГ», где ДД – день месяца от 0 до 31, ММ – месяц от 0 до 12, ГГГГ – год в четырехзначном представлении.

Вся информация в базе данных представлена в кодировке UTF-8 – распространённой кодировке, реализующей представление Юникода, совместимое с 8-битным кодированием текста, позволяющая представить знаки практически всех письменных языков.

2.2.2 Характеристика нормативно-справочной и входной оперативной информации

Для обеспечения функционирования информационных систем и портала в Институте реализована структура справочников. Все методы объектов реализованы с помощью программных средств организации, а так же с использованием существующей системы управления содержимым сайта.

Для исследуемой предметной области можно выделить следующие основные справочники:

- студенты;
- преподаватели;
- группы;
- аудитории.

2.2.3 Характеристика результатной информации

Так как приложение будет иметь клиент-серверную архитектуру, в качестве результатной информации работы API портала (сервера) для мобильного приложения (клиента) будут выступать:

- списки групп, аудиторий и преподавателей;
- личная информация о студенте (форма вывода информации о номере группы, ФИО студента и других данных, предназначено

соответственно для студентов);

– сведения об успеваемости (форма вывода информации о текущей успеваемости и наличии (отсутствии) задолженностей, предназначено соответственно для студентов);

– расписание занятий (представляет собой интерактивную форму выбора вывода расписания по различным параметрам; предназначено в основном для преподавателей и студентов);

– расписание консультаций (форма вывода информации о проведении консультационных занятий преподавателей по времени и аудитории, предназначено соответственно для студентов и преподавателей);

– список предметных комиссий (форма вывода информации о проведении предметных комиссий, с составом преподавателей и времени проведения, сгруппированный по кафедрам, предназначено соответственно для студентов и преподавателей).

Все перечисленные результатные сведения будут передаваться мобильному устройству для дальнейшей обработки и представления в удобном для пользователя виде. В качестве формата для передачи данных мобильному устройству могут быть такие форматы, как: JSON, XML, YAML и INI.

Основными требованиями к формату стали:

– удобство редактирования;

– быстрота парсинга;

– быстрота сериализации, а так же размер в сериализованном виде;

– распространённость;

– поддержка редакторами.

Результаты экспертного сравнения форматов представлены в таблице 2.1. В итоге, исходя из показателей, в качестве формата для передачи мобильному устройству выбран формат передачи данных – JSON. Мобильное приложение, посредством POST-запроса, будет передавать

необходимые параметры серверу, а в качестве результатной информации будет получать ответ в формате JSON [3, с. 23-30].

Таблица 2.1 – Сравнение форматов данных

Свойство	JSON	XML	YAML	INI
Удобство восприятия	3	1	4	3
Удобство редактирования	3	1	4	3
Произвольная иерархия	3	3	3	1
Простота реализации	3	2	1	3
Скорость парсинга/сериализации	3	1	1	3
Размер в сериализованном виде	3	1	4	5
Поддержка поточной обработки	0	0	5	4
Бинарная безопасность	3	0	0	0
Распространённость	5	5	3	3
Поддержка редакторами	5	5	3	5
Поддержка языками программирования	5	5	3	5
Итоговая оценка	36	24	31	35

На рисунке 2.4 показан действующий способ доступа к ресурсам, в котором осуществлено взаимодействие между API и информационно-образовательным порталом с помощью цифровых устройств, посредством веб-сайта. Тот же способ взаимодействия будет использован и в разрабатываемом мобильном приложении.

Использование формата данных JSON позволит сократить мобильный трафик и упростить обработку результатной информации за счет того, что передаваемый JSON-объект содержит только данные в виде текста, в отличие от использования веб-сайта, когда клиенту передается весь исходный код страницы, включая данные разметки, компоненты и т.д.

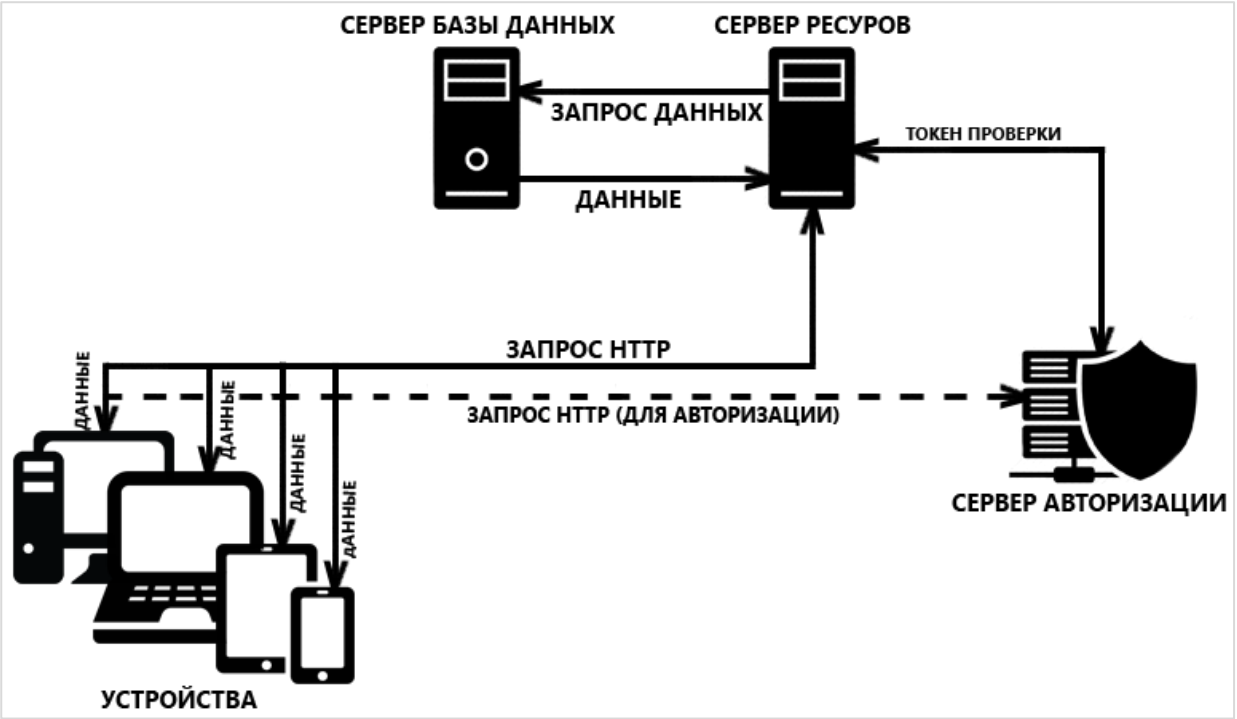

Рисунок 2.4 – Общая схема получения доступа к данным сервера

В результате объект JSON будет отправлен на мобильное устройство. JSON представляет собой текстовый формат обмена данными на основе JavaScript. Несмотря на свое происхождение из JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования.

На многих языках имеется готовый код для создания и обработки данных в формате JSON. Из-за его краткости по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, в этом ключе это актуально в задачах обмена данными между браузером и сервером (AJAX), а также между самими серверами (программными HTTP-интерфейсами). JSON-текст представляет собой одну из двух структур:

– набор пар ключей: значение. На разных языках это реализуется как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив. Ключ может быть только строкой, значение - любая форма;

– упорядоченный набор значений. На многих языках это реализуется как массив, вектор, список или последовательность.

Это универсальные структуры данных: как правило, любой современный язык программирования поддерживает их в той или иной форме. Они легли в основу JSON, поскольку используются для обмена данными между различными языками программирования.

2.2.4 Структура результатной информации

В качестве значений в JSON используются структуры:

1. Объектом является неупорядоченный набор пар ключ:значение, заключенное в фигурные скобки "{}". Ключ описывается строкой, символ «:» стоит между ним и значением. Ключи-значения разделяются друг от друга запятыми. Массив (одномерный) является упорядоченным набором значений.

2. Массив заключен в квадратные скобки «[]». Значения разделяются запятыми, значение может быть строкой в двойных кавычках, числом, объектом, массивом, одним из литералов: истинным, ложным или нулевым, поэтому структуры могут быть вложены внутрь друг друга.

3. Строка представляет собой упорядоченный набор из нуля или более символов юникода, заключенных в двойные кавычки. Символы могут быть указаны с использованием избавлений последовательностей, которые начинаются с обратного слэша «\».

Перед отправкой JSON файла необходимо определить его структуру. Структура JSON должна удовлетворять заданным требованиям:

– понятность для человека;

– простота редактирования;

– удобство для синтаксического анализа.

Переданные данные должны полностью описывать переданное расписание. Поэтому существует необходимость рассмотрения данных, которые будут формировать JSON файл [23].

Для расписания занятий необходимо передавать следующие данные:

– дата занятия;
– занятия;
– номер пары;
– id аудитории;
– наименование пары;
– наименование дисциплины;
– идентификатор преподавателей занятия;
– идентификаторы групп на занятии.

Структура файла для передачи расписания занятий представлена на рисунке 2.5.


```
{
  "Error": null,
  "Data": [
    {
      "Date": "05.02.2018 00:00",
      "Lessons": [
        {
          "Number": "3",
          "Room": "233",
          "LessonName": "Лекция №2",
          "DisciplineName": "Интеллектуальные информационные системы",
          "Teachers": [
            "43c9db1e-a271-11de-a57b-505054503030"
          ],
          "Groups": [
            "2798",
            "2574"
          ]
        },
        {
          "Number": "4",
          "Room": "27",
          "LessonName": "Лабораторная работа №2",
          "DisciplineName": "Учебное предприятие",
          "Teachers": [
            "d68878a0-ade5-11dc-8c11-0016e652202c"
          ],
          "Groups": [
            "2798",
            "2574"
          ]
        },
        {
          "Number": "3",
          "Room": "46",
          "LessonName": "Лабораторная работа №4",
          "DisciplineName": "Сертификация 1С:Профессионал",
          "Teachers": [
            "e3b75a0b-884f-11e5-95d4-00259001ddd3"
          ],
          "Groups": [
            "2798",
            "2574"
          ]
        }
      ],
      "TypeHour": "Workday"
    }
  ]
}
```


Рисунок 2.5 – Структура JSON передаваемого расписания занятий

В данном JSON запросе имеются запросы, которые выполняются отдельно от запроса расписания, такие как наименование аудитории, данные о преподавателе и наименование групп. Структура этих запросов представлена на рисунках 2.6-2.8.

Рисунок 2.6 – Структура JSON передаваемого наименования аудитории

Рисунок 2.7 – Структура JSON передаваемого наименования группы

```
{
  "Error": null,
  "Data": [
    {
      "ID": "92f7341b-ade5-11dc-8c11-0016e652202c",
      "LASTNAME": "Анисимов",
      "FIRSTNAME": "Константин",
      "PATRONYMIC": "Геннадьевич",
      "POSITION": "Доцент",
      "ACADEMICSTATUS": "Кандидат физико-математических наук"
    },
    {
      "ID": "92f7341c-ade5-11dc-8c11-0016e652202c",
      "LASTNAME": "Анисимова",
      "FIRSTNAME": "Елена",
      "PATRONYMIC": "Александровна",
      "POSITION": "Доцент",
      "ACADEMICSTATUS": "Кандидат технических наук"
    },
    {
      "ID": "1e4ff2d4-a58e-11e2-9287-00259001ddd3",
      "LASTNAME": "Аничкин",
      "FIRSTNAME": "Евгений",
      "PATRONYMIC": "Сергеевич",
      "POSITION": "Профессор",
      "ACADEMICSTATUS": "Доктор юридических наук"
    },
    {
      "ID": "43c9db1e-a271-11de-a57b-505054503030",
      "LASTNAME": "Шевченко",
      "FIRSTNAME": "Алеся",
      "PATRONYMIC": "Сергеевна",
      "POSITION": "Доцент",
      "ACADEMICSTATUS": "Кандидат физико-математических наук"
    }
  ]
}
```


Рисунок 2.8 – Структура JSON передаваемых данных о преподавателе

В данном JSON запросе так же имеются запросы, которые выполняются отдельно от запроса расписания консультаций, это тип расписания звонков. Идентификатор преподавателя и идентификатор аудитории были представлены на рисунках 2.6 и 2.8.

Структура этого запроса представлена на рисунке 2.9.

Рисунок 2.9 – Структура JSON передаваемого типа расписания звонков

Для расписания консультаций необходимо передавать следующие данные:

– идентификатор аудитории;
– дата консультации;
– номер пары;
– идентификатор преподавателя;
– тип расписания звонков.

Структура файла для передачи расписания консультации представлена на рисунке 2.10.

Рисунок 2.10 – Структура JSON передаваемого расписания консультаций

Для расписания предметных комиссий необходимо передавать следующие данные:

– название предметной комиссии;
– идентификатор аудитории;
– дата;

– идентификаторы преподавателей в предметной комиссии;

– тип расписания;

– список кафедр;

– номер пары.

Структура файла для передачи предметных комиссий представлена на рисунке 2.11.


```
{
  "Error": null,
  "Data": [
    {
      "Name": "Иностранный язык      ",
      "Date": "23.04.2018 00:00",
      "IDRoom": "126",
      "Teachers": [
        "09462ff2-e8f4-11dc-a555-505054503030",
        "ddfbd20a-ade5-11dc-8c11-0016e652202c",
        "ddfbd1c4-ade5-11dc-8c11-0016e652202c",
        "e4f7f56e-ade5-11dc-8c11-0016e652202c"
      ],
      "TypeHour": "Workday",
      "LessonNumber": 6
    },
    {
      "Name": "История, философия, БЖД, русский язык и культура речи, литература, физическая культура ",
      "Date": "30.04.2018 00:00",
      "IDRoom": "126",
      "Teachers": [
        "bae8fa37-ade5-11dc-8c11-0016e652202c",
        "ad894957-ade5-11dc-8c11-0016e652202c",
        "5b7cd357-6e57-11e6-80e7-00259001ddd2",
        "ad894943-ade5-11dc-8c11-0016e652202c",
        "ad894900-ade5-11dc-8c11-0016e652202c"
      ],
      "TypeHour": "Workday",
      "LessonNumber": 6
    }
  ]
}
```


Рисунок 2.11 – Структура JSON передаваемой информации о предметных комиссиях

В данном запросе имеется дополнительный запрос, который

выполняются отдельно от основного, и содержит список кафедр. Структура этого запроса представлена на рисунке 2.12.

Рисунок 2.12 – Структура JSON передаваемой информации о списке кафедр

Для списка пересдач необходимо передавать следующие данные:

– дата;
– номер пары;
– идентификатор преподавателя;
– идентификатор аудитории;
– тип расписания звонков.

Структура файла для передачи списка пересдач представлена на рисунке 2.13.

Рисунок 2.13 – Структура JSON передаваемой информации о предметных комиссиях

Так же существует дополнительный JSON запрос, который необходим для работы приложения, а именно список новостной ленты. Для списка новостной ленты необходимо передавать следующие данные:

– заголовок новости;
– категория новости;
– дата новости;
– ссылка на изображение новости;
– ссылка с новостью на портале.

Структура файла для передачи списка новостей представлена на рисунке 2.14.


```
{
  "Error": null,
  "Data": [
    {
      "Name": "«Разные люди» играют в полуфинале «Лиги КВН АлтГУ»",
      "Category": "События",
      "Date": "19.04.2018 11:11",
      "URLPhoto": "/public/uploads/1524111062_KVN_polufinal.jpg",
      "URL": "/content/article/9687#start"
    },
    {
      "Name": "В Институте прошли олимпиады по математике и физике для студентов высшего и среднего профессионального образования",
      "Category": "События",
      "Date": "19.04.2018 09:00",
      "URLPhoto": "/public/uploads/1524103256_AGU_olimpiada.jpg",
      "URL": "/content/article/9686#start"
    },
    {
      "Name": "Рубцовский институт (филиал) АлтГУ проводит конкурс социальных проектов «Если бы я стал Губернатором?»",
      "Category": "События",
      "Date": "13.04.2018 14:53",
      "URLPhoto": "/public/uploads/1523605982_Gubernator.jpg",
      "URL": "/content/article/9656#start"
    },
    {
      "Name": "На юбилейный Форум «АТР» отправятся 13 представителей Рубцовского Института (филиала) АлтГУ",
      "Category": "События",
      "Date": "13.04.2018 14:41",
      "URLPhoto": "/public/uploads/1523869708_2.png",
      "URL": "/content/article/9654#start"
    }
  ]
}
```


Рисунок 2.14 – Структура JSON передаваемой информации о списке новостей

2.2.5 Информационная модель и ее описание

Начальным этапом проектирования системы баз данных является построение семантической модели домена, основанной на анализе свойств и природы объектов в домене и информационных потребностях будущих пользователей разрабатываемой системы. Этот этап называется концептуальным дизайном системы, а его результат – концептуальной (инфологической) моделью области [16, с. 44].

Такие модели обобщают информационные потребности пользователей системы, созданных с точки зрения использования хранимых данных и по существу являются средством коммуникации как для разработчиков, так и для пользователей на разных этапах жизненного цикла базы данных.

Инфологические модели включают различные компоненты, по-разному и различные средства, отражающие предметную область. Помимо наиболее известного описания объектов и связей между ними, к инфологическому уровню описания предметной области можно отнести следующие компоненты:

– систему атрибутов и средств описания предметной области;

– логические связи между индикаторами или лингвистические свойства языка, используемые для вербального представления объектов;

– ограничения целостности данных, которые определяют достоверность значения отдельных полей и отношений, как на уровне семантики содержимого базы данных, так и ее физической структуры;

– описание информационных потребностей пользователей, отражающих процедурные характеристики доступа к данным.

Entity решает проблему несоответствия типов путем внедрения посредника между реляционной базой данных и классом в .NET. Посредник – это абстрактная модель EDM (Entity Data Model).

Она хранит информацию о таблицах, столбцах и связях в виде строготипизированых классов.

В приложениях, где модель предметной области это база данных, какой бы подход не был использован, работа будет осуществлена с помощью модели EDM. Проектирование новой базы данных – это моделирование, и объявление классов C# – тоже моделирование.

Вся суть этапов работы с Entity Framework делится на два вида:

– моделирование;

– доступ к моделям и сущностям для изменений.

Традиционный подход к разработке модели предметной области – это последовательная конкретизация модели, которая содержит в себе следующие этапы:

– модель домена;

– логическая модель базы данных;

– физическая модель базы данных.

Модель домена – определяет какие таблицы будут в базе данных и как они будут связаны. Это самая высокая абстракция.

Логическая модель – включает информацию о ключах таблиц и всех ограничениях – это нормализованная модель.

Физическая – определяет в каких файлах хранятся таблицы и как эти файлы хранятся на сервере.

Этот подход работал многие годы. Физическую модель разрабатывают поставщики базы данных и совершенствуют администраторы.

Модель домена и логическую модель совершенствуют проектировщики.

На деле, модель домена почти никогда не разрабатывалась, а если и разрабатывалась, то только в виде общей схемы сущностей и связей.

Проектировщики сразу начинали определять таблицы, внешние и первичные ключи, то есть создавали базу данных. Программистам доставалась готовая логическая модель, к которой формировались SQL запросы и хранимые процедуры, из чего следует что самых главный этап – логический, на него делался упор и в нем старались максимально правильно

сформировать схему связей.

Entity это изменил, поставив на первое место модель домена, назвав ее концептуальной моделью. Оставил логическую модель и добавил еще одну связующую между ними – модель сопоставления [25].

Каждая модель хранится в отдельном XML-файле, который является частью EDM:

1. Концептуальная модель (модель домена) – в файлах с расширением CSDL.
2. Модель хранения данных (логическая модель) – в файлах с расширением SSDL.
3. Модель сопоставления (Мэппинг) – в файлах с расширением MSL.

Каждый из этих файлов можно изменить, управляя общей моделью EDM. Изменения в концептуальной модели требуют изменений в моделях хранения и сопоставления.

Последние можно изменять без необходимости менять концептуальную модель.

То, что Entity Framework ставит во главу концептуальную модель, освобождает от необходимости писать специализированные (для конкретной базы данных) SQL запросы и вызывать хранимые процедуры.

После моделирования следует разработка кода манипуляции с сущностями в базе данных. Такой код разрабатывается на основе:

1. Entity SQL.
2. LINQ to Entities.

Оба подхода подразумевают запросы к концептуальной модели.

LINQ – это набор методов расширения языков .NET, позволяющий манипулировать сущностями в БД не зависящие от источника данных.

Запросы LINQ транслируются деревьям команд, а затем в запросы SQL, специфичные для хранилища [22].

Архитектура взаимодействия представлена на рисунке 2.15.

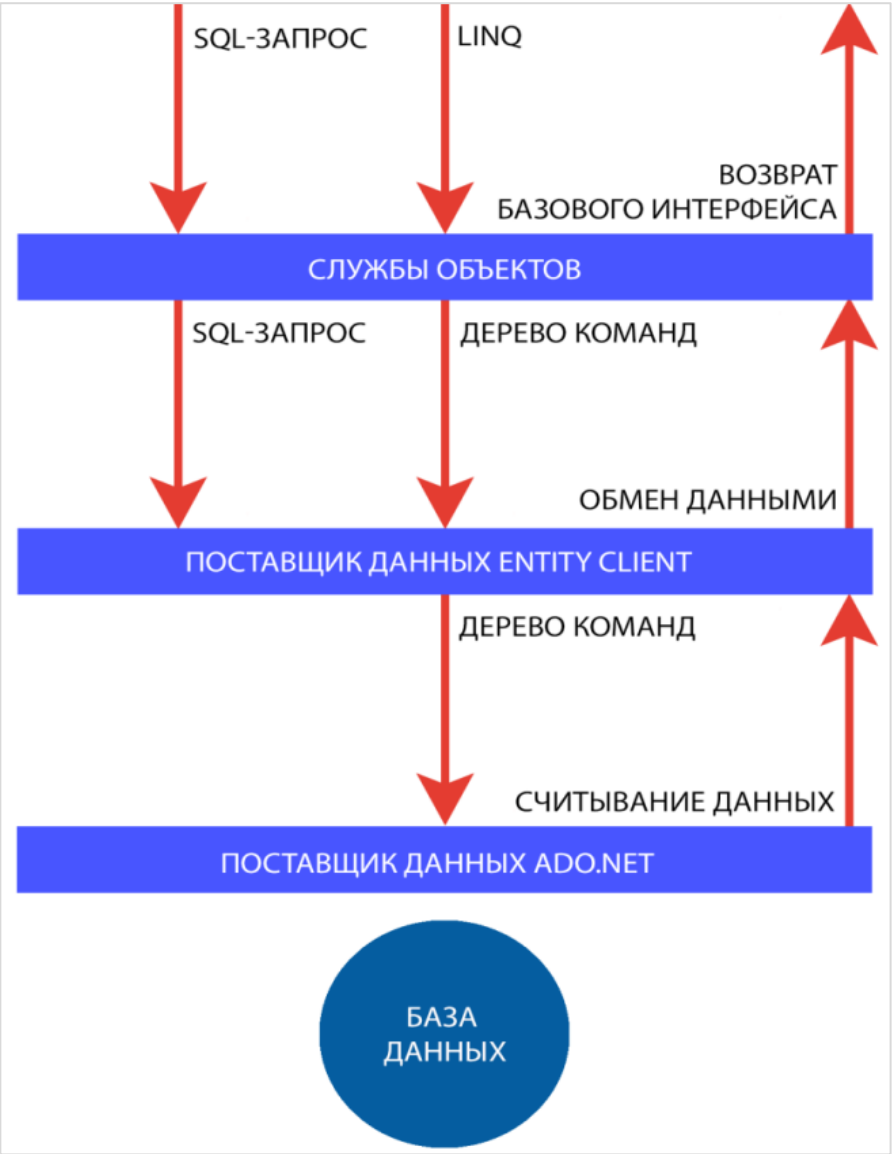

Рисунок 2.15 – Архитектура доступа к данным

Entity SQL – это обобщенный SQL, не зависимый от конкретного хранилища. Реализует более низкоуровневый подход.

В рамках Entity Framework существует три различные возможности проектирования базы данных:

– code-first;
– model-first;
– database-first.

Database-First был первым подходом, который появился в Entity Framework. Данный подход во многом похож на Model-First и подходит для тех случаев, когда разработчик уже имеет готовую базу данных.

Позже был добавлен подход Model-First, который очень долгое время был одним из приоритетных подходов. Он позволял создать вручную с помощью визуального редактора модель EDMX (Entity Data Model XML), и по ней создать базу данных.

EDMX-файл – это XML файл, определяющий концептуальную модель, режим хранения и их сопоставление.

Кроме того, EDMX-файл содержит сведения, используемые конструктором моделей EDM (конструктор сущностей) для подготовки модели к просмотру в графическом режиме [22].

На рисунке 2.16 представлена схема разработанной локальной базы данных, на которой указана схема взаимодействия сущностей. База данных была автоматически сгенерирована при помощи механизма Code-First в следствии работы инфраструктуры Entity Framework Core.

Данные модели служат для формирования данных сущностей а так же непосредственно для хранения данных передаваемых API.

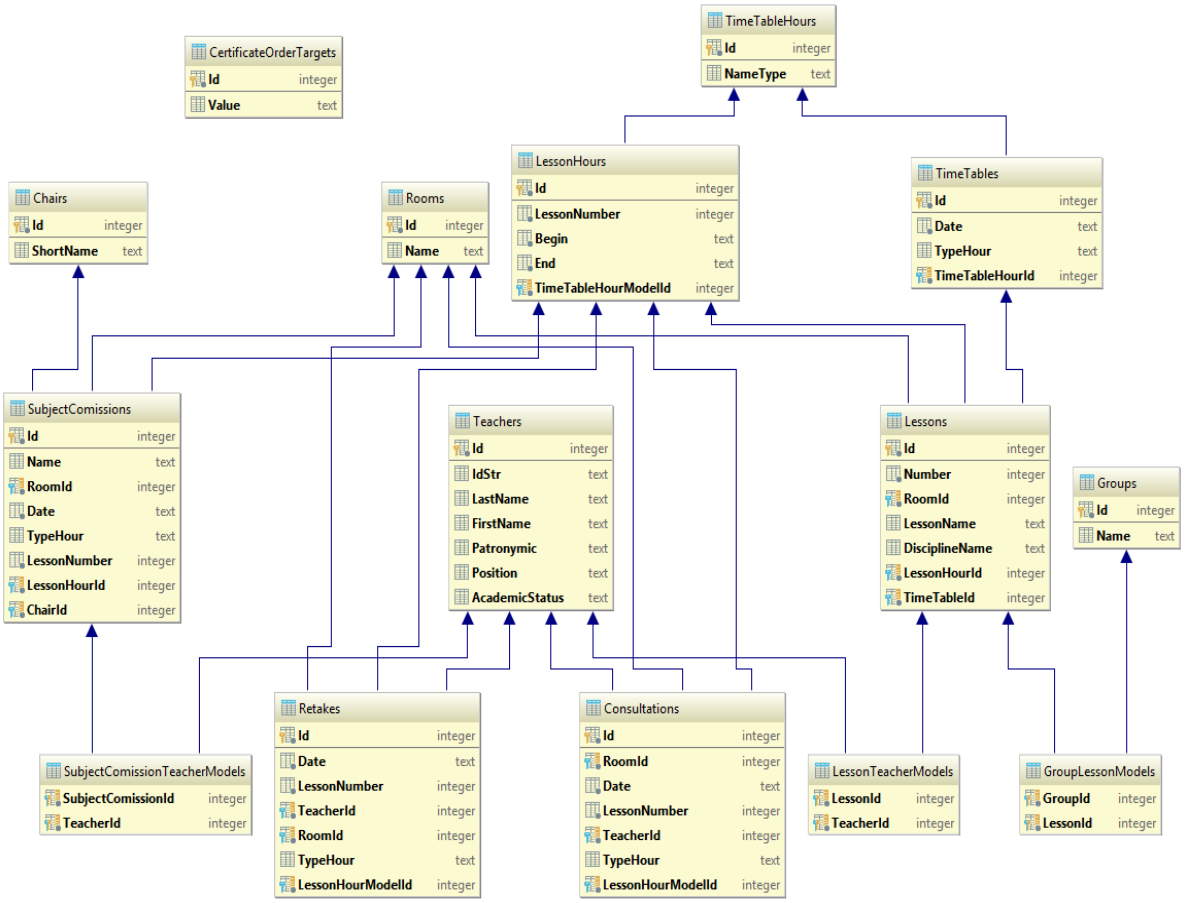

Рисунок 2.16 – Схема локальной базы данных

На схеме имеются следующие таблицы:

– chairs – таблица кафедры, которая хранит данные о кафедрах;

– subjectcomissions – таблица «предметные комиссии», которая хранит данные о предметных комиссиях;

– subjectcomissionteachermodels – таблица «преподаватели предметных комиссий», которая хранит данные о преподавателях предметных комиссий;

– retakes – таблица «пересдачи», которая хранит данные о пересдачах;

– groups – таблица «группы», которая хранит данные об учебных группах;

– lessons – таблица «занятия», которая хранит данные о всех учебных занятиях;

– rooms – таблица «аудитории», которая хранит данные о аудиториях;

– teachers – таблица «преподаватели», которая хранит данные обо всех преподавателях;

– consultations – таблица «консультации», которая хранит данные о консультациях;

– lessonteachermodels – таблица «преподаватели занятий», которая хранит данные о преподавателях занятий;

– grouplessonmodel – таблица «занятия групп», которая хранит данные о занятиях групп;

– timetable – таблица расписание занятий, которая хранит данные о расписании занятияй;

– lessonhours – таблица «время начала и окончания занятий», которая хранит данные о времени начала и окончания занятий;

– timetablehours – таблица «время расписания», которая хранит данные о времени расписания;

– certificateordertargets – таблица «типы справок», которая хранит данные о типах справок.

Последние несколько лет предпочтительным подходом для создания базы данных становится тактика Code-First. При подходе Code-First проектирование базы данных начинается с кода, а не с создания базы данных или ее модели.

Для предметной области проектируемого приложения создаются необходимые классы, а механизм Code-First позволяет им участвовать в работе инфраструктуры Entity Framework, которая берет на себя всю обработку взаимодействия с базой данных и автоматически отслеживает изменения.

В качестве способа хранения данных в мобильном приложении была выбрана сериализация.

Сериализация – процесс перевода структур данных в последовательность битов. Обратной к операции сериализации является операция десериализации (структуризации) – восстановление начального состояния структуры данных из битовой последовательности. Сериализация используется для передачи объектов по сети и для сохранения их в файлы [16, с. 55]. Использование данного способа хранения данных обусловлено следующими факторами:

– небольшое количество данных;
– отсутствие сложных выборок;
– скорость работы;
– простота реализации.

Данные будут сохраняться в памяти мобильного устройства в файле с расширением *.db.

2.3 Разработка программного обеспечения

Проектируемое «Мобильное приложение РИ АлтГУ», позволяет:

– получать расписание занятий, консультаций, предметных комиссий;

– корректно отображать информацию на любых экранах мобильных устройств;

– обновлять списки групп, преподавателей и аудиторий;

– сохранять пользовательские настройки;

– выполнять автоматическую фоновую загрузку выбранного расписания;

– оповещать об изменениях состояния расписания;

– иметь доступ к личному кабинету для просмотра текущей успеваемости;

– заказывать необходимые справки;

– просматривать актуальные новости информационного портала;

– экономить мобильный трафик.

Разработка мобильного приложения произведена с помощью языка C# и XML. Серверная часть имеет набор готовых классов, процедур, функций, структур и констант, которые предоставляются приложению посредством API.

C# (произносится «Си-шарп») – это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework.

Язык C# прост, типобезопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C [19, с. 15-18].

API (программный интерфейс приложения, интерфейс прикладного программирования) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

2.3.1 Описание серверных программных модулей

В Рубцовском институте (филиале) АлтГУ, для обмена данными между корпоративной информационной системой и приложениями, реализовано интегрированное API, поэтому для обеспечения функционирования приложения разработка серверных проектных модулей не нуждается в реализации.

API (программный интерфейс приложения, интерфейс прикладного программирования) (англ. application programming interface) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Функциональность определяется приложением (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

Программные компоненты взаимодействуют друг с другом посредством API.

При этом обычно компоненты образуют иерархию – высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

По такому принципу построены протоколы передачи данных в сети Интернет.

Стандартный стек протоколов (сетевая модель OSI) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам HTTP и IMAP).

Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню, для обеспечения лучшей работы [14, с. 76-107].

API библиотеки функций и классов включает в себя описание сигнатур

и семантики функций, таких как – REST (Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле, компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

REST является альтернативой RPC. REST API подразумевает под собой простые правила:

– каждый URL является ресурсом;

– при обращении к ресурсу методом GET возвращается описание этого ресурса;

– метод POST добавляет новый ресурс;

– метод PUT изменяет ресурс;

– метод DELETE удаляет ресурс.

Эти правила предоставляют простой CRUD интерфейс (создать, прочесть, обновить, удалить) для других приложений, взаимодействие с которым происходит через протокол HTTP.

REST API интерфейс очень удобен для межпрограммного взаимодействия, например, мобильное приложение может выступать в роли клиента, которое манипулирует данными посредством REST.

2.3.2 Описание клиентских программных модулей

Для полноценного функционирования приложения необходимо наличие следующих модулей:

1. Модуль для работы с API. Модуль вынесен в отдельное решение, для возможности реализации функционала на других платформах и

приложениях. Данный модуль реализует программный интерфейс обмена данными с серверной частью API.

Вид данного модуля представлен на рисунке 2.17.

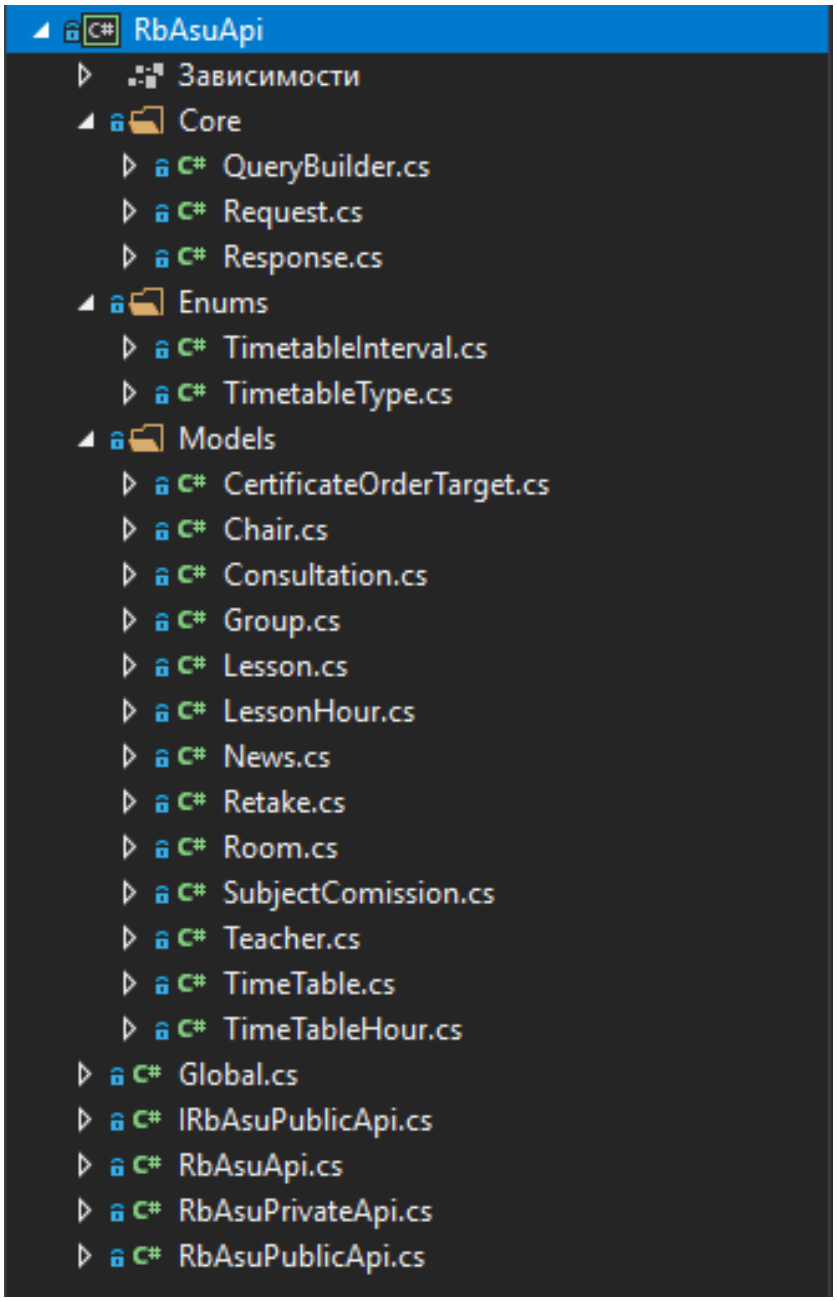

Рисунок 2.17 – Модуль для работы с API

2. Модуль внедрения зависимостей (сервис). Данный модуль хранит в себе сведения о других модулях, а так же сведения о том, как их использовать.

В этом модуле используется принцип инверсии управления (англ. Inversion of Control, IoC) – важный принцип объектно-ориентированного

программирования, используемый для уменьшения зацепления в компьютерных программах и мобильных приложениях, также архитектурное решение интеграции, упрощающее расширение возможностей системы, при котором контроль над потоком управления программы остаётся за каркасом.

В приложении реализацией IoC стало применении управления зависимостей в виде внедрения зависимостей (англ. dependency injection).

Если сравнить с более низкоуровневыми технологиями, IoC-контейнер – это компоновщик, который собирает не объектные файлы, а объекты ООП (экземпляры класса) во время исполнения приложения. Аналогом такого компоновщика является компилятор, одной из функций которого является создание объектных файлов [16, с. 99].

Предоставление инструментов для внедрения зависимостей дало значительно большую гибкость в разработке и удобство в тестировании кода. Вид данного модуля представлен на рисунке 2.18.

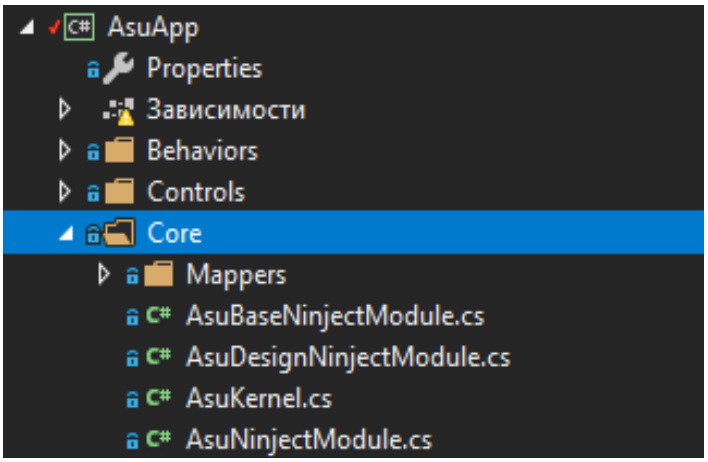

Рисунок 2.18 – Вид модуля внедрения зависимостей

3. Модуль работы с базой данных. Данный модуль служит для хранения, записи и удалениях информации из локальной базы данных. Вид данного модуля представлен на рисунке 2.19.

Объектно-ориентированная, легковесная и расширяемая технология от компании Microsoft – EF Core (Entity Framework Core) служит для взаимосвязи и доступа к данным. Вид данного подмодуля представлен на рисунке 2.19.

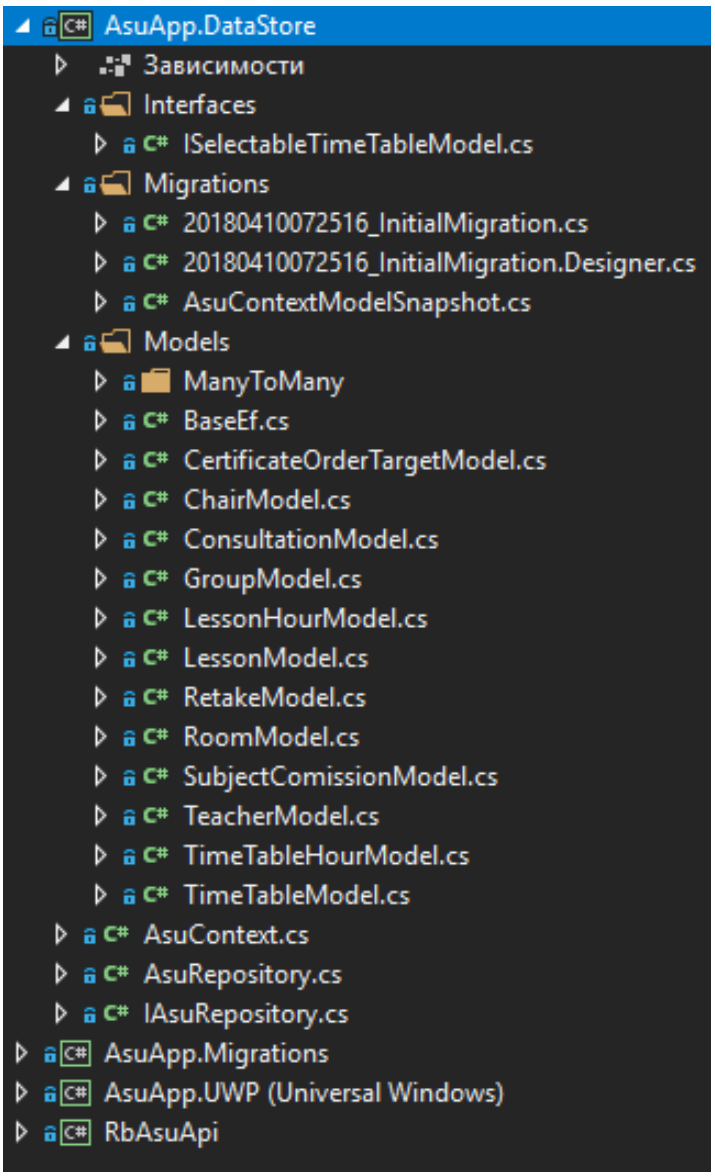

Рисунок 2.19 – Модуль для работы с базой данных

EF Core является ORM-инструментом (object-relational mapping – отображение данных на реальные объекты), то есть EF Core позволяет работать с базами данных, но представляет собой более высокий уровень абстракции. Данный инструмент позволяет абстрагироваться от самой базы данных, а так же ее таблиц, и работать с данными независимо от типа хранилища. Если на физическом уровне происходит оперирование таблицами, первичными и внешними ключами, то на концептуальном уровне, который предлагает Entity Framework, идет работа с объектами.

Entity Framework Core поддерживает множество различных систем баз данных. По умолчанию технология предоставляет ряд встроенных

провайдеров для работы, а именно с SQlite и MySQL. Так же существуют и сторонние провайдеры, например: CouchDB, Cassandra, HBase и другие [22].

CouchDB – нереляционная распределённая база данных, которая реализована в рамках подхода NoSQL. Для хранения данных используется JSON а для реализации Map / Reduce-запросов – JavaScript. Одной из особенностей СУБД является поддержка репликации с несколькими ведущими узлами. CouchDB можно рассматривать как сервер веб-приложений. Не распределенная база данных CouchDB, может служить прослойкой для хранения данных и распределенных систем.

Cassandra – нереляционная распределённая база данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша. Использует модель хранения данных на базе семейства столбцов (Memtable / SSTable) а для реализации Thrift запросы – JavaScript, чем отличается от систем, подобных MemcacheDB, которые хранят данные только в связке «ключ – значение» [3, с. 56-58].

HBase – нереляционная распределённая база данных, которая реализована в рамках подхода NoSQL, с открытым исходным кодом, которая написана на Java и является аналогом Google BigTable. Данные хранятся в таблицах, состоящих из строк и столбцов. Для ячеек таблицы (пересечения строк и столбцов) действует контроль версии. По умолчанию в качестве версии используется временная метка, автоматически назначаемая HBase на момент вставки. Содержимое ячейки представляет собой неинтерпретируемый массив байтов. Для хранения данных используется JSON а для реализации Map / Reduce-запросов – JavaScript [14, с. 55-57].

SQlite – компактная встраиваемая реляционная СУБД. Слово «встраиваемая» означает, что SQLite не использует парадигму клиент-сервер, то есть SQLite не является отдельно работающим процессом, с которым взаимодействует приложение, а представляет собой библиотеку, с которой компонуется приложение, и становится его составной частью.

Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает приложение. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том устройстве, на котором выполняется приложение. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется, а ACID-функции достигаются, в том числе за счёт создания файла журнала.

MySQL – свободная реляционная система управления базами данных. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей.

MySQL имеет архитектуру клиент-сервер, что не позволяет ей быть встраиваемой СУБД.

Гибкость СУБД MySQL обеспечивается поддержкой типов таблиц, таких как MyISAM поддерживающий полнотекстовый поиск и InnoDB поддерживающий транзакции на уровне отдельных записей.

Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц [22].

Все из представленных баз данных обеспечивают высокий отказоустойчивый способ хранения больших объёмов данных. В таблице 2.2 приведен сравнительный обзор рассмотренных типов баз данных.

Благодаря своей архитектуре, возможности реализации реляционных баз данных, встраивания, а так же стандартно поставляемой СУБД, SQLite стала отличным выбором для создаваемого приложения, поэтому локальная база данных была реализована с помощью этой СУБД [9, с. 155-174].

Для наглядности хранения данных, была поострена схема локальной базы данных, которая была представлена на рисунке 2.15.

Таблица 2.2 – Сравнение типов баз данных

Наименование	Модель данных	Представление данных	Возможность встраивания
SQLite	Append-only B-Tree (документы)	Реляционная (SQL)	Да
MySQL	Append-only B-Tree (документы)	Реляционная (SQL)	Нет
CouchDB	Append-only B-Tree (документы)	Не реляционная (NoSQL)	Да
Cassandra	Memtable / SSTable (семейства столбцов)	Не реляционная (NoSQL)	Да
HBase	Memtable / SSTable on HDFS (семейства столбцов)	Не реляционная (NoSQL)	Да

4. Подмодуль осуществления связи между локальной базой данных и API. Данный модуль осуществляет поддержку связи между модулем хранения локальной базы данных и модулем API, посредством осуществления запросов к модулю API и последующей записи информации при помощи модуля базы данных (DataStore). Также данный модуль при своей работе отслеживает актуальность данных, хранимых в локальной базе данных, и при необходимости запрашивает новые данные и сохраняет их. Вид данного модуля представлен на рисунке 2.20.

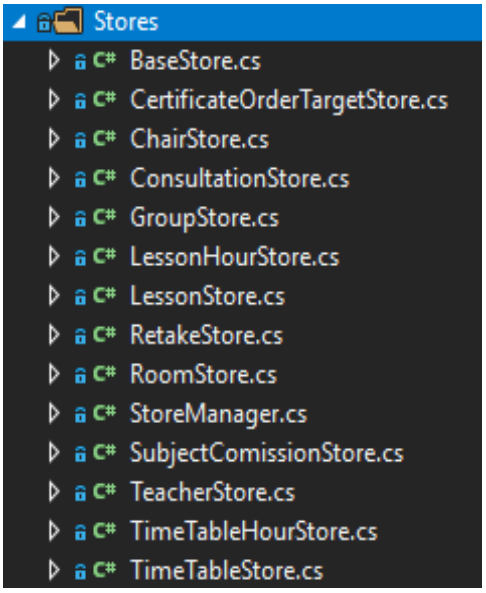

Рисунок 2.20 – Вид подмодуля осуществления связи между локальной базой данных и API

2.3.3 Компоненты пользовательского интерфейса

Интерфейс пользователя – коммуникационный канал, реализующий взаимодействие пользователя с программным обеспечением с помощью элементов управления.

Графический интерфейс пользователя – среда организации взаимодействия пользователя с вычислительной системой [20].

Стандартный графический интерфейс пользователя должен отвечать ряду требований:

– поддерживать информационную технологию работы пользователя с программным продуктом – содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки, расположенные в естественной последовательности использования;

– ориентироваться на конечного пользователя, который общается с программой на внешнем уровне взаимодействия;

– удовлетворять правилу «шести» – в одну линейку меню включать не более шести понятий, каждое из которых содержит не более шести операций;

– графические объекты сохраняют свое стандартизованное назначение и, по возможности, местоположение на экране.

Разработка пользовательского интерфейса – одна из самых сложных и ответственных задач проектирования. Это объясняется тем, что пользователя интересует в первую очередь удобство, эргономичность и наглядность.

Графический интерфейс Android-приложений в Xamarin строится на декларативном языке разметки XAML, на иерархии из элементов двух типов: View и ViewGroup [23].

Язык XAML упрощает создание пользовательского интерфейса для приложения .NET Framework. Он напрямую представляет создание экземпляров объектов в конкретном наборе резервных типов, определенных в сборках. В этом заключается его отличие от большинства других языков

разметки, которые, как правило, представляют собой интерпретируемые языки без прямой связи с системой резервных типов.

При представлении в виде текста файлы XAML являются XML-файлами, которые обычно имеют расширение .xaml. Файлы можно сохранять в любой кодировке, поддерживаемой XML, но обычно используется кодировка UTF-8.

Элементами типа View являются любые дочерние элементы пользовательского интерфейса, такие как кнопка, поле для ввода и т.д.

ViewGroup – это невидимый контейнер графических элементов, определяющий их размещение на экране. Это может быть вертикальный, горизонтальный список или таблицы элементов [15, с. 87-91].

Один ViewGroup может содержать множество View и другие ViewGroup.

Оперируя различными графическими элементами и контейнерами элементов можно создавать интерфейсы.

Иерархия View и ViewGroup представлена на рисунке 2.21.

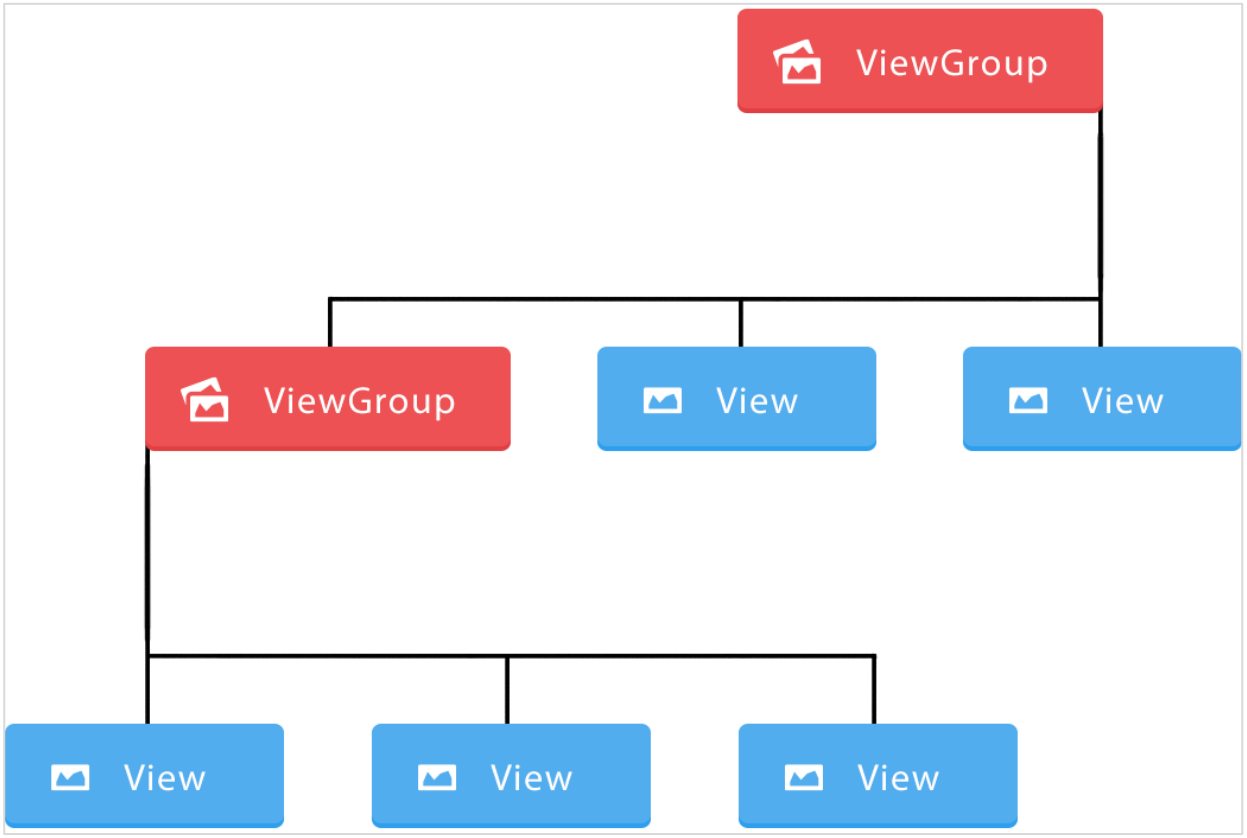

Рисунок 2.21 – Иерархия View и ViewGroup

Доступны следующие виды ViewGroup:

1. FrameLayout – самая простая разметка, прикрепляет каждое новое, дочернее представление к левому верхнему углу экрана, накладывая новый элемент на предыдущий, заслоняя его.

2. LinearLayout – помещает дочерние представления в горизонтальный или вертикальный ряд. Вертикальная разметка представляет собой колонку, а горизонтальная – строку с элементами. Данная разметка позволяет задавать не только размеры, но и «относительный вес» дочерних элементов, благодаря чему можно гибко контролировать их размещение на экране.

3. RelativeLayout – наиболее гибкий среди стандартных видов разметки.

4. Позволяет указывать позиции дочерних Представлений относительно границ свободного пространства и других представлений.

5. TableLayout – позволяет размещать дочерние представления внутри ячеек «сетки», состоящей из строк и столбцов. Размеры ячеек могут оставаться постоянными или автоматически растягиваться при необходимости.

6. GridLayout – отображает прокручиваемую сетку со строками и столбцами [4, с. 160-190].

Для создания пользовательского интерфейса в Android приложениях используется расширяемы язык разметки XML. Для создания окна или группы элементов необходимо создать отдельный файл с расширением .xaml.

Каждый созданный .xaml файл должен иметь хотя бы одну ViewGroup, которая будет описывать то, каким образом будут располагаться элементы на форме [23].

На рисунке 2.22 представлена стартовая форма приложения.

Она представляет собой окно с предложением для входа в аккаунт, либо пропуском этого шага.

Так же имеется возможность аутентификации пользователя через

браузер смартфона. При повторном открытии приложения, форма авторизации открываться не будет, а сразу перейдет на страницу новостей (или расписания, в зависимости от настроек приложения).

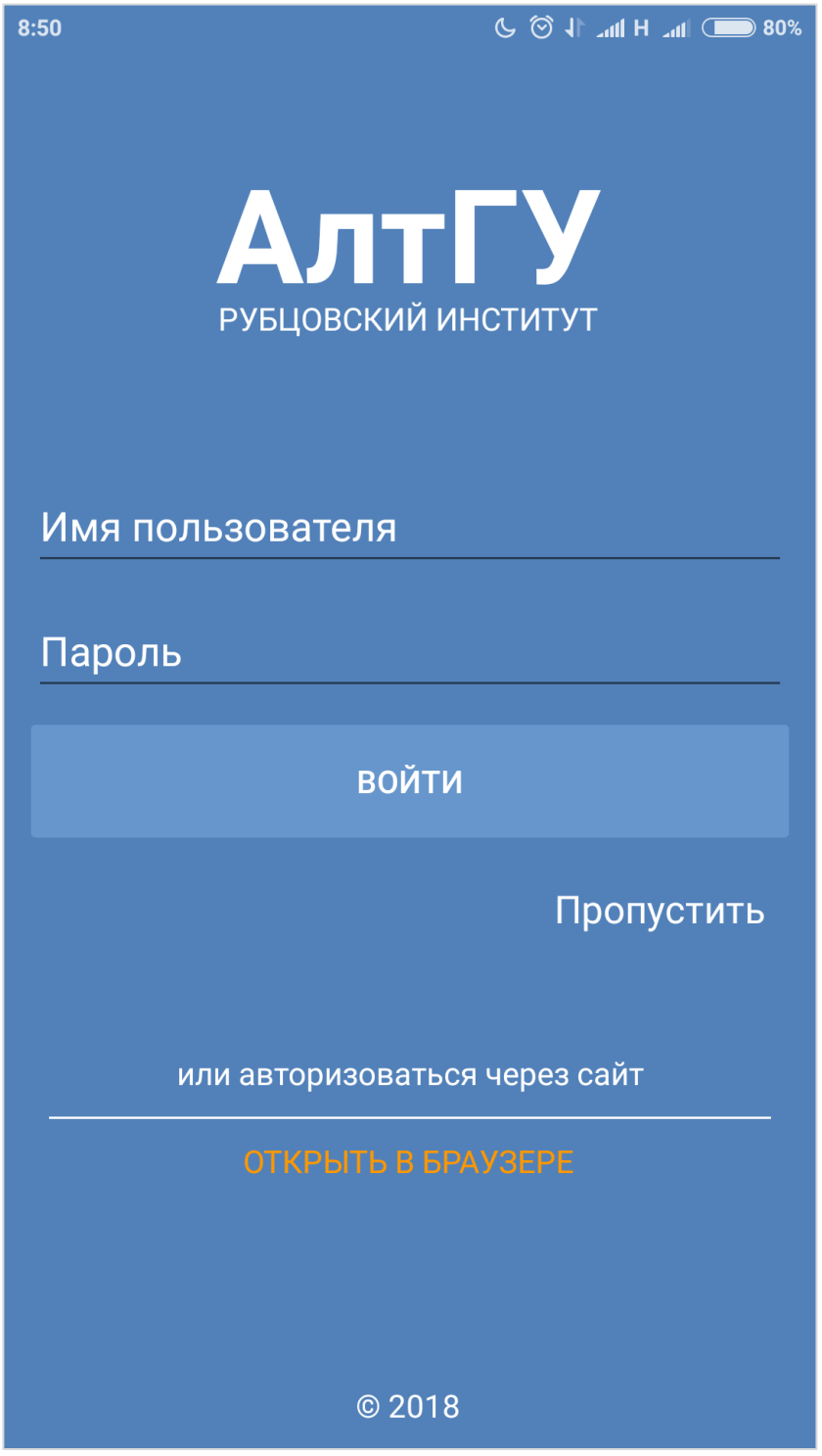

Рисунок 2.22 – Внешний вид стартовой формы приложения

После прохождения процедуры авторизации (пропуска), будет открыта новостная лента Рубцовского института (филиала) АлтГУ, на которой

предоставляется возможность перейти на сайт для подробного ознакомления с новостью, а так же возможность поделиться ссылкой на эту новость через социальные сети и другие сервисы. Вид данной формы представлен на рисунке 2.23.

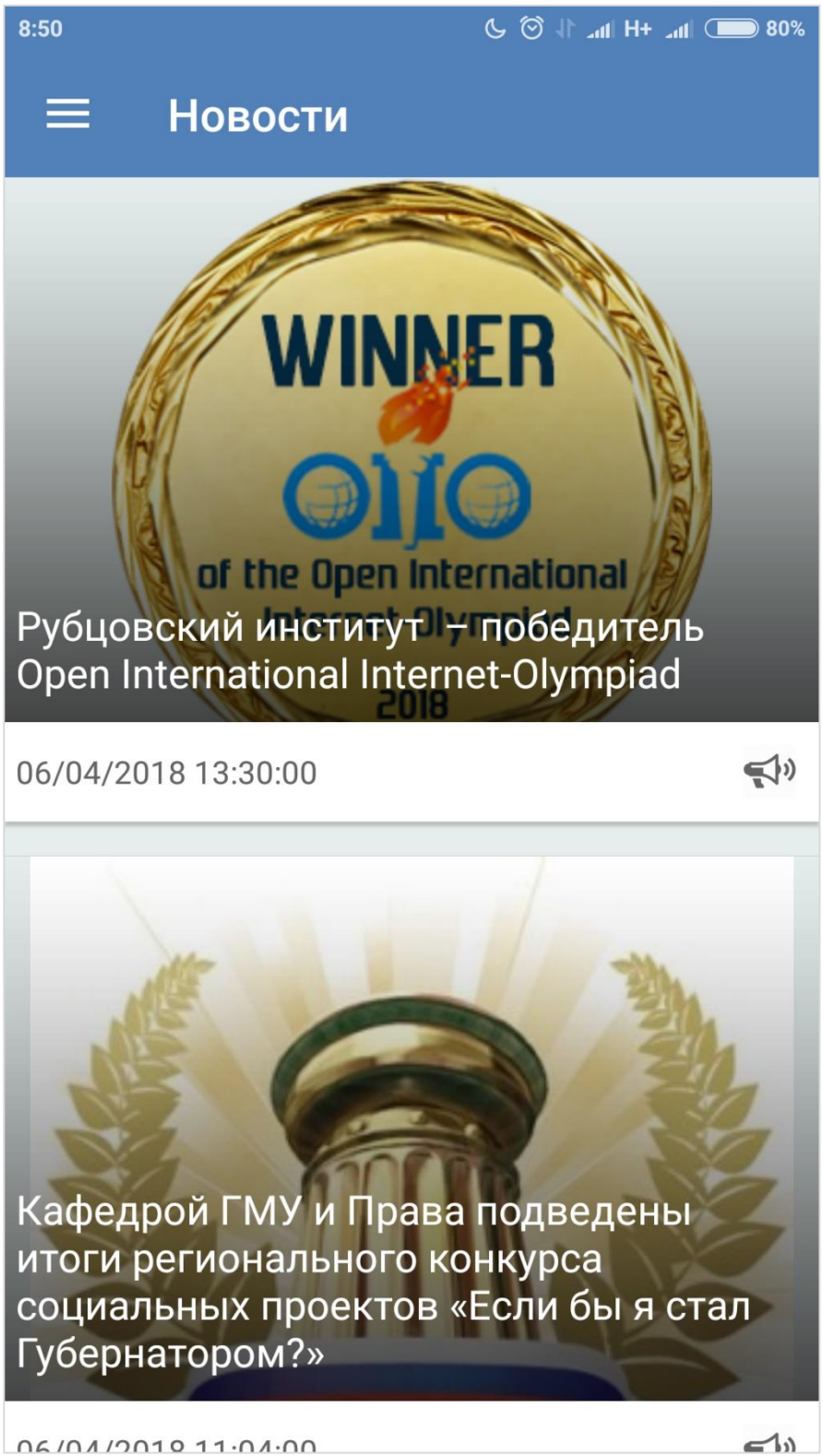

Рисунок 2.23 – Внешний вид формы новостной ленты приложения

В приложении имеется меню выбора необходимых пунктов. Данное меню реализовано последовательным списком, который находится слева от новостной ленты. Меню можно открыть, потянув от левого края вправо, либо нажав на кнопку меню в верхней левой части приложения. Пункты меню показаны на рисунке 2.24. В зависимости от того, авторизовался пользователь или нет, в меню будет отображено соответствующее верхнее меню. Для неавторизованного пользователя будет предложено авторизоваться в приложении, для доступа к личному кабинету и загрузки автоматического расписания на группу/преподавателя.

Для авторизованного пользователя будет отображена информация о фамилии и имени, номере группы, среднему баллу и количеству задолженностей.

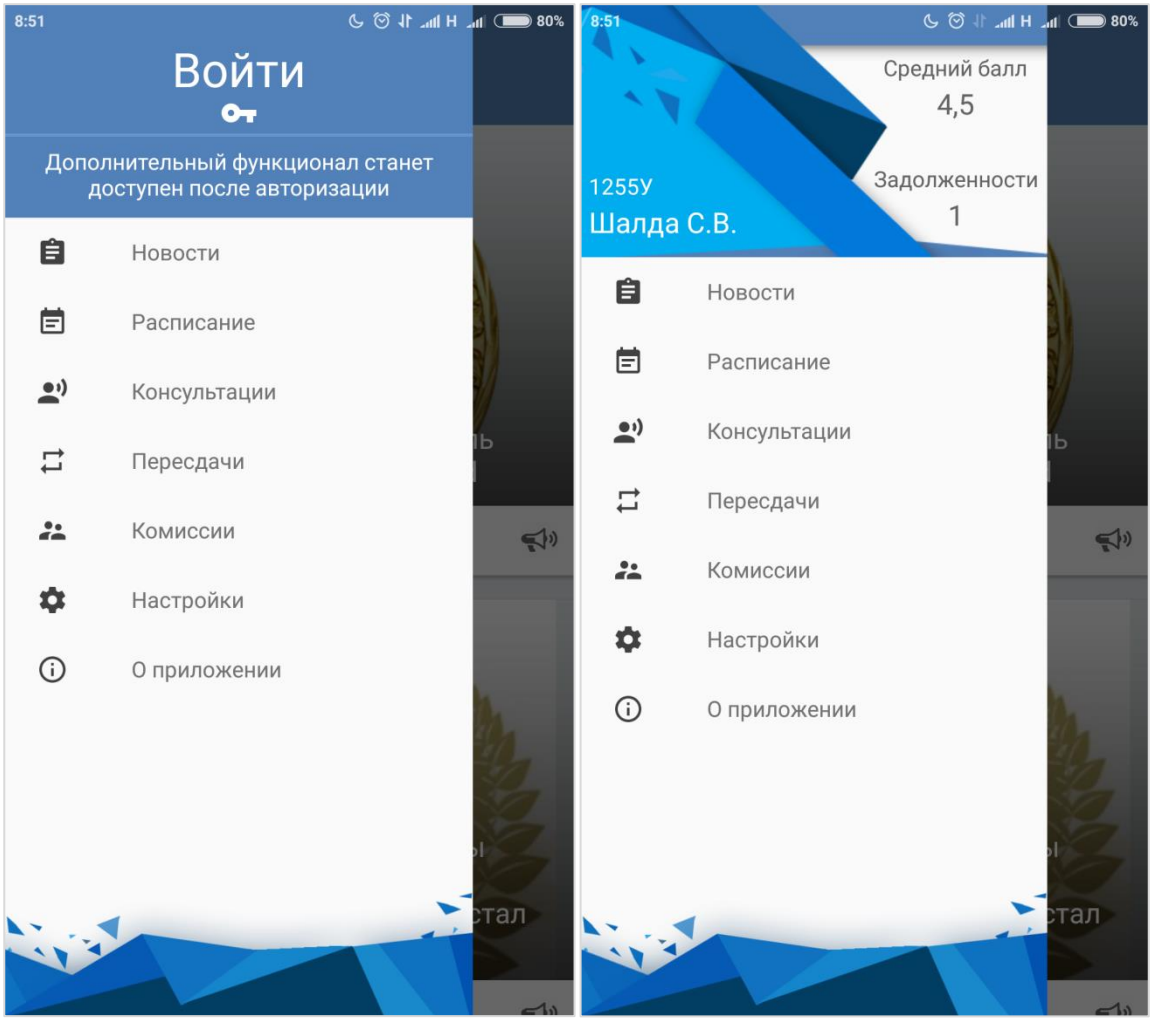

Рисунок 2.24 – Внешний бокового вид меню приложения для авторизованного и неавторизованного пользователя

У авторизованного пользователя появится ряд возможностей, с помощью которых он сможет узнать интересующую для себя информацию, а именно доступу к личному кабинету. В нем пользователь может узнать личную информацию для доступа к ресурсам Рубцовского института (филиала) АлтГУ, просмотреть данные о текущей успеваемости, а так же заказать нужную для обучаемого справку.

Для входа в личный кабинет следует после авторизации в приложении открыть боковое меню и нажать в любом месте на верхней информационной панели. Внешний вид вкладок «Информация», «Успеваемость» и «Заказ справок» в личном кабинете представлен на рисунках 2.25 и 2.26.

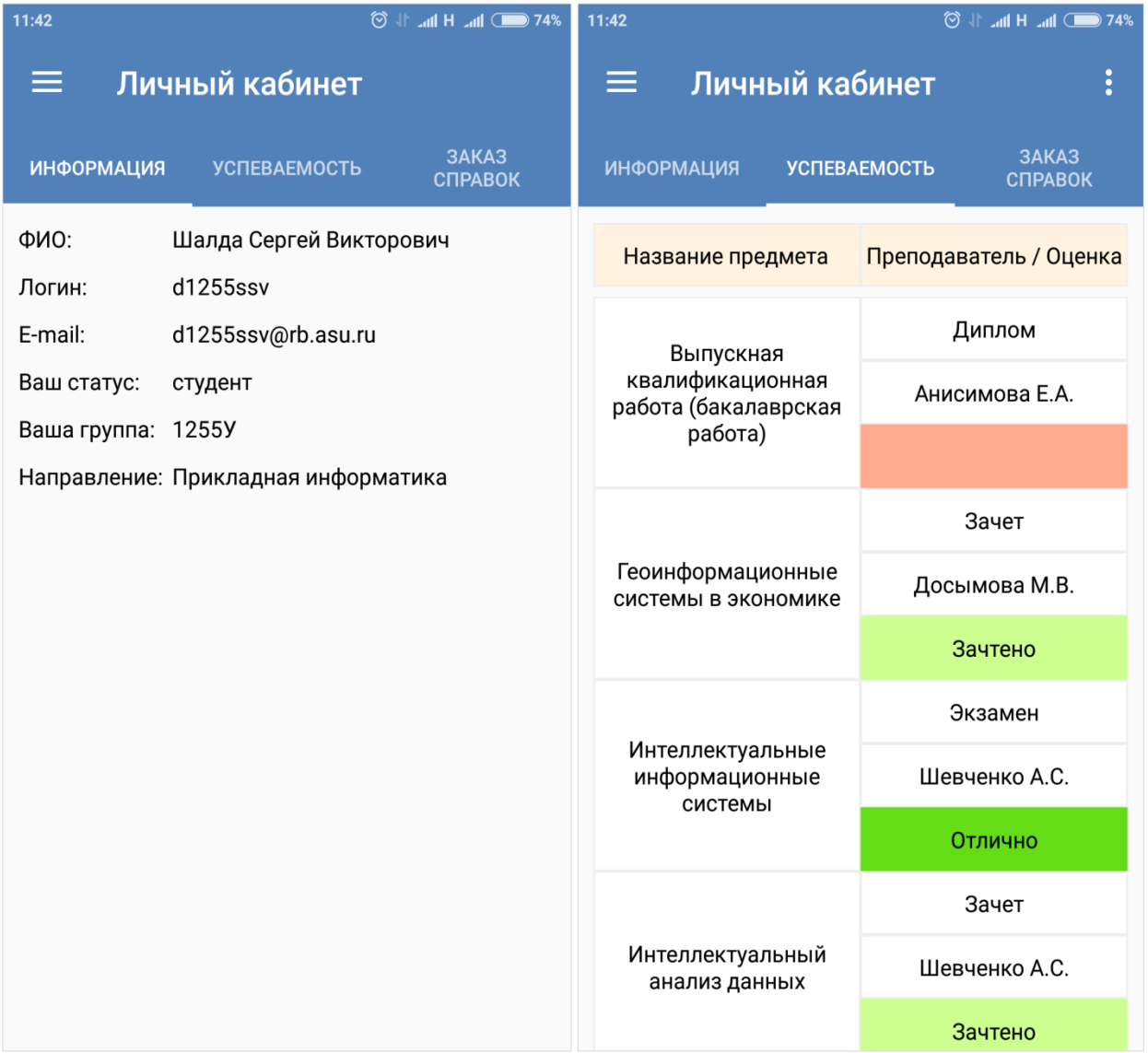

Рисунок 2.25 – Внешний вид вкладок «Информация» и «Успеваемость» в личном кабинете

На вкладке «Информация» показаны данные о студенте или преподавателе, а именно: ФИО, логин, e-mail, статус, и другие данные.

На вкладке «Успеваемость» для студента будут показаны данные об его академической успеваемости, а именно: список с названием изучаемой дисциплины, ФИО преподавателя и оценка. При нажатии на три точки в верхнем правом углу данной вкладки можно переключать сортировку списка, и видеть только задолженности.

На вкладке «Заказ справок» будет предложена возможность выбрать из предлагаемого списка места требования нужный пункт, после этого указать требуемое количество справок (не более 5), и нажать кнопку заказать для подачи заявки на получение справки.

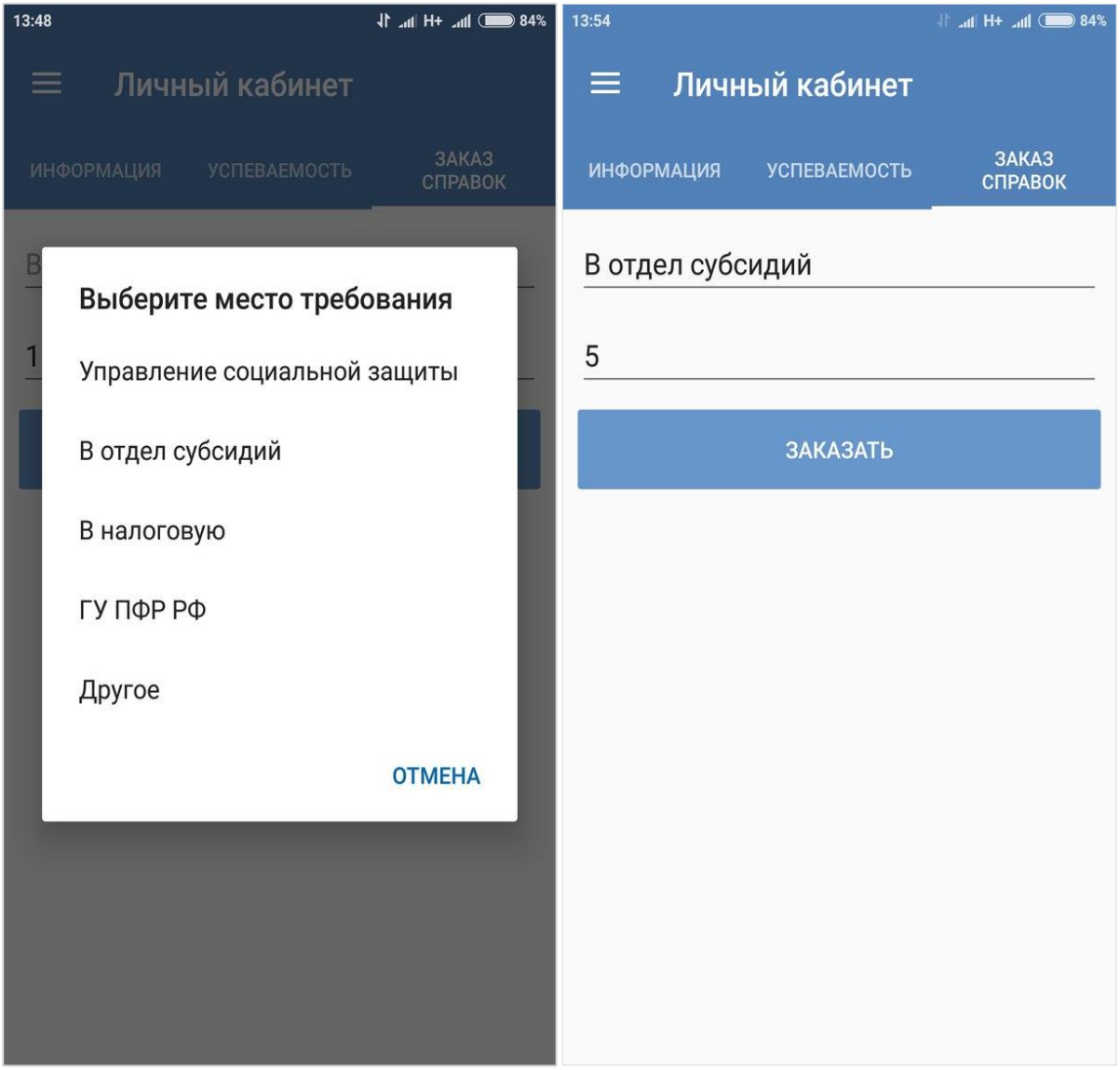

Рисунок 2.26 – Внешний вид вкладки «Заказ справок» в личном кабинете

При выборе пункта расписание откроется форма для отображения требуемого расписания. На ней можно выбрать тип получаемого расписания: группа, аудитория, преподаватель. Кроме этого на этой форме можно выбрать дату, на которую необходимо получить расписание.

Расписание будет автоматически загружать актуальное расписание на выбранную дату, и иметь вид ленты, то есть если пользователь прокрутит ленту вниз, то будет загружено расписание на еще одну неделю. Главной особенностью данной формы является то, что теперь можно добавлять сразу несколько независимых вкладок с требуемым расписанием. Для добавления новой вкладки с расписанием, внизу формы имеется удобная кнопка со знаком плюс, при нажатии на которую будет открыта новая вкладка.

Вид данной формы представлен на рисунке 2.27.

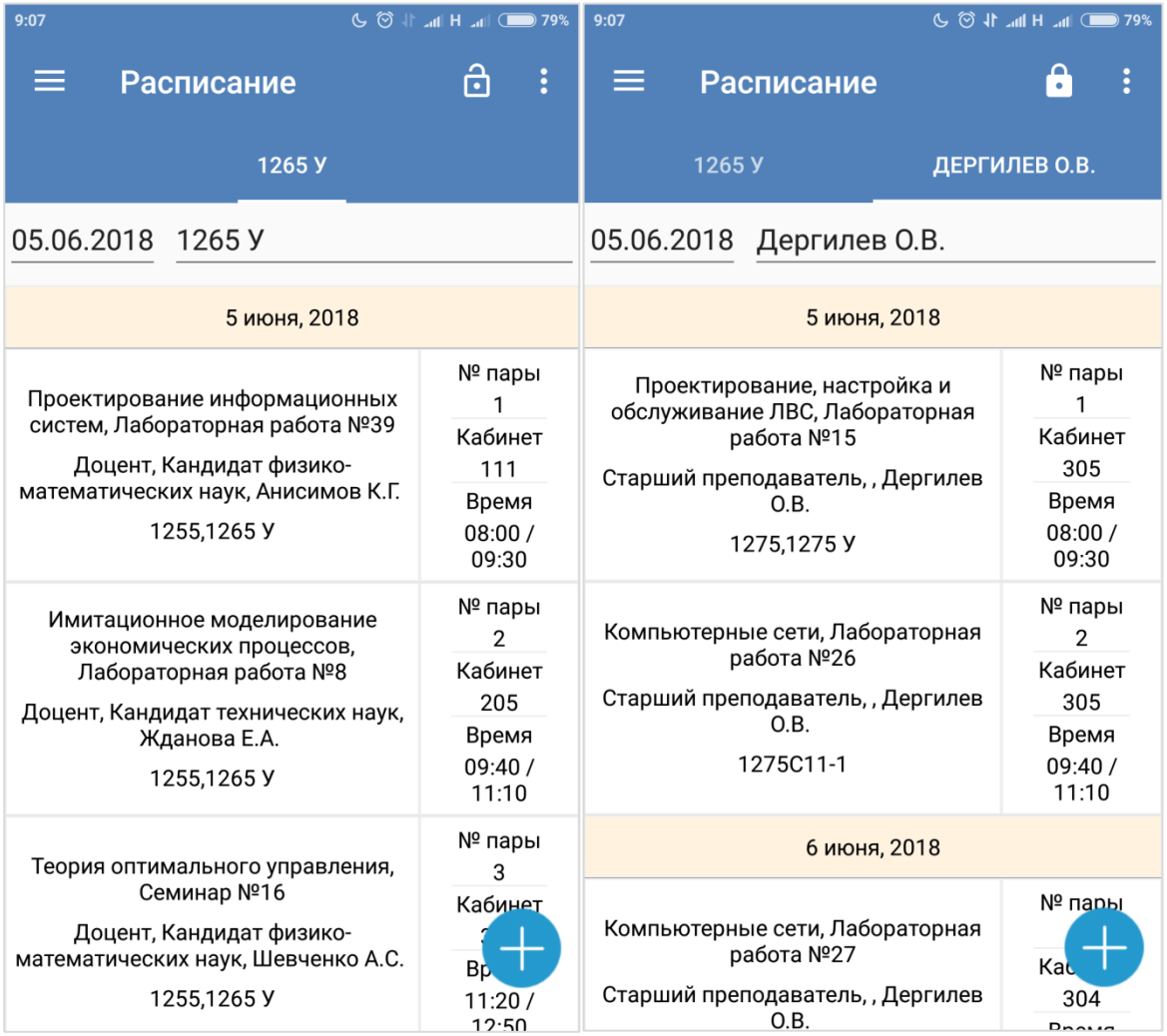

Рисунок 2.27 – Внешний вид формы списка расписания

При выборе пункта консультации откроется форма с отображением всех имеющихся на данный момент консультаций у преподавателей. Форма получения консультаций представлена в виде списка с фамилией, именем и отчеством преподавателя, датой консультации, номером аудитории и временем проведения.

Вид данной формы представлен на рисунке 2.28.

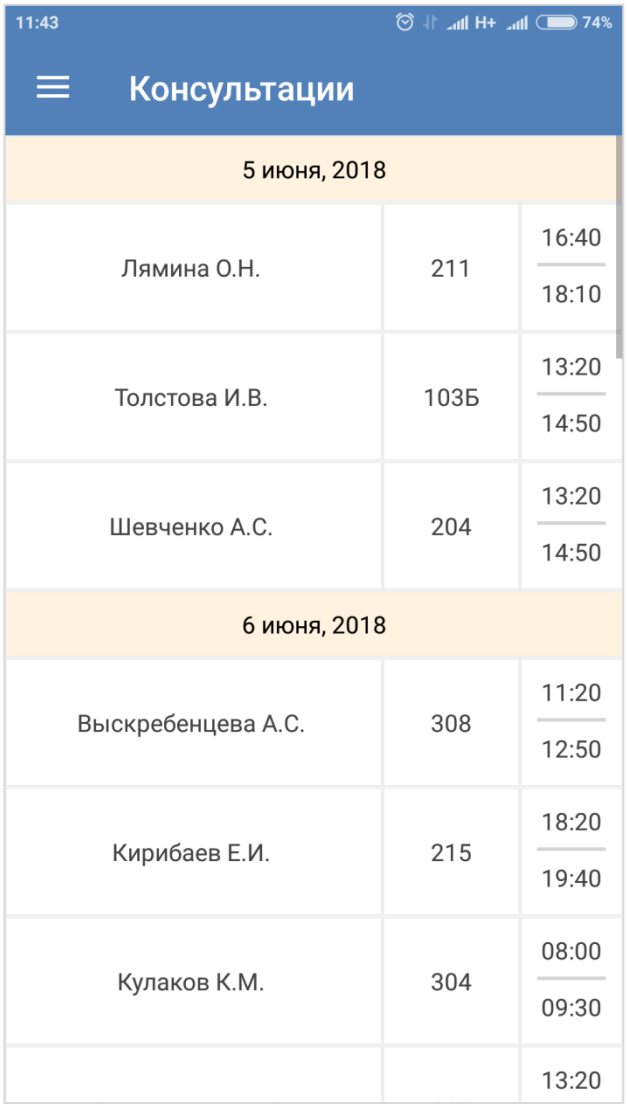

Рисунок 2.28 – Внешний вид формы списка консультаций

При выборе пункта пересдачи откроется форма с отображением всех имеющихся на данный момент пересдач.

Форма получения списка пересдач представлена в виде списка с фамилией, именем и отчеством преподавателя, датой пересдачи, номером аудитории и временем проведения. Вид данной формы представлен на

рисунке 2.29. При выборе пункта комиссии откроется форма с возможностью получения списка предметных комиссий по требуемой кафедре.

Данная форма представляет собой окно с возможностью выбора необходимой кафедры и получением списка с преподавателями, номером аудитории со временем и датой проведения.

Вид данной формы представлен на рисунке 2.30.

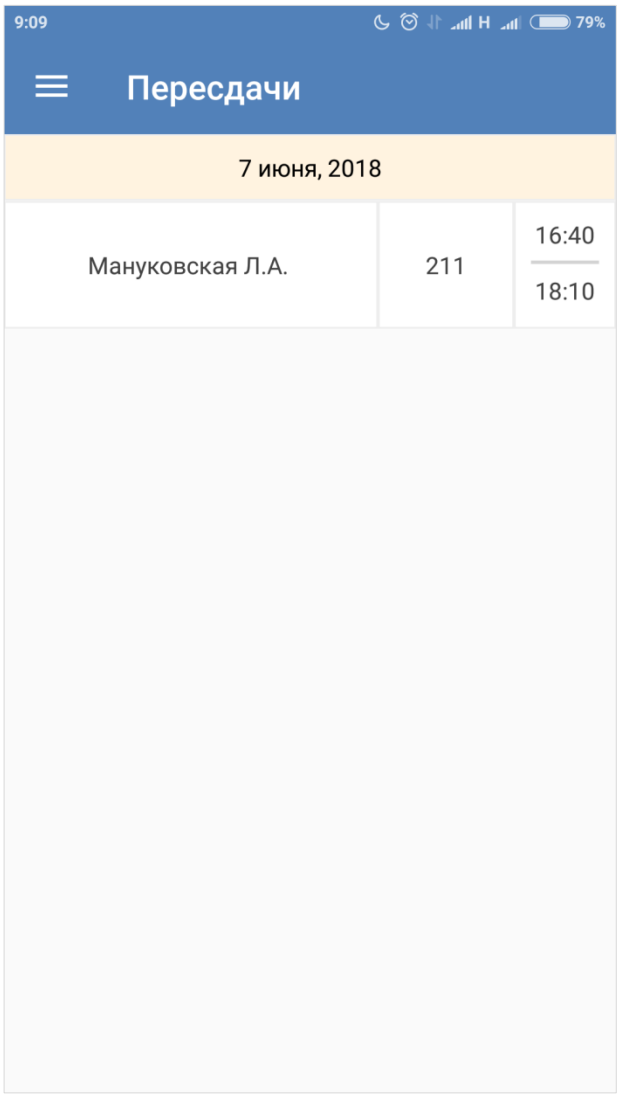

Рисунок 2.29 – Внешний вид формы получения списка пересдач

Настройки приложения можно запустить, нажав на пункт «Настройки» в боковом меню. На данной форме илжно отключить уведомления об обновлении расписания, задать начальную страницу при запуске приложения, а так же возможность выйти из аккаунта авторизированному пользователю. Вид данной формы представлен на рисунке 2.31.

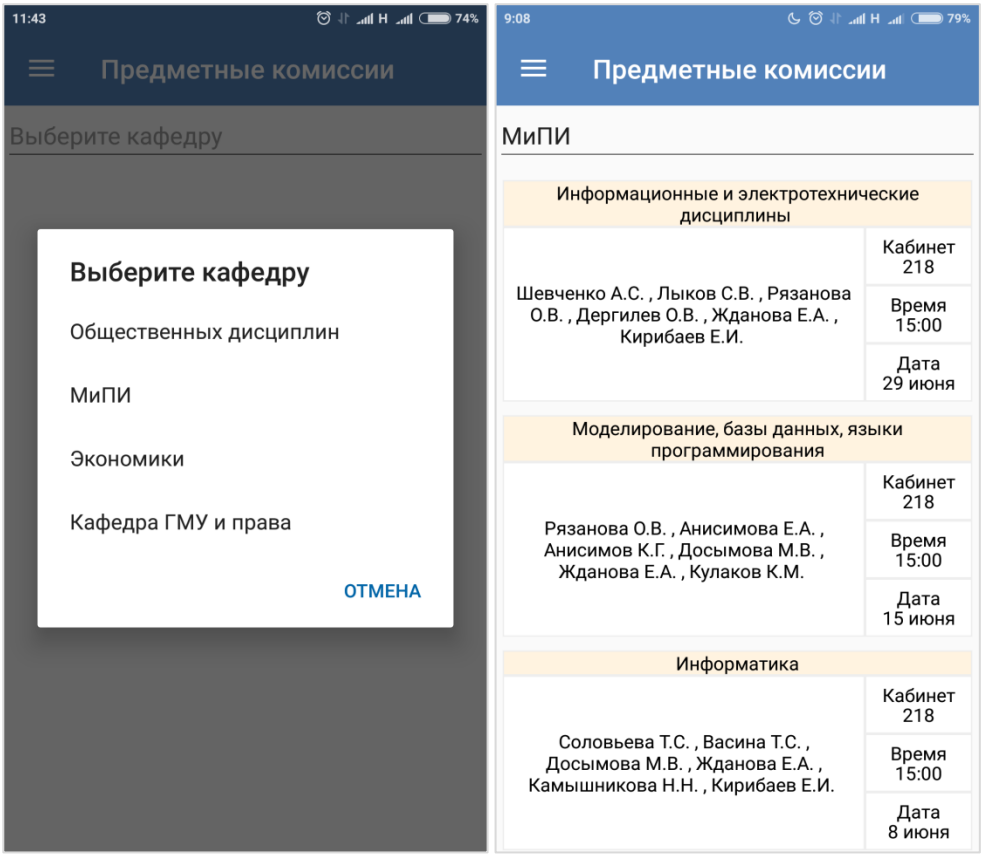

Рисунок 2.30 – Внешний вид формы получения списка предметных комиссий

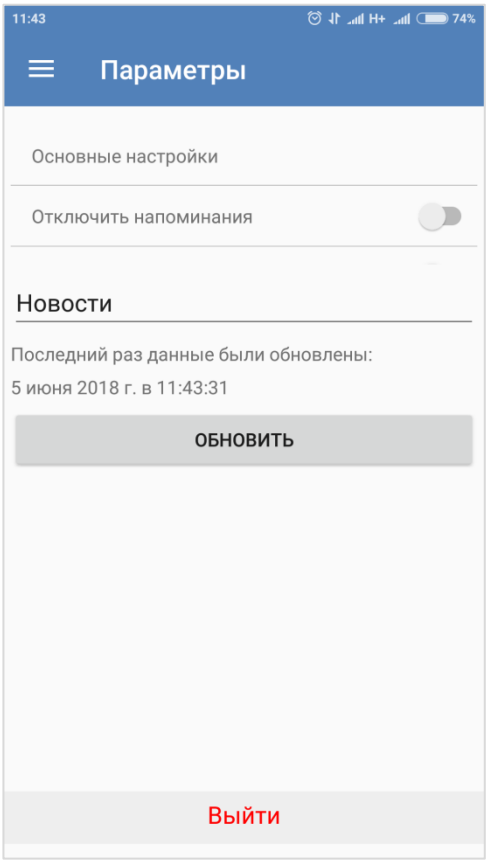

Рисунок 2.31 – Внешний вид формы настроек приложения

Еще одним пунктом меню является пункт «О приложении». При выборе этого пункта откроется форма с данными о разработчике и активными тестерами приложения. При двойном нажатии на надпись о версии, откроется окно со списком всех изменений, сделанных в приложении. Вид данной формы представлен на рисунке 2.32.

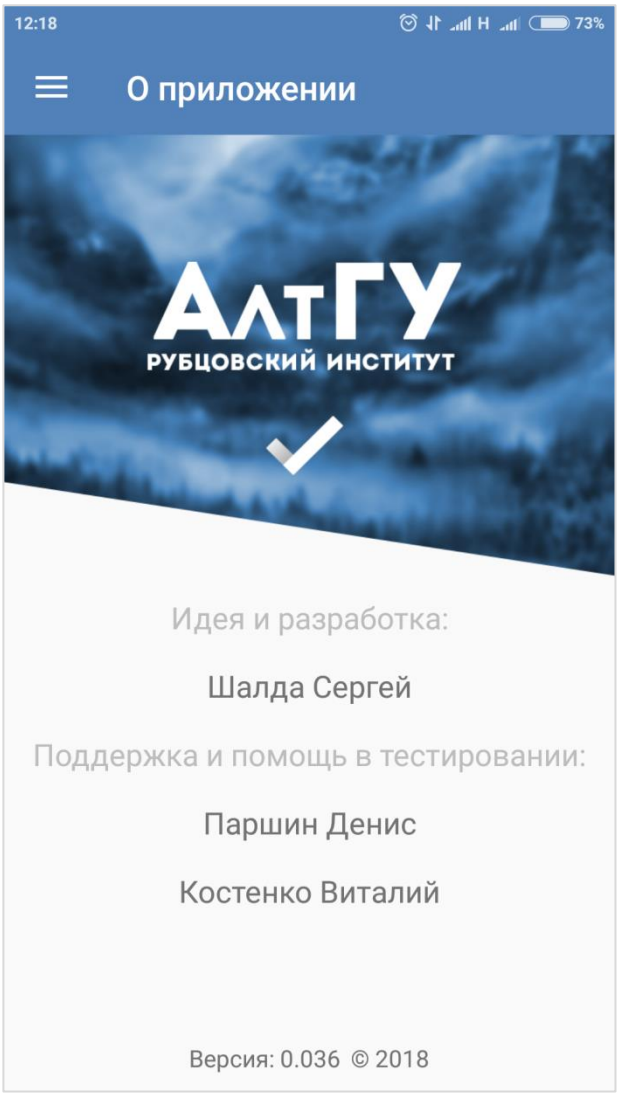

Рисунок 2.32 – Внешний вид формы «О приложении»

2.4 Компьютерно-сетевое обеспечение

Мобильное приложение имеет архитектуру клиент-сервер. Серверная часть приложения использует существующее сетевое оборудование Рубцовского института (филиала) АлтГУ без каких-либо изменений. Структура сети также не подвергается изменениям, так как её

характеристики полностью удовлетворяют предъявленным требованиям. Используется следующее оборудование:

1. Сервер WWW. Выполняет функции базового WEB-сервера со следующим набором сервисов: HTTP, FTP, DNS, MAIL, PROXY, APACHE, MySQL. Технические характеристики: 2xP4Xeon, 3,0GHz, 4x2Gb, RAID1 SATA 2x500Gb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

2. Сервер ORACLE. Выполняет функции сервера БД для хранения информации для портала и информации необходимой для учебного процесса. Технические характеристики: 2xP4Xeon, 2,8GHz, 24Gb, RAID5 SATA 4x200Gb, 2xGigabit Ethernet. Установленное ПО – Gentoo Linux Server Edition.

2.5 Обеспечение информационной безопасности

Информационная безопасность (ИБ) – состояние защищённости информационной среды общества, обеспечивающее её формирование, использование и развитие в интересах отдельных граждан, организаций и государства.

Персональные данные – любая информация, относящаяся к определенному или определяемому на основании такой информации физическому лицу (субъекту персональных данных), в том числе его фамилия, имя, отчество, год, месяц, дата и место рождения, адрес, семейное, социальное, имущественное положение, образование, профессия, доходы, другая информация. Информационная система персональных данных – информационная система, представляющая собой совокупность персональных данных, содержащихся в базе данных, а также информационных технологий и технических средств, позволяющих осуществлять обработку таких персональных данных с использованием средств автоматизации или без использования таких средств [2, с. 100].

Классификация информационной системы персональных данных

Определение категории обрабатываемых персональных данных:

– первая категория – персональные данные, касающиеся расовой, национальной принадлежности, политических взглядов, религиозных и философских убеждений, состояния здоровья, интимной жизни;

– вторая категория – персональные данные, позволяющие идентифицировать субъекта персональных данных и получить о нём дополнительную информацию, за исключением персональных данных, относящихся к категории 1;

– третья категория – персональные данные, позволяющие идентифицировать субъекта персональных данных;

– четвертая категория – обезличенные и (или) общедоступные персональные данные.

По результатам анализа исходных данных ИС присваивается один из следующих классов:

– класс 1 – информационные системы, для которых нарушение заданной характеристики безопасности персональных данных, обрабатываемых в них, может привести к значительным негативным последствиям для субъектов персональных данных;

– класс 2 – информационные системы, для которых нарушение заданной характеристики безопасности персональных данных, обрабатываемых в них, может привести к негативным последствиям для субъектов персональных данных;

– класс 3 – информационные системы, для которых нарушение заданной характеристики безопасности персональных данных, обрабатываемых в них, может привести к незначительным негативным последствиям для субъектов персональных данных;

– класс 4 – информационные системы, для которых нарушение заданной характеристики безопасности персональных данных,

обрабатываемых в них, не приводит к негативным последствиям для субъектов персональных данных.

Для защиты от постороннего вторжения предусматриваются определенные меры безопасности. В ИС это осуществляется программными средствами, которые выполняют следящие функции:

– идентификация субъектов и объектов;

– разграничение доступа к ресурсам и информации;

– контроль и регистрация действий [8, с. 114-119].

Процедура идентификации и подтверждения подлинности предполагает проверку, является ли субъект, осуществляющий доступ, или объект, к которому осуществляется доступ, тем за кого себя выдает. Здесь используются следующие методы:

– простые, сложные или одноразовые пароли;

– средства анализа индивидуальных характеристик субъекта (геометрических данных);

– ключи, жетоны, магнитные карты;

– обмен вопросами и ответами с администратором системы;

– специальные идентификаторы и контрольные суммы для программ и данных.

После процедуры идентификации пользователь получает доступ к системе, где защита от несанкционированного доступа реализуется на трех уровнях:

– на уровне аппаратуры;

– на уровне программного обеспечения;

– на уровне данных.

Защита информации на 1 и 2 уровнях предусматривает управление доступом к различным вычислительным ресурсам (отдельным устройствам, операционной системы, служебным или личным приложениям пользователя).

Защита информации на уровне данных направлена на защиту информации в процессе обращения к ней, в процессе работы с файловой

структурой, в процессе передачи информации по каналам связи.

В общем, комплекс программно-технических средств и организованных решений по защите информации от несанкционированного доступа характеризуется следующими действиями:

- логическое управление доступом;
- регистрацией, контролем и учетом работы;
- применением криптографических средств;
- обеспечением целостности информации.

Выделяют следующие формы контроля и управления доступом:

- предотвращение доступа к жесткому диску, каталогам и файлам;
- установка привилегий к группам файлов;
- защита от модификаций и изменений;
- защита от уничтожения;
- предотвращение копирования.

Под средствами защиты от копирования понимается такие средства, которые обеспечивают выполнение программой своих функций только при опознании некоторого уникального не копированного элемента. Таким элементом может быть ключевой сменный носитель, часть оборудования или специальное устройство, подключаемое к используемому устройству [2, с. 233]. Защита от копирования реализуется выполнением ряда функций:

- идентификация среды, из которой запускается система;
- проверка подлинности среды, из которой произошел запуск;
- немедленная реакция на запуск из несанкционированной среды;
- обязательная регистрация санкционированного копирования;
- противодействие изучению алгоритмов работы системы.

Область информационной безопасности является компетенцией отдела программного и технического обеспечения Рубцовского института (филиала) АлтГУ. Данным отделом регулярно производятся плановые профилактические работы всех программно-аппаратных систем Института.

С недавнего времени в Рубцовском институте (филиале) АлтГУ был изменен подход к осуществлению авторизации пользователей, а именно была реализована авторизация по протоколу OAuth 2.0.

OAuth 2.0 – это непосредственно сервер, который производит выдачу разрешений на доступ к ресурсам (токены).

При использовании OAuth-авторизации пользователь не передает свой логин и пароль к защищенным ресурсам напрямую в приложение, поэтому у пользователя больше оснований доверять приложению, поскольку пользователь может быть уверен, что несанкционированный доступ к его личным данным невозможен. Не владея логином и паролем пользователя, приложение сможет выполнять только те действия с данными, которые разрешил пользователь, и никакие другие, так же при разработке приложения не нужно заботиться об обеспечении конфиденциальности логина и пароля пользователя. Логин и пароль не передаются приложению, а, следовательно, они не могут попасть в руки злоумышленников [24].

На рисунке 2.33 представлена схема получения разрешений приложения, для доступа к ресурсам сервера.

Главной особенностью схемы авторизации OAuth 2.0 является то, что клиентское приложение не имеет прямого доступа к авторизационным и идентификационным данным пользователя, а сам процесс авторизации происходит на сервере авторизации Рубцовского института (филиала) АлтГУ.

Сам процесс обмена данными и идентификации пользователя в клиентском приложении происходит при помощи токенов и ключей приложения ACCEESSTOKEN, RESOURCETOKEN и REFRESHTOKEN, а так же ключей PUBLICKEY и SECRETKEY, выданных при регистрации приложения. Процесс авторизации пользователя и получения данных имеет несколько этапов, а именно:

1. Инициализации процесса авторизации с посыла публичного ключа приложения к OAUTH-SERVER по определенному URL.

2. Пользователь загружается стандартная форма авторизации на ресурсах поставщика данных.

3. После ввода авторотационных и идентификационных данных пользователя, OAUTH-SERVER производит его авторизацию, и в случае успешного исхода авторизует пользователя и создает ACCESSTOKEN для клиентского приложения запросившего авторизацию. В случае не авторизации пользователя формируется ответ с ошибкой авторизации пользователя.

4. При успешной авторизации пользователя клиентскому приложению передается выданный ACCESSTOKEN и URL для получения RESOURCETOKEN и REFRESHTOKEN на следующем этапе. В случае неудачи авторизации пользователя, клиентскому приложения пересылается данные об ошибке сформированные на предыдущем этапе.

5. Для получения RESOURCETOKEN и REFRESHTOKEN клиентское приложение отправляет свой SECRETTOKEN и полученный ACCESSTOKEN на данное соединение.

6. OAUTH-SERVER после авторизации клиентского приложения по полученным данным отправляет REFRESHTOKEN (это автоматическая замена пере-аутентификации пользователя, которая не требует лишний раз вводить логин и пароль) и RESOURCETOKEN для последующего обмена данными. В случае если произошла ошибка авторизуемого соединения, формируется объект ошибки и отправляется клиентскому приложению.

Так же имеются возможности, которые могут потребоваться в зависимости от различных ситуаций, а именно:

1. Обновление RESOURCETOKEN. RESOURCETOKEN имеет определенный период существования, при окончании которого токен устаревает и для дальнейшего обмена данными пользователю придется пройти процедуру авторизации заново. Для устранения этого недостатка на 6 этапе нам передается REFRESHTOKEN при помощи которого мы может обновить токен ресурсов.

Процесс обновления выглядит следующим образом: на стороне клиентского приложения срабатывает событие об окончании времени жизни RESOURCETOKEN. Клиентское приложение посылает запрос на обновление токен ресурса, с выданным ему ранее REFRESHTOKEN.

Сервер авторизации идентифицирует REFRESHTOKEN и в случае его успеха, генерирует новый RESOURCETOKEN и пересылает его обратно приложению. В случае неудачи, формируется объект ошибки и отправляется клиентскому приложению.

2. Обновление REFRESHTOKEN. REFRESHTOKEN так же имеет определенный период существования, при окончании которого токен устаревает и при помощи него невозможно получить новый RESOURCETOKEN.

Для получения нового REFRESHTOKEN OAUTH-SERVER пересылается старый REFRESHTOKEN и дополнительные параметры. В случае успеха клиентскому приложению выдается новый REFRESHTOKEN, старый при этом перестает быть действительным.

В случае неудачи, формируется объект ошибки и отправляется клиентскому приложению.

3. Обмен данными. Клиентское приложение делает запрос по определенному URL передавая дополнительные параметры и выданный ранее RESOURCETOKEN. RESOURCE_SERVER проверяет действительность RESOURCETOKEN и в случае успеха формирует объект данных или объект ошибки запроса данных.

В случае недействительности RESOURCETOKEN формируется объект с сообщением об ошибки авторизации клиентского приложения.

В Рубцовском институте (филиале) АлтГУ доступ к информационным системам управления вузом осуществляется на основании приказа директора «Об оптимизации доступа к БД в целях защиты информации», в котором указаны лица, имеющие полный доступ на внесение и изменение

информации, и лица, имеющие доступ для чтения информации, в случае ее использования для принятия решения.

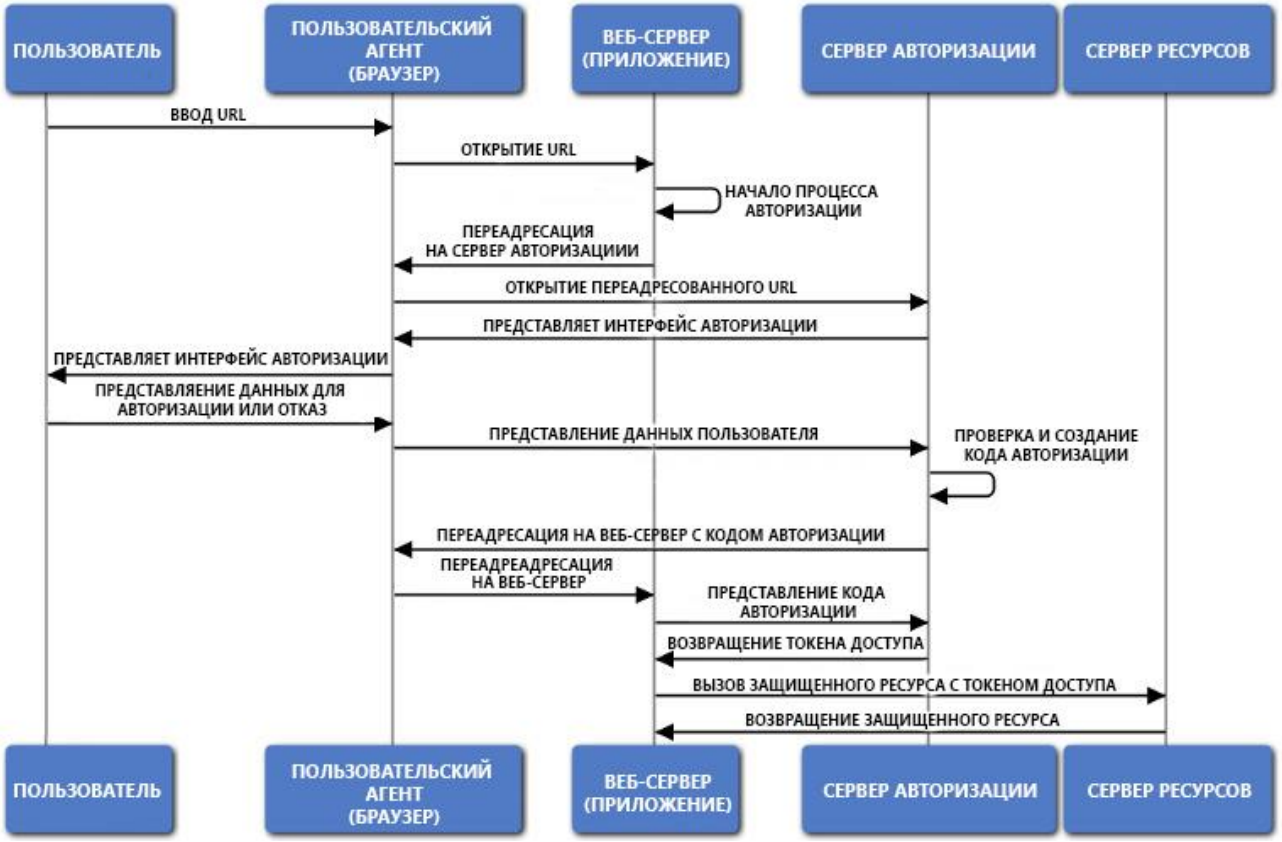

Рисунок 2.33 – Процесс авторизации по протоколу OAuth 2.0

Полный доступ на изменение информации имеют только сотрудники отдела по работе со студентами. У методистов кафедр, заведующих кафедрами, директора и его заместителей имеется доступ на чтение информации [24].

В приложении реализована авторизация по протоколу OAuth 2.0., что подразумевает под собой безопасный доступ и защиту от несанкционированного доступа к личной и конфиденциальной информации пользователя, который будет проходить этап авторизации в приложении.

Общедоступная информация будет доступна любому пользователю приложения.

3 Оценка эффективности от внедрения мобильного приложения

3.1 Общие положения

Эффективность процессов характеризуется системой показателей, отражающих соотношение их затрат и результатов. Эффективность процесса тем выше, чем выше результаты и ниже приложенные усилия.

Эффективность мобильного приложения – это свойство способности выполнять поставленную цель в заданных условиях использования и с определенным качеством. Эта характеристика отражает:

– действенность, т.е. степень соответствия приложения своему назначению (прагматическая эффективность);

– техническое совершенство приложения (техническая эффективность);

– простоту, технологичность разработки и создание приложения (технологическая эффективность);

– удобство использования и обслуживания приложения (эксплуатационная эффективность);

– улучшение и облегчение условий труда, изменение его содержания, развитие творческих функций, способностей и потребностей людей, преодоление существенных различий в труде и др. (социальная эффективность);

– экономическую целесообразность внедрения приложения, т.е. целесообразность произведенных на создание и функционирование приложения (экономическая эффективность).

Понятие эффективности связано с получением некоторого полезного результата – эффекта использования.

В соответствии с ГОСТ Р ИСО 9000-2001, эффективность функционирования приложения определяется соотношением результата (эффекта) и затраченных ресурсов. Приведенной оценкой затрат ресурсов выступает их стоимость. Затраты на функционирование ИС состоят из:

– стоимости приобретения программной платформы;

– стоимости доработки;

– стоимости внедрения;

– стоимости системного и вспомогательного программного обеспечения;

– стоимости аппаратного и сетевого обеспечения приложения;

– количества циклов (лет) эксплуатации;

– стоимости эксплуатации [1, с. 96-114].

Основные задачи, стоящие при создании приложения – минимизация стоимости и обеспечение требуемого качества. Качество – это совокупность свойств системы, обусловливающих возможность ее использования для удовлетворения определенных потребностей пользователей в соответствии с ее назначением.

Основными показателями качества мобильного приложения являются:

– надежность;

– достоверность;

– безопасность.

Надежность – свойство приложения сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных условиях применения.

Надежность приложения является средством обеспечения актуальной и достоверной информации на выходе.

Достоверность функционирования – свойство приложения, обусловливающее безошибочность производимых ею преобразований информации.

Достоверность функционирования приложения полностью определяется и измеряется достоверностью ее результатной информации.

Безопасность – свойство приложения, заключающееся в способности обеспечить конфиденциальность и целостность информации, т.е. защиту информации от несанкционированного доступа. В любой сфере человеческой деятельности оценка эффективности внедрения любой новой техники, технологий и приложений осуществляется с помощью множества показателей. К ним относятся показатели прагматической, технической, эксплуатационной, социальной и экономической эффективности [11, с. 66].

3.2 Показатели эффективности

В качестве показателей прагматической эффективности для мобильного приложения могут выступать:

– показатели безопасности информационной системы (защита сервера от различных атак, уменьшение уязвимости приложения, за счет контроля вводимой пользователем информации);

– показатели оперативности (использование созданного приложения позволяет снизить затраты ресурсов на доступ к информации, значительно снижены временные затраты на поиск необходимой информации).

Показатели технической эффективности должны оценивать техническое совершенство приложения, научно-технический уровень организации и функционирования.

Разработанное приложение:

– работает по архитектуре клиент-сервер;

– серверная часть базируется на отработанном и проверенном годами реляционном принципе построения баз данных.

Показатели эксплуатационной эффективности:

– показатели надежности – обрабатываемые данные зависят от надежности устройства и спроектированного приложения;

– функциональные возможности – появилась возможность использования портала с помощью мобильных устройств;

– технология обслуживания – система не требует вмешательства программиста во время работы.

Среди показателей социальной эффективности можно выделить предоставление постоянного и удобного доступа к необходимому расписанию, доступу к личному кабинету, а также свежим новостям института, что способствует повышению престижности ВУЗа и его конкурентоспособности в современной обстановке.

Обобщающими показателями эффективности любого мобильного приложения являются показатели экономической эффективности. Часто прибыль определяется путем экспертной оценки или по аналогии с другими подобными системами. Для оценки эффективности могут использоваться две группы показателей: интегральные традиционные показатели и частные показатели.

Обычно в качестве экономических показателей используются:

– годовой экономический эффект;

– коэффициент экономической эффективности капитальных вложений;

– срок окупаемости капитальных вложений;

– трудоемкость обработки информации;

– эксплуатационная стоимость затрат;

– расчет текущих затрат пользователя;

– экономия текущих затрат при автоматизации;

– годовая экономия затрат на материалы.

Экономический эффект – это результат внедрения какого-либо мероприятия, выраженный в стоимостной форме в виде экономии от его осуществления.

Основными источниками экономии являются:

– улучшение показателей и их основной деятельности, происходящих в результате использования спроектированного продукта;

– повышение технического уровня, качества и объёмов вы числительных работ;

– увеличение объёмов и сокращение сроков переработки информации;

– повышение коэффициента использования вычислительных ресурсов, а так же средств подготовки и передачи информации;

– уменьшение численности персонала, занятого обработкой исходных данных, переработкой и получением необходимой информации;

– снижение затрат на эксплуатационные материалы.

Предварительный экономический эффект рассчитывается до выполнения разработки на основе данных технических предложений и прогноза использования. Предварительный эффект является элементом технико-экономического обоснования (ТЭО) разработки проекта. Потенциальный экономический эффект рассчитывается по окончании разработки на основе достигнутых технико-экономических характеристик и прогнозных данных о максимальных объёмах использования программного изделия.

Коэффициент экономической эффективности капитальных вложений показывает величину годового прироста прибыли, образующуюся в результате производства или эксплуатации программного изделия на один рубль капитальных единовременных вложений.

Срок окупаемости (величина, обратная коэффициенту эффективности) – показатель эффективности использования капиталовложений, представляет собой период времени, в течение которого произведённые затраты на реализацию программного продукта окупаются полученным эффектом.

Для оценки экономической эффективности внедрения мобильного приложения можно использовать систему частных показателей. Частные

показатели необходимы для оценки частного экономического эффекта, получаемого по отдельным источникам экономии.

Например, частные показатели в сфере оказания образовательных услуг:

– сокращение доли неквалифицированного и ручного труда;
– сокращение материальных затрат;
– сокращение инвестиций в рекламный бюджет;
– снижение уровня «бумажного» документооборота;
– сокращение времени на принятие решения звене управления.

3.3 Расчет экономической эффективности

Экономическая эффективность учитывает затраты и результаты реализации проекта, выходящие за пределы прямых финансовых интересов его участников.

Экономическая эффективность позволяет судить о необходимости внедрения программного продукта. В основе исчисления экономической эффективности лежит сопоставление существующего реально метода обработки данных (базовый вариант) и внедряемого метода обработки (проектный вариант), при этом обязательно проводится анализ затрат, необходимых для выполнения всех операций, сопутствующих внедрению нового метода обработки данных [5, с. 82].

К таким затратам относятся затраты на разработку, реализацию, внедрение и эксплуатацию программного продукта.

Выбор базы для сравнения зависит от цели расчета эффективности, от того, что требуется определить: ожидаемую, а также фактическую эффективность в конкретных условиях применения вычислительной техники или наиболее выгодный способ обработки данных. В первом случае за базу для сравнения следует принять способ выполнения работ, существующий в

конкретных условиях до применения данной вычислительной техники, во втором случае – предлагаемый лучший способ обработки данных.

Особенностью расчетов сравнительной эффективности автоматизированной обработки данных является то, что в отдельных случаях базовый вариант может отсутствовать. Весь эффект определяется сопоставлением экономии от использования информации с затратами на ее получение.

В данном дипломном проекте в качестве базового варианта выступает действующее мобильное приложение «РИ АлтГУ – Расписание». В качестве предлагаемого варианта используется созданное в дипломном проекте мобильное приложение для получения доступа к ресурсам информационно-образовательного портала РИ АлтГУ.

Сопоставление базового и проектного вариантов производится на основании расчета экономических показателей. Основными из них являются:

- показатель трудоемкости обработки информации;
- показатель эксплуатационных стоимостных затрат;
- экономический эффект;
- текущие затраты пользователя;
- экономия текущих затрат при автоматизации;
- относительная годовая экономия затрат на материалы.

3.3.1 Расчет трудоемкости обработки информации

Пусть T_0 – трудозатраты по базовому варианту (чел/час), T_j – трудозатраты по предлагаемому варианту (чел/час).

Базовый вариант

Действующее мобильное приложение информационно-образовательного портала приспособлено для работы с мобильными устройствами, однако имеет ряд существенных недостатков. Поддержка не всех версий операционной системы Android, а также отсутствие

функциональных возможностей для доступа к личному кабинету довольно сильно затрудняют процесс представления информации. В общей сложности для загрузки информации в приложении через мобильный интернет приходится в среднем около одной секунды. В связи с обновлением интерфейса информационно-образовательного портала института, многократно выросло число обращений к его сервисам. В среднем на каждого пользователя регистрируется около 812 обращений в год.

В мобильном же приложении «Расписание – РИ АлтГУ» регистрируется около 465 обращений, в течение года.

$$T_0 = 465*(2/3600) = 0{,}2583 \text{ чел/час.}$$

Проектный вариант

Проектируемое мобильное приложение, в отличие от приложения «Расписание – РИ АлтГУ», получает информацию, которая содержат данные о расписании, консультациях, предметных комиссиях, новости и данные из личного кабинета, что несущественно больше чем в базовом варианте, потому что «Расписание – РИ АлтГУ» получает данные только о расписании, что дает возможность сэкономить мобильный трафик.

Данное решение позволяет снизить нагрузку на сервер, потребление мобильного трафика (имеется опциональная возможность отключения загрузки изображений новостной ленты) и время загрузки информационных данных. Время загрузки расписания составляет ~ 1,5 секунды. При наличии расписания в памяти устройства отображение данных происходит за 0,4 секунды.

$$T_{j\ интернет} = 465*(1/3600) = 0{,}1255 \text{ чел/час.}$$
$$T_{j\ память} = 465*(0{,}4/3600) = 0{,}0516 \text{ чел/час.}$$

Показатель снижения трудовых затрат (ΔT) рассчитывается по формуле:

$$\Delta T = T_0 - T_j \quad (3.1)$$

$$\Delta T_{0\ интернет} = 0,2583 - 0,1255 = 0,1328 \text{ чел/час.}$$

$$\Delta T_{j\ память} = 0,2583 - 0,0516 = 0,2067 \text{ чел/час.}$$

Коэффициент снижения трудовых затрат (K_m) вычисляется по формуле:

$$K_m = \Delta T_j / T_0 \quad (3.2)$$

$$K_{m\ интернет} = 0,1328 / 0,2583 = 0,51.$$

$$K_{m\ память} = 0,2067 / 0,2583 = 0,80.$$

Таким образом, на 51% и 80% процентов снижаются временные затраты предлагаемого варианта в зависимости от способа получения расписания, по сравнению с базовым вариантом.

3.3.2 Расчет трудоемкости разработки программного обеспечения

Расчет затрат времени на разработку программного обеспечения охватывает работы, выполняемые специалистами на следующих стадиях, каждая из которых имеет следующую трудоемкость:

- техническое задание (3 дня);
- эскизный проект (7 дней);
- технический проект (15 дней);
- рабочий проект (35 дней);
- внедрение (2 дня).

Нормы времени рассчитаны на комплексы задач (задачи) и указаны в человеко-днях. При расчете фактических затрат времени на программирование необходимо учесть влияние таких факторов, как:

- степень новизны комплекса задач;
- сложность алгоритма;
- виды используемой информации;

– сложность контроля входной и выходной информации.

Предусматриваются четыре степени новизны разрабатываемых задач:

А – разработка комплекса задач, предусматривающая применение принципиально новых методов разработки, проведение научно-исследовательских работ;

Б – разработка типовых проектных решений, оригинальных задач и систем, не имеющих аналогов;

В – разработка проекта с использованием типовых проектных решений, при условии их изменения; разработка проектов, имеющих аналогичные решения;

Г – привязка типовых проектных решений.

Так как, данное приложение уже имеет аналоги на рынке, то степень новизны можно определить, как – разработка проекта с использованием типовых проектных решений, при условии их изменения, разработка проектов, имеющих аналогичные решения (В).

Сложность алгоритма

Сложность алгоритма представлена тремя группами:

1. Алгоритмы оптимизации и моделирования систем и объектов.
2. Алгоритмы учета и отчетности, статистики, поиска.
3. Алгоритмы, реализующие стандартные методы решения, а также не предусматривающие применения сложных численных и логических методов.

Сложность данного приложения при разработке была определена как «Алгоритмы оптимизации и моделирования систем и объектов».

Трудоемкость разработки проекта зависит также от вида:

1) используемой информации:

– переменной информации;

– нормативно-справочной информации;

– банк данных;

2) разработки и режима работы:

– режим работы в реальном времени;

– телекоммуникационная обработка данных и управление удаленными объектами, от объема входной информации.

Данное приложение относится к приложениям в режиме работы реального времени.

Сложность организации контроля

Сложность организации контроля входной и выходной информации представлена следующими группами:

11 – входные данные и документы разнообразного формата и структуры. Контроль осуществляется перекрестно, т.е. учитывается связь между показателями различных документов;

12 – входные данные и документы однообразной формы и содержания, осуществляется формальный контроль;

21 – печать документов сложной многоуровневой структуры разнообразной формы и содержания;

22 – печать документов однообразной формы и содержания, вывод массивов данных на машинные носители.

Данное приложение предполагает следующий уровень сложности:

– входная информация – 12;

– выходная информация – 22.

Все необходимые коэффициенты выбираются из таблиц, представленных в приложении А.

При использовании информации различных видов, поправочный коэффициент на стадиях «Технический проект» и «Рабочий проект» рассчитывается по формуле:

$$К_П = \frac{m \cdot К1 + n \cdot К2 + p \cdot К3}{m + n + p}, \quad (3.3)$$

где $К_П$ – поправочный коэффициент; К1, К2, К3 – поправочные коэффициенты согласно таблицам А.3 и А.4 приложения А; m, n, p –

количество наборов данных переменной информации, нормативно-справочной информации и информации при использовании банка данных соответственно.

$$m = 4, n = 1, p = 4;$$

$$K_1 = 1, K_2 = 0{,}72, K_3 = 2{,}08;$$

Технический проект: $К_п = (1*4+0{,}72*1+2{,}08*4)/(4+1+4) = 1{,}45$.

Рабочий проект: $К_п = (1{,}20*4+0{,}65*1+0{,}54*4)/(4+1+4) = 0{,}85$.

Расчет общей трудоемкости

Общий поправочный коэффициент $К_{об}$ определяется как произведение всех применяемых коэффициентов по следующей формуле:

$$К_{об} = K_1 * K_2 *\ldots* K_n, \quad (3.4)$$

где K_1, K_2, …, K_n – поправочные коэффициенты, учитывающие влияние факторов на изменение затрат времени при выполнении конкретной стадии проектирования, $К_{об}$ – общий поправочный коэффициент (i-го вида работы).

Стадии разработки:

– техническое задание – 3 дня;
– эскизный проект – 7 дней;
– технический проект – 15 ($К_{об} = 1{,}45*1{,}00*1{,}26 = 1{,}83$);
– рабочий проект – 45 ($К_{об} = 0{,}85*1{,}00*1{,}36 = 1{,}15$);
– внедрение – 2 ($К_{об} = 1{,}21*1{,}00 = 1{,}21$).

Трудоемкость по этапам с учетом коэффициентов:

– техническое задание – 3 дня;
– эскизный проект – 7 дней;
– технический проект – 23 дня;
– рабочий проект – 50 дней;
– внедрение – 2 дня.

Расчет общей трудоемкости разработки проекта $Т_{об}$ производится по

формуле:

$$6_{\tilde{N}} \; L \; \tilde{A}_5^{\acute{a}} \; P, \qquad (3.5)$$

где, t – трудоемкость работ по стадиям проектирования (от 1 до n), дней.

$$T_{об} = 3 + 7 + 23 + 50 + 2 = 85 \text{ дней.}$$

Численность исполнителей, необходимая для выполнения работ по стадиям проектирования и по комплексам задач (задаче) в целом, вычисляется по формуле:

$$Ч = \text{—}, \qquad (3.6)$$

где, Ч – численность специалистов чел, $T_{об}$ – общая трудоемкость разработки проекта, $Ф_{пл}$ – плановый фонд рабочего времени одного специалиста.

$$Ч = 85 / 150 = 0{,}56.$$

Поскольку коэффициент численности исполнителей не превышает единицы, значит для разработки приложения достаточно одного специалиста.

3.3.3 Смета затрат на разработку программного обеспечения

В смету затрат на разработку программного обеспечения включаются:

– основная и дополнительная зарплата разработчика;

– отчисления на социальные нужды;

– стоимость машинного времени на подготовку и отладку программ;

– материальные затраты.

Основная заработная плата разработчика ($З_{осн}$) рассчитывается по следующим формулам:

$$З_{осн} = Т_{об} * З_{ср.дн}, \qquad (3.7)$$

$$З_{ср.дн} = \frac{\text{–Ô äÐ}}{\text{™ÕĘÇ}}, \qquad (3.8)$$

где, $З_{ср.дн}$ – среднедневная зарплата персонала в рублях, $Т_{об}$ – общая трудоемкость проекта в днях, $З_{ср.мес}$. – среднемесячная заработная плата, $К_{р.д}$. –среднее количество рабочих дней в месяце.

Среднемесячная заработная плата рассчитана и составляет 9200 рублей в месяц.

$$К_{р.д.} = 22 \text{ дня.}$$

$$З_{ср.дн} = 9200 / 22 = 418,18 \text{ р.}$$

$$З_{осн} = 418,18 * 85 = 35545,3 \text{ р.}$$

Дополнительная заработная плата рассчитывается по формуле:

$$З_{доп} = З_{осн} * 10\%, \qquad (3.9)$$

$$З_{доп} = 35545,3 * 0,1 = 3554,53 \text{ р.}$$

Отчисления на социальные нужды составляют 26% от основной и дополнительной заработной платы или 10165,95 рублей.

Стоимость машинного времени зависит от себестоимости машино-часа работы машины, времени работы и амортизацию машины и оборудования ($А_м$) а так же затраты на электроэнергию ($З_{эл}$):

$$А_м = \frac{\text{• × ÛœÃ}}{\text{7:9 Û544}} \text{Û¡ ï}, \qquad (3.10)$$

$О_ф$ – стоимость ЭВМ и оборудования в рублях, $Н_{ам}$ – норма амортизации, принята равной 25%, $А_м$ – амортизационные отчисления, денежные единицы, $Т_м$ – время использования оборудования в днях, равное:

$$Т_м = 0,3 * Т_{тех.пр} + 0,8 * Т_{раб.пр} + 0,6 * Т_{вн}, \quad (3.11)$$

где, $Т_{тех.пр}$, $Т_{раб.пр}$, $Т_{вн}$ – затраты времени на разработку технического проекта, рабочего проекта и внедрения соответственно.

$$Т_м = 0,3 * 15 + 0,8 * 45 + 0,6 * 2 = 42 \text{ дня}.$$

Средняя стоимость компьютера, системные параметры которого будут достаточны для реализации проекта, составляет 30000 рублей, норма амортизации, принята равной 25%.

$$А_м = (30000 * 25 / 365 * 100) * 39 = 801,45 \text{ р}.$$

Затраты на электроэнергию находятся по формуле:

$$З_{эл} = С_{эл} * М_{эвм} * Т_м * Т_{сущ}, \quad (3.12)$$

где $С_{эл}$ – стоимость 1 кВт/ч электроэнергии в рублях (3,5 рубля), $М_{ЭВМ}$ – мощность машины (1,9 кВт/ч), $Т_{сут}$ – суточное время работы машины в часах (8 ч.).

$$Зэл = 3,5 * 1,9 * 42 * 8 = 2234,4 \text{ р}.$$

Материальные затраты можно определить в размере 100 рублей. Это затраты на различного рода расходные материалы.

Полный перечень затрат на разработку программного обеспечения представлен в таблице 3.1.

Таблица 3.1 – Таблица затрат

Элемент затрат	Стоимость (в рублях)
Основная заработная плата	35545
Дополнительная заработная плата	3555
Отчисления на социальные нужды	10166

Продолжение таблицы 3.1

Амортизация ЭВМ и оборудования	801
Затраты на электроэнергию	2234
Материальные затраты	100
Итого:	52402

Из таблицы видно, что невысокие затраты на создание мобильного приложения, экономия трафика и времени пользователей (около 85%), повышение удобства пользования, принесут еще и значительную социальную эффективность.

Можно выделить следующие положительные стороны внедрения мобильного приложения:

– увеличится количество посетителей за счет пользователей мобильного интернета, следовательно, и потенциальных клиентов Института;

– возрастет престижность ВУЗа;

– повысится удобство;

– сократится время доступа к информации;

– сократятся случаи, связанные с опозданиями;

– сократятся затраты пользователей в связи со снижением трафика.

В итоге Институт не только не теряет потенциальных посетителей, но и наоборот приобретёт совершенно новую аудиторию пользователей.

ЗАКЛЮЧЕНИЕ

Целью выпускной квалификационной работы являлась разработка мобильного приложения для организации доступа к ресурсам информационно-образовательного портала Рубцовского института (филиала) Алтайского государственного университета (для платформы Android).

Для достижения поставленной цели были решены следующие задачи:

– дана характеристика исследуемой предметной области;

– проведен анализ предметной области;

– построена функциональная модель обеспечения разработанного программного приложения;

– выявлены недостатки существующей системы предоставления доступа пользователей к информации;

– разработано функциональное «Мобильное приложение РИ АлтГУ»;

– оценена эффективность от внедрения мобильного приложения.

Результатом работы является «Мобильное приложение РИ АлтГУ», которое позволяет:

– получать расписания занятий, консультаций, предметных комиссий;

– корректно отображать информацию на любых экранах мобильных устройств;

– обновлять списки групп, преподавателей и аудиторий;

– сохранять пользовательские настройки;

– автоматически (в фоновом режиме) загружать выбранное расписание;

– оповещать об изменениях состояния расписания;

– заходить в личный кабинет для просмотра текущей успеваемости;

– заказать необходимые справки;

– просматривать актуальные новости информационно-образовательного портала;

– экономить (опционально) мобильный трафик.

Таким образом, поставленная цель была полностью достигнута в рамках выполненного проекта. Разработанное мобильное приложение удовлетворяет информационным потребностям студентов и сотрудников, соответствует современным стандартам, имеет интуитивно-понятный интерфейс, удобный для пользователя, и отлично справляется с выполнением своих функций.

«Мобильное приложение РИ АлтГУ» доступно для загрузки всем пользователям платформы Android в магазине приложений Google Play.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аджич, Г. Как повысить эффективность программных продуктов и проектов по их разработке / Г. Аджич. – М.: Альпина Паблишер, 2017. – 235 с.

2. Артемов, А. Информационная безопасность / А. Артемов. – М.: Академия безопасности и выживания, 2014. – 340 с.

3. Баженова, И.Ю. Основы проектирования приложений баз данных / И.Ю. Баженова. – М.:Национальный Открытый Университет «ИНТУИТ», 2016. – 238 с.

4. Билгин, К. Освоение кросс-платформенной разработки с помощью Xamarin / К. Билгин. – М.: Пакт, 2016. – 390 с.

5. Болодурина, И.П. Системный анализ / И.П. Болодурина. – Оренбург: ОГУ, 2013. – 193 с.

6. Гаврилова, И.В. Разработка приложений / И.В. Гаврилова. – М: ФЛИНТА, 2017. – 243 с.

7. Гущин, А.Н. Базы данных: учебно-методическое пособие / А.Н. Гущин. – М.: Директ-Медиа, 2015. – 311 с.

8. Ерохин, В.В. Безопасность информационных систем / В.В. Ерохин, Д.А. Погонышева, И.Г. Степченко. – М.: ФЛИНТА:Наука, 2016. – 184 с.

9. Кара-Ушанов, В. SQL – язык реляционных баз данных / В. Кара-Ушанов. – М: ФЛИНТА, 2017. – 210 с.

10. Качала, В.В. Теория систем и системный анализ / В.В. Качала. – М.: Горячая Линия-Телеком, 2013. – 272 с.

11. Куликов, С. Тестирование программного обеспечения / С. Куликов. – Минск: Четыре четверти, 2017. – 315 с.

12. Кузнецов, С. Введение в реляционные базы данных / С. Кузнецов. – М.:Национальный Открытый Университет «ИНТУИТ», 2016. – 248 с.

13. Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем / С.В. Маклаков. – М.:Диалог-МИФИ, 2013. – 306 с.

14. Медведкова, И.Е. Базы данных / И.Е. Медведкова, Ю.В. Бугаев, С.В. Чикунов. – Воронежский государственный университет инженерных технологий, 2014. – 108 с.

15. Найролерс, М. Xamarin для Android-программирования. C# / М. Найролерс. – М.: Пакт, 2015. – 298 с.

16. Николаев, Е.И. Базы данных в высокопроизводительных информационных системах / Е.Н. Николаев. – Ставрополь: Северо-Кавказский федеральный университет, 2016. – 163 с.

17. Рыбаков, М. Бизнес-процессы. Как их описать, отладить и внедрить / М. Рыбаков. – М.:Дрофа, 2016. – 570 с.

18. Тидвелл, Д. Разработка пользовательских интерфейсов / Д. Тидвелл – Питер, 2016. – 416 с.

19. Шарп, Д. Подробное руководство по Microsoft Visual C# / Д. Шарп. – СПб.: Питер, 2017. – 848 с.

20. Material Design: на Луну и обратно [Электронный ресурс]. – Режим доступа: https://habr.com/company/redmadrobot/blog/252773/. – Загл. с экрана.

21. Разработка мобильных приложений: с чего начать [Электронный ресурс]. – Режим доступа: https://habr.com/company/mailru/blog/179113/. – Загл. с экрана.

22. SQLite, MySQL и PostgreSQL: Популярные реляционные СУБД [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/. – Загл. с экрана.

23. Xamarin и кросс-платформенная разработка [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/xamarin/. – Загл. с экрана.

24. Введение в OAuth 2 [Электронный ресурс]. – Режим доступа: https://www.digitalocean.com/community/tutorials/oauth-2-ru. – Загл. с экрана.

25. Использование Entity Framework Core code-first с СУБД SQLite [Электронный ресурс]. – Режим доступа: https://habr.com/post/324272/. – Загл. с экрана.

ПРИЛОЖЕНИЕ А

Вспомогательные таблицы для расчета экономической эффективности

Таблица А.1 – Затраты времени при выполнении работ на стадии «Техническое задание»

Комплекс задач, задачи подсистем	Степень новизны			
	А	Б	В	Г
Перспективное планирование развития и размещения отрасли, управление проектированием и капитальным строительством, технико-экономическое планирование	79	57	37	34
Управление материально-техническим снабжением, управление сбытом продукции, управление комплектацией, управление экспортными и импортными поставками	105	76	42	30
Бухгалтерский учет, управление финансовой деятельностью	103	72	30	35
Управление организацией труда и заработной платой, управление кадрами, управление охраной труда	63	46	30	19
Управление качеством продукции, управление технологическими процессами, управление стандартизацией, управление технической подготовкой производства	64	47	31	22
Управление транспортными перевозками, управление техническим обслуживанием производства, управление вспомогательными службами и энергоснабжением	91	66	43	26
Управление НИР и ОКР	50	36	24	15
Управление научно-технической информацией	50	36	24	15
Совершенствование документооборота и контроль исполнения документов	50	36	24	15
Управление охраной природы и окружающей средой	50	36	24	15
Учет пенсий, пособий и страховых операций	79	55	36	26
Статистические задачи	12	111	61	38
Задачи расчетного характера	92	69	47	29

Таблица А.2 – Затраты времени при выполнении работ на стадии «Эскизный проект»

Комплекс задач, задачи подсистем	Степень новизны			
	А	Б	В	Г
Перспективное планирование развития и размещения отрасли, управление проектированием и капитальным строительством, технико-экономическое планирование, оперативное управление, управление ценообразованием	175	117	77	53
Управление материально-техническим снабжением, управление сбытом продукции, управление комплектацией, управление экспортными и импортными поставками	115	79	53	35
Бухгалтерский учет, управление финансовой деятельностью	166	112	67	57
Управление организацией труда и заработной платой, управление кадрами, управление охраной труда	151	101	67	46
Управление качеством продукции, управление технологическими процессами, управление стандартизацией, управление технической подготовкой производства	157	99	67	44
Управление транспортными перевозками, управление техническим обслуживанием производства, управление вспомогательными службами и энергоснабжением	170	100	70	45
Управление НИР и ОКР	151	101	67	46
Управление научно-технической информацией	151	101	67	46
Совершенствование документооборота и контроль исполнения документов	151	101	67	46
Управление охраной природы и окружающей средой	151	101	67	46
Учет пенсий, пособий и страховых операций	103	70	45	36
Статистические задачи расчетного характера	–	–	–	49

Таблица А.3 – Поправочные коэффициенты для определения трудоемкости работ стадии «Технический проект» (К1, К2, К3)

Вид используемой информации	Степень новизны			
	А	Б	В	Г
ПИ	1,70	1,20	1,00	0,50
НСИ	1,45	1,08	0,72	0,43
БД	4,37	3,12	2,08	1,25

Таблица А.4 – Поправочные коэффициенты для определения трудоемкости работ стадии «Рабочий проект» (К1 К2 К3)

Вид используемой информации	Группа сложности алгоритма	Степень новизны			
		А	Б	В	Г
ПИ	1	2,27	1,62	1,20	0,65
	2	2,02	1,44	1,10	0,58
	3	1,68	1,20	1,00	0,48
НСИ	1	1,36	0,97	0,65	0,40
	2	1,21	0,86	0,58	0,34
	3	1,01	0,72	0,48	0,29
БД	1	1,14	0,81	0,54	0,32
	2	1,05	0,72	0,48	0,29
	3	0,85	0,60	0,40	0,24

Таблица А.5 – Поправочные коэффициенты, учитывающие сложность контроля входной и выходной информации на стадиях «Рабочий проект» и «Внедрение»

Сложность контроля входной информации	Сложность контроля выходной информации	
	21	22
11	1,16	1,07
12	1,08	1,00

Таблица А.6 – Поправочные коэффициенты, учитывающие вид информации на стадиях «Рабочий проект», «Внедрение» и «Технический проект»

Стадия разработки проекта	Вид обработки	Степень новизны			
		А	Б	В	Г
Технический проект	РВ	1,67	1,45	1,26	1,10
	ТОУ	1,75	1,52	1,36	1,15
Рабочий проект	РВ	1,75	1,52	1,36	1,15
	ТОУ	1,92	1,67	1,44	1,25
Внедрение	РВ	1,60	1,39	1,21	1,05
	ТОУ	1,67	1,45	1,26	1,10