Профессионального образовательного учреждения
«КОЛЛЕДЖ СОВРЕМЕННОГО УПРАВЛЕНИЯ»

109316, Россия, г. Москва, Волгоградский пр-кт, д. 42, корп. 7. Тел: +7(495) 640-64-36 info@nou-ksu.ru

Специальность: 09.02.05 Прикладная информатика (по отраслям)
ПМ.02 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности
МДК.02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Курсовая работа

Тема: «Проектирование и прототипирование мобильного приложения «Расписание занятий в учебном заведении»

Допущена к защите
зам. директора по УМР: _____
Обучающийся: _____
Работа выполнена: _____ «___» _____20___ г
(Подпись)
Оценка _____
Руководитель работы: _____
_____«___» _____20___ г
(Подпись)

Москва, 2024

Оглавление

Введение

Актуальность исследования обоснована тем, что для студента смартфон стал многофункциональным устройством для решения множества задач – поиск информации в Интернете, коммуникация с одногруппниками и преподавателями, запись лекций, организация собственного учебного процесса и другие.

Проблематика исследования вытекает из того, что качественная организация учебного процесса в ВУЗе сегодня невозможна без создания удобного автоматизированного учебного расписания. Автоматизированное учебное расписание обеспечит равномерную нагрузку всех студенческих групп и преподавательского состава. Точно составленное расписание обеспечивает равномерную загрузку студенческих групп и профессорско-преподавательского состава.

При этом, в случае изменения расписания студенты не всегда оперативно получают информацию о произошедших изменениях.

Интерес к проблеме информирования об изменениях в электронном расписании обусловлен необходимостью оптимизации времени студента, а самым удобным решением, на сегодняшний день, является мобильное приложение.

Объектом исследования, в таком случае, стала система оповещения об имеющихся изменениях расписания в Пермском кампусе ООО «НИ», а предметом – процесс автоматизации составления, изменения и просмотра расписания.

Целью курсовой работы является разработка мобильного приложения расписания занятий для вечерне-заочного факультета с режимом автоматического информирования пользователей. В соответствии с данной целью необходимо решить следующие задачи:

Выделить основные автоматизируемые операции в процессе формирования, просмотра и изменения расписания.

Рассмотреть способы создания мобильного приложения.

Обобщить существующие методы и подходы к разработке мобильного приложения.

Изучить методы составления и способы передачи студентам электронного расписания занятий.

Разработать прототип мобильного приложения.

В исследовании применены такие методы, как сравнительный и описательный метод исследования книг, статей, по теме разработки мобильного приложения; сравнение, обобщение и классификация полученной информации. Также выбор средств программирования, структуры проекта, методов проектирования.

Практическое значение работы представлено разработанным приложением расписания занятий для ООО «НИ» с режимом автоматического информирования пользователей.

Глава 1. Анализ предметной области составления расписания

1.1 Постановка задачи

Расписание занятий вечерне-заочного факультета экономики и управления ООО «НИ» составляется вручную и заполняется в электронных таблицах, после чего распечатывается и выкладывается на сайт. Если нужно оперативно узнать расписание, необходимо перейти на сайт и скачать файл с расписанием. При использовании компьютера никаких проблем не возникает, но при использовании телефона такой формат влечет за собой массу проблем таких как отсутствие приложения для открытия файлов формата XLS, достаточно мелкий шрифт в самой таблице, а также низкая восприимчивость расписания в целом, это видно на рисунке 1.

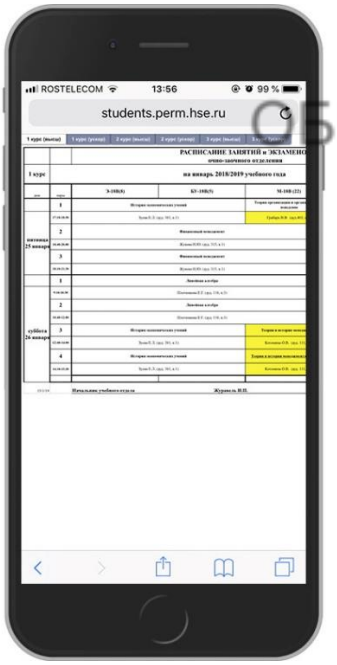

Рис. 1. Вид расписания с экрана смартфона.

Так как расписание составляется вручную не исключен человеческий фактор, то есть ошибки в расписании или опечатки в названии предметов и имен преподавателей.

Также нет системы оповещения об изменении расписания, следствии чего учебной части необходимо оповещать студентов по электронной почте. Система антиспама представителя услуг электронной почты может распознать такое письмо как спам и пользователь не может вовремя узнать о изменении расписание, или же письмо будет доставлено слишком поздно.

Расписание опубликовано на сайте, для просмотра необходим доступ в интернет поэтому не исключен вариант, когда выход в интернет ограничен или же сайт недоступен.

Перед реализацией системы необходимо выделить критерии, которые содержат функциональные особенности. Список выделанных критериев:

Общедоступность, бесперебойная работа везде, где есть связь с интернетом.

Автономность системы, без доступа к сети программа может работать, храня расписание в базе на устройстве, и оно будет актуально в течении 24 часов.

Простой и понятный интерфейс, просмотр расписания будет удобен для просмотра с возможностью масштабирования под любой размер экрана. С навигацией в приложении можно будет разобраться за пару минут.

Надёжность и стабильность работы, приложение будет быстро работать, не заставляя ждать пользователя и не будет закрываться при критических ошибках.

Безопасность, приложение обеспечивает надежную защиту пользовательских данных путем хеширования паролей пользователя.

Задача составления расписаний являются предметом научных исследований с середины прошлого века. Область применения расписания затрагивает многие сферы человеческой жизни, например промышленность, образование, логистику, транспортные перевозки и т.д.

Деятельность человека предлагает задачи, найти эффективное решение которых невозможно без исследования всех вариантов составления расписания. Для большинства моделей теории расписаний нахождение

лучшего варианта является нелёгкой задачей, так как варианты расписания должны удовлетворять множеству ограничений организационного и производственного характера, которые в свою очередь могут создавать конфликтные ситуации между разными видами ограничений.

Качественный и количественный рост учащихся ООО «НИ» требует применения автоматизированного подхода к управлению научной, учебной и хозяйственной деятельностью вузов. Новый подход к учебному расписанию в последнее время реализуется с помощью применения математических методов и современных средств вычислительной техники в управлении высшими учебными заведениями.

В настоящее время существует тенденция к автоматизации технических процессов, которые ранее выполнялись вручную. Например, прогнозируемые системы в науке и технике, экспертные системы, способные выполнять ряд действий за опытных специалистов, система принятия решения в маркетинге и т.д.

Таким же процессом автоматизации технических процессов относится и составление расписание. Во многих учебных заведениях до сих пор расписание создается вручную. Хотя применение современных технологий в данной сфере позволят наилучшим образом организовать учебный процесс.

Подходы к автоматизации составления расписания

Во многих университетах расписание формируется в ручном режиме, используя картонные планшетах и рукописный текст. Распространения расписания при таком режиме формирование крайне затруднительно.

Автоматизация ввода и хранения информации о расписании занятий приведет к систематизации информации, упростит процедуру формирования и распространения расписания.

Программное обеспечение позволяет успешно в автоматическом режиме формировать расписание занятий. Однако при этом, теряется накопленный опыт и устойчивая структура составления расписания, которые выработаны при использовании ручного режима.

Вместе с тем, программы чаще используют локальный подход, то есть автоматизация происходит лишь в отделе, который отвечает за составление расписания. Сотрудникам данного отдела требуется ввести большой объем исходной информации в единую базу данных для автоматизации процесса составления расписания.

Как показывает нижеприведенная схема на рисунке 2, объем входных данных намного превышает объем выходных данных, что повлечет за собой огромные затраты времени только для составления расписания.

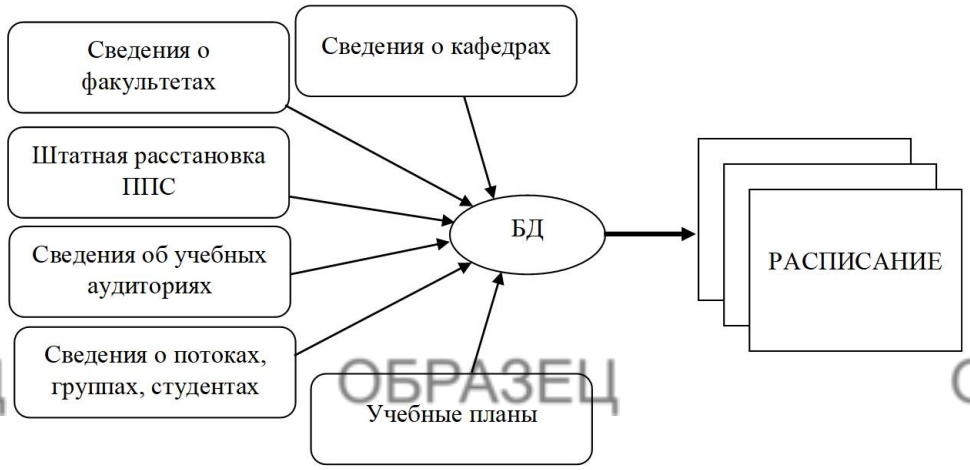

Рис. 2. Схема входных и выходных данных.

Возможность использования базы данных для решения других задач отсутствует.

1.2 Анализ подходов к автоматизации составления расписаний учебных занятий в образовательных учреждениях

Задача составления расписания – это процесс распределения конечного набора событий во времени с учетом определённых ограничений.

Исходными данными для составления расписания являются перечни дисциплин, специальностей, списки групп студентов, преподавателей,

учебная нагрузка для каждой группы. Исходные данные выражены в качестве списков:

Учебных групп.

Профессорско-преподавательского состава.

Дисциплин, которые изучает та или иная учебная группа.

Аудиторий вуза.

Количества учебных дней для учебных групп.

Максимального количества занятий в день и других.

На основе исходных данных, необходимо, во-первых, удобно образом распределить все исходные данные в соответствии с запросами, во-вторых, для каждой группы учащихся определить оптимальное время для занятий.

Составления расписания является задачей, сложность решения которой может расти сопоставимо с ростом числа возможных вариаций. А также, для подобных задач характерно наличие большого объема исходной информации, которая в свою очередь имеет различный состав и множество дополнительный требований. Вышеперечисленные сложности создают проблемы при автоматизации процедуры составления расписания при существовании множества вариантов.

Благодаря данным методам возможно получить точную математическую модель, отвечающую необходимым ограничениям. Однако из-за сложности поставленной задачи эта модель будет достаточно сложной и объемной.

Для решения проблемы применяется метод моделирование, при котором алгоритм пользуется непосредственно расписанием и списком занятий, которые необходимо вставить в расписание. При этом весь процесс начинается с пустой формы, в которой все занятия пока находятся в списке неучтенных в расписании.

После чего алгоритм переходит от одного незаполненного расписания учебной группы к другому, стремясь наилучшим образом расположить все имеющиеся неучтенные занятия.

Процесс продолжается пока не будет выполнено одно из условий: либо не будет сформировано полное расписание, которое будет отвечать всем существующим ограничениям, либо алгоритм не выполнит фиксированное количество итераций.

При реализации данного алгоритма отдельное внимание уделяется разработке эвристических правил выбора занятия из списка, определение наиболее удачной позиции в расписании и оценке получившегося расписания.

У данного подхода имеются как положительные, так и отрицательные стороны. К положительным сторонам можно отнести возможность учета всей специфики задачи в случае составления расписания для конкретного вуза. К отрицательным сторонам относятся ограничения применения разработанной системы в других учебных заведениях, а также при внутренних изменениях в ВУЗе понадобится вносить большие изменения в алгоритм.

Делая вывод из всего вышеперечисленного, можно отметить, что существуют разные варианты автоматизации учебного расписания в высшем учебном заведении. В каждом случае необходимо выбрать оптимальный вариант, который будет отвечать потребностям конкретной задачи.

Однако в данной работе, целью которой является разработка готового мобильного приложения, пользователи приложения будут пользоваться лишь результатами автоматизации процесса составления расписания. Поэтому автоматизация, в данном случае, фигурирует как способ достижения готового результата.

1.3 Анализ аналогичных систем

После определения критериев к системе необходимо проанализировать существующие аналоги, чтобы выявить положительные и отрицательные стороны приложений для учета в дальнейшей разработке собственного приложения. Для этого проведен обзор приложений для расписания учебных занятий в сети Интернет.

В ходе обзора выделены три приложения подобного направления: «Расписание занятий», «Расписашка» «Studify – расписание ВУЗов». Далее приводится краткий обзор каждого приложения.

Приложение «Расписание занятий»1 разработчиком которого является Ефремов Иван, создано для обобщения расписания занятий ОГАПОУ "Белгородского индустриального колледжа" на рисунке 3.

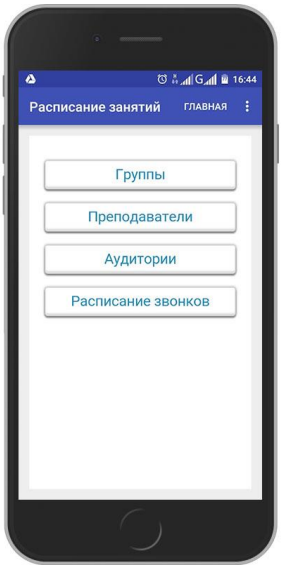

Рис. 3. Главная страница приложения.

Основными возможностями приложения являются просмотр расписания отдельной группы, преподавателя, учебной аудитории и расписание звонков.

Следующим приложением для анализа стало «Расписашка»2 от разработчика team13 на рисунке 4.

[1] Расписание занятий URL: https://play.google.com/store/apps/details?id=ru.bincol.rasp

[2] Расписашка URL: https://play.google.com/store/apps/details?id=tk.frostbit.timetable

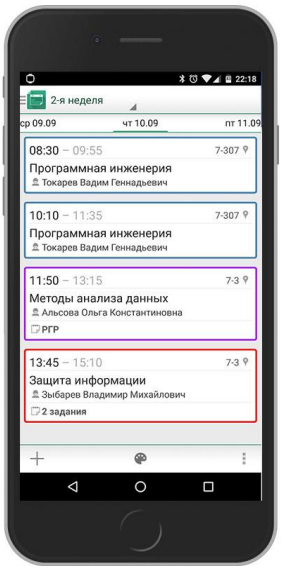

Рис. 4. Приложение «Расписашка»

Как указывает разработчик в описании, «Расписашка» – это мгновенный доступ расписанию. По сравнению с первым, данное приложение имеет больше возможностей для пользователей

автоматическое включение беззвучного режима на занятиях;

показ расписания на текущую неделю или все недели;

создание заданий с заданным сроком выполнения и без него;

раскраска занятий по предметам, преподавателям, видам;

использование нескольких расписаний с быстрым переключением между ними;

выбор любых недель для занятий;

экспорт расписания в календарь на устройстве;

праздники в расписании;

виджет, показывающий все предстоящие занятия на день и время до окончания, текущего;

импорт/экспорт расписания на SD карту.

Последним приложением в анализе аналогичных систем стало приложение «Studify – расписание ВУЗов»3 от разработчика Studify and Talks на рисунке 5.

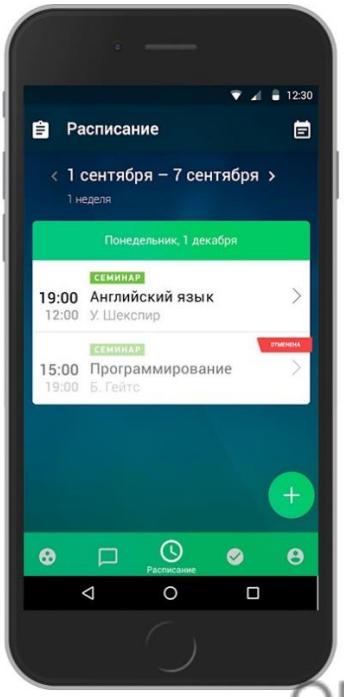

Рис. 5. Приложение Studify – расписание ВУЗов

Данное приложение предназначено для студентов всей страны. В нем содержится расписание более 440 университетов и институтов, более 35 000 групп с расписанием.

Основными функциями данного приложения являются актуальное расписание группы доступно онлайн и оффлайн, управление расписанием, расписание для преподавателей с возможностями ведения журнала посещаемости группы и добавления домашних заданий по предметам, собственный виджет, возможность настройки отображения расписания:

– нумерация недель;

– календарь;

– понедельный просмотр расписания;

[3] Studify – расписание ВУЗов URL: https://play.google.com/store/apps/details?id=com.raspisaniyevuzov.app

– карточка деталей пары;

– маркировка типов пар.

Данное приложение выходит за рамки «локального» приложения, что является его главным отличием от других приложений, рассмотренных ранее. Дополнительно в нем есть удобные функции для преподавателя и возможность просмотра расписания оффлайн.

Анализ аналогичных систем произведен по функциональным параметрам:

расписание группы;

расписание преподавателя;

управление расписанием;

чат группы;

отметка об отсутствии;

маркировка типов пар;

виджет.

Результаты анализа представлены в таблице 1.

Таблица 1

Анализ аналогичных систем

Функционал	Расписание занятий	Расписашка	Studify расписание ВУЗов
Расписания группы	+	+	+
Расписание преподавателей	+	+	+
Управление расписанием	-	+	+
Чат группы	-	+	-
Отметка об отсутствии	-	-	-
Маркировка типов пар	-	+	+
Виджет	-	+	-

Анализ аналогичных приложений показал, что каждое приложение имеет свои уникальные функции. Приложение «Расписание занятий»

достаточно простое, не нагруженное функциями приложение. Приложение «Расписашка» имеет огромные функциональные возможности начиная от самого расписания занятий и заканчивая виджетом. Приложение «Studify» многофункциональное приложение, сотрудничает со множеством ВУЗов.

1.4 Формирование функциональных и нефункциональных требований

Расписанием пользуются многие сотрудники университета, а также сами учащиеся, но чаще всего им пользуются преподаватели и студенты, чтобы узнать в каком кабинете пара, какой урок будет проходить, время начала урока, а также какая группа или какой преподаватель будет.

Для успешной коммуникации между преподавателем и студентами в системе должна быть предусмотрена авторизация пользователей для создания чатов групп с преподавателями.

После авторизации в системе студент имеет возможность поставить отметку об отсутствии на занятии, добавить заметку к предмету, добавить напоминание.

Преподаватель в свою очередь должен права в системе на изменение кабинета занятия и возможность добавления домашнего задания.

Диаграммы прецедентов (Use case diagram) – диаграммы, позволяющие создавать список операций, выполняемых системой. Иногда такие диаграммы называют диаграммой функций, так как в основе ее лежит список требований к системе, который определяется множеством выполняемых функций.

Каждая такая диаграмма (Use case) – это определение сценария поведения, которому следуют действующие лица (Actor). Такая диаграмма впервые была предложена Иваром Якобсоном в 1986 году. Диаграммы использования не меняет свой вид уже более двадцати лет.

Прецедент – функциональность системы, которая позволяет пользователю получить ощутимый и измеримый для него результат.

Прецедент соответствует определяет способ использования этой системы пользователем.

Прецеденты применяются для спецификации требований к проектируемой системе или поведения уже существующей системы. Дополнительно прецеденты описывают способы взаимодействия системы и пользователя, которые позволяют правильно работать с предоставляемыми системой сервисами.

Проанализировав функционал системы из предыдущей главы можно выделить следующие прецеденты:

посмотреть расписание;

добавить заметку к предмету;

добавить напоминание;

установить отметку об отсутствии;

начать чат;

авторизация;

добавить изменения к предмету;

добавить домашнее задание.

Также можно выделить два субъекта: Студент и Преподаватель.

Прецеденты и субъекты, данной модели, представлены в виде следующей диаграммы:

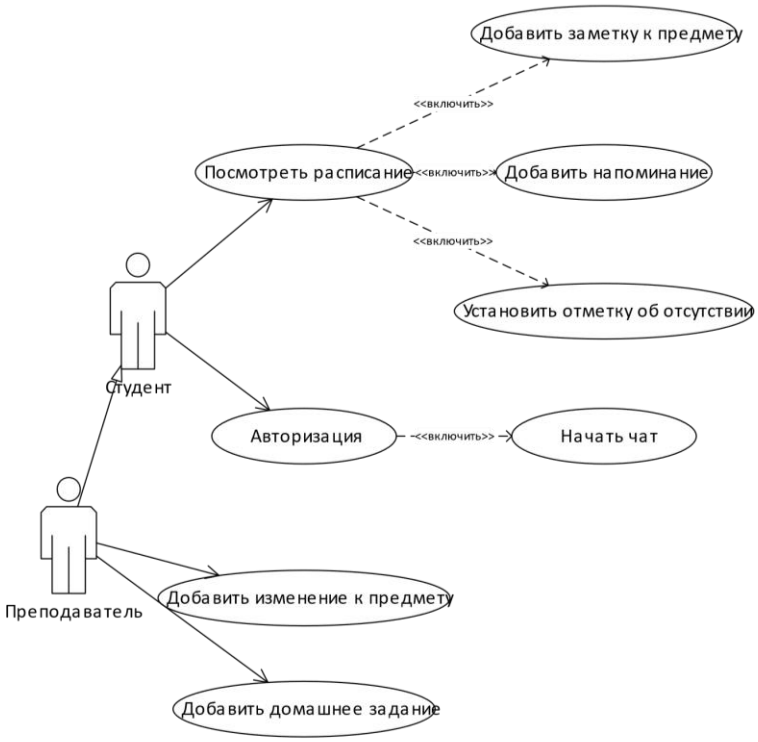

Рис. 6. Диаграмма прецедентов

Диаграмма прецедентов – это не просто некая схема, она представляет собой полностью документированную модель системы. Такой вид диаграмм необходим для моделирования и организации поведения системы. На схеме представлены актеры прецеденты и акторы, отношения между ними.

. Развернутое описание прецедентов

Описание документа должно содержать:

Название прецедента.

Краткое описание.

Участвующие субъекты (Акторы).

Предусловия или триггер.

Детализированное описание потока событий:

основной поток, который разбивается на подчиненные потоки событий;

альтернативные потоки, определяющие исключительные ситуации.

Постусловия, которые отражают состояние, по достижении которого прецедент завершается.

Таблица 2

Развернутое описание прецедента «Авторизация»

Название прецедента:	Авторизация.
Акторы:	Студент, Преподаватель.
Краткое описание:	Студент входит в систему, система загружает экран приветствия.
Триггер:	Нет.
Основной поток:	
Действия акторов:	Отклик системы:
Студент (E1) вводит логин и пароль для пользователя, и нажимает кнопку входа.	Система подтверждает правильность ввода (E2), и загружает главное меню программы.
Альтернативный поток:	
E1: Преподаватель вводит логин и пароль, и нажимает кнопку входа. Прецедент продолжается. E2: Система выдает ошибку ввода логина или пароля, и предлагает ввести заново, либо восстановить пароль.	

Таблица 3

Развернутое описание прецедента «Посмотреть расписание»

Название прецедента:	Посмотреть расписание.
Акторы:	Студент.
Краткое описание:	Студент просматривает расписание занятий.
Триггер:	Нет.
Основной поток:	
Действия акторов:	Отклик системы:
Студент выбирает подменю расписание.	Система открывает расписание занятий на текущую неделю.(E1)
Альтернативный поток:	
E1: Если расписания нет на текущую неделю студенту будет показано сообщение об отсутствии расписания. Прецедент продолжается.	

Таблица 4

Развернутое описание прецедента «Добавить заметку к предмету»

Название прецедента:	Добавить заметку к предмету.
Акторы:	Студент.
Краткое описание:	Студент добавляет заметку к предмету.

Триггер:	Посмотреть расписание.
Основной поток:	
Действия акторов:	Отклик системы:
Студент выбирает предмет.	Система открывает предмет.
Студент нажимает кнопку «Добавить заметку».	Система отрывает поле ввода заметки.
Студент вводит заметку и нажимает кнопку сохранения.	Система сохраняет
Альтернативный поток:	

Таблица 5

Развернутое описание прецедента «Добавить напоминание»

Название прецедента:	Добавить напоминание.
Акторы:	Студент.
Краткое описание:	Студент добавляет напоминание к предмету.
Триггер:	Посмотреть расписание.
Основной поток:	
Действия акторов:	Отклик системы:
Студент выбирает предмет.	Система открывает предмет .
Студент нажимает кнопку «Добавить напоминание».	Система отрывает поле ввода напоминания и ввода даты.
Студент добавляет напоминание.	Система сохраняет.
Альтернативный поток:	

Таблица 6

Развернутое описание прецедента «Установить отметку об отсутствии»

Название прецедента:	Установить отметку об отсутствии.
Акторы:	Студент.
Краткое описание:	Студент устанавливает отметку об отсутствии.
Триггер:	Посмотреть расписание.
Основной поток:	
Действия акторов:	Отклик системы:
Студент выбирает предмет.	Система открывает предмет .
Студент нажимает кнопку «Установить отметку».	Система устанавливает отметку.
Альтернативный поток:	

Таблица 7

Развернутое описание прецедента «Начать чат»

Название прецедента:	Начать чат.
Акторы:	Студент, Преподаватель.
Краткое описание:	Студент устанавливает отметку об отсутствии.
Триггер:	Нет.
Основной поток:	
Действия акторов:	Отклик системы:
Студент (E1) входит в меню чатов.	Система открывает меню чатов.
Студент выбирает чат по предмет.	Система открывает чат.
Альтернативный поток:	
E1: Преподаватель входит в меню чатов.	

Таблица 8

Развернутое описание прецедента «Добавить домашнее задание»

Название прецедента:	Добавить домашнее задание.
Акторы:	Преподаватель.
Краткое описание:	Преподаватель добавляет домашнее задание.
Триггер:	Нет.
Основной поток:	
Действия акторов:	Отклик системы:
Преподаватель входит в меню предметов.	Система открывает меню предметов.
Преподаватель выбирает предмет.	Система открывает предмет.
Преподаватель добавляет домашнее задание.	Система сохраняет домашнее задание.
Альтернативный поток:	

Таблица 9

Описание прецедента «Добавить изменения к домашнему заданию»

Название прецедента:	Добавить изменения к предмету.
Акторы:	Преподаватель.
Краткое описание:	Преподаватель добавляет изменения к предмету.
Триггер:	Нет.
Основной поток:	
Действия акторов:	Отклик системы:
Преподаватель входит в меню предметов.	Система открывает меню предметов.

Преподаватель выбирает предмет.	Система открывает предмет.
Преподаватель добавляет изменения к предмету.	Система сохраняет изменения.
Альтернативный поток:	

Выводы

Для успешной коммуникации между преподавателем и студентами в системе должна быть предусмотрена авторизация пользователей для создания чатов групп с преподавателями.

Анализ аналогичных приложений показал, что каждое приложение имеет свои уникальные функции, но не одно не имеет полный набор необходимых инструментов.

Глава 2. Проектирование информационной системы мобильного расписания

2.1 Выявление классов – сущностей

Одновременно с моделированием вариантов использования выявляются классы сущности, их атрибуты и взаимосвязи между ними, что является диаграммами классов. Диаграммы классов используются для построения логической модели системы, которая содержит лишь постоянные аспекты функционирования системы.

Под классом в языке UML подразумеваются объекты, обладающие одинаковой структурой, поведением и взаимодействием с объектами других классов.

Класс-сущность — это класс, способный определить типы объектов и связи между ними. Это постоянные элементы системы, которые могут извлекаться, меняться, перезаписываться.

Выполнив анализ предметной области, были выявлены следующие классы- сущности: Преподаватель, Предмет, Студент, Группа, Аудитория, Чат, Сообщение, Пользователь, Занятие.

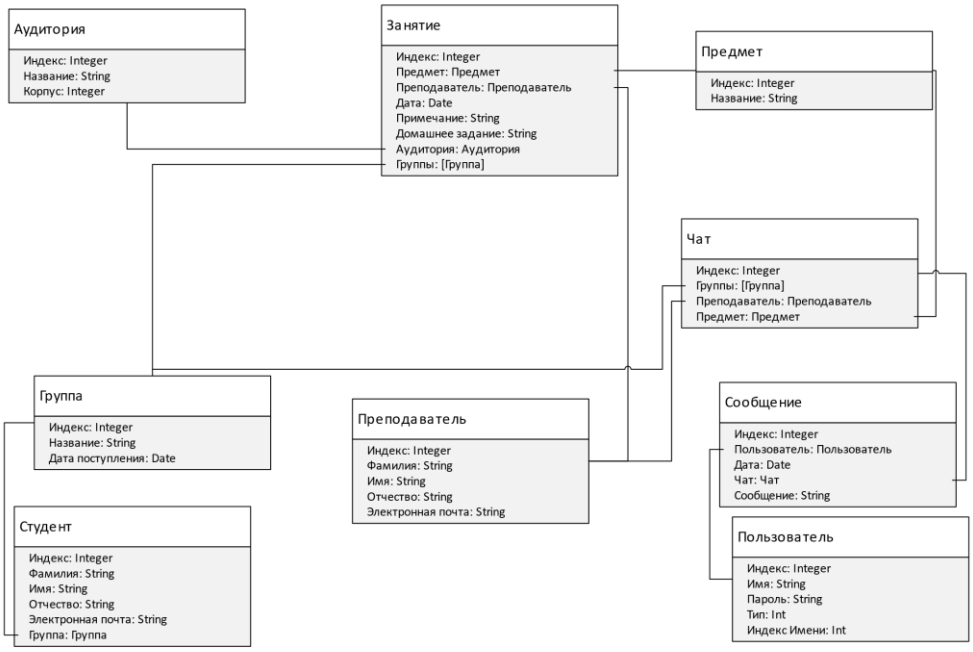

Рис. 7. Классификация классов-сущностей

Класс Предмет содержит информацию об наименовании урока, имеет два параметра Индекс типа Целое число и Название типа Строка.

Класс Преподаватель содержит информацию о преподавателя: Фамилию, Имя, Отчество и Электронную почту типа Строка.

Класс Группа содержит информацию о наименовании группы, почту и список студентов.

Класс Студент содержит информацию о имени студента и его группе.

Класс Аудитория – информация о номере кабинета и корпуса, где будет проходить занятие.

Класс Занятие является моделью, описывающей представление занятия. Содержит ссылки на преподавателя ведущего занятие, название предмета, массив групп, аудиторию. Также имеет дату начала занятия, домашнее задание и примечание преподавателя.

Класс Чат описывает модель чата в которой студенты разных групп и преподаватель смогут общаться по текущей дисциплине.

Класс Сообщения – сообщения пользователей в чате.

Класс Пользователи описывает представление пользователей приложения, имеет имя, пароль, тип (студент или преподаватель), ссылку на данные в зависимости от типа.

Модели деятельности основаны на описании общей последовательности действий для объектов и вариантов использования. Этот вид диаграмм относится к динамическим представлениям системы, является полезным при моделировании функционирования системы, так как отражает передачу потока управления между объектами.

Основным элементом диаграммы является деятельность. На концептуальной диаграмме деятельность — это задача, которую нужно выполнить вручную или автоматизировать. На диаграмме, построенной на спецификации или реализации, деятельность представляет собой реализацию метода над классом.

Диаграмма деятельностей устанавливает основные правила последовательности. Такая возможность необходима при моделировании бизнес-процессов. Данный метод наилучшим образом работает с бизнесс-процессами, которые не обязательно выполнять последовательно. Метод позволяет выполнять процессы параллельно.

Диаграммы деятельности являются полезными и при параллельном программировании. С помощью данной диаграммы возможно графически изобразить все ветви процесса и определить момент синхронизации.

При этом, необходимо синхронизировать параллельные деятельности, при их наличии. Простая линейка синхронизации определяет активность выходной деятельности при выполнении обеих входные деятельности.

Диаграммы деятельности применяются при описании потоков событий. С помощью текста можно подробно рассказать о потоке событий. Однако в сложных потоках с альтернативными ветвями будет сложно понять всю логику событий. Диаграммы деятельности предоставляют

Диаграмма деятельности данного вида отражает:

события, инициирующие действия или являющиеся конечным результатом;

последовательность действий;

условия расширения сценария.

Для того чтобы отобразить соответствие деятельности определенному пользователю или Системе к данной диаграмме можно применить «дорожки». Так как в нашем случае все действия выполняются менеджером, применение разделителей не целесообразно.

Данный способ иллюстрации бизнес-процесса может охватывать не только действия, происходящие внутри, разрабатываемой системы, но производящиеся за ее пределами, что необходимо для формирования четкого представления о процессе в целом.

2.2 Диаграммы деятельности для проектируемой системы

Диаграмма деятельности – технология, позволяет описать логику процедур, бизнес-процессы и потоки работ. Диаграмма деятельности акцентирует внимание на последовательности выполнения определенных действий, которые в совокупности приводят к получению желаемого результата. Они могут быть построены для отдельного варианта использования, кооперации, метода.

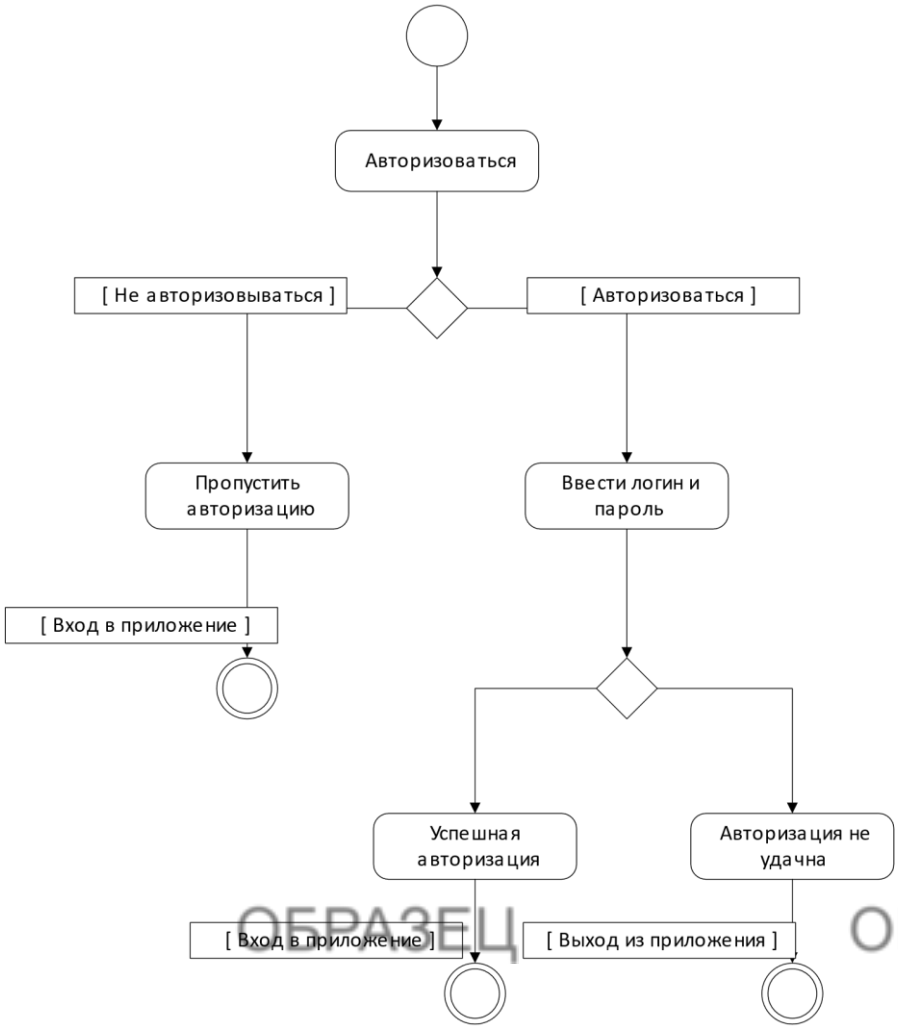

Рис. 8. Авторизация в приложении

На рисунке 8 показан процесс авторизации в системе. При первом запуске приложения система предлагает авторизоваться. Пользователь может пропустить авторизацию и просто просматривать расписание. Если пользователь авторизуется ему станут доступны функции общения по предметам между студентами и преподавателями.

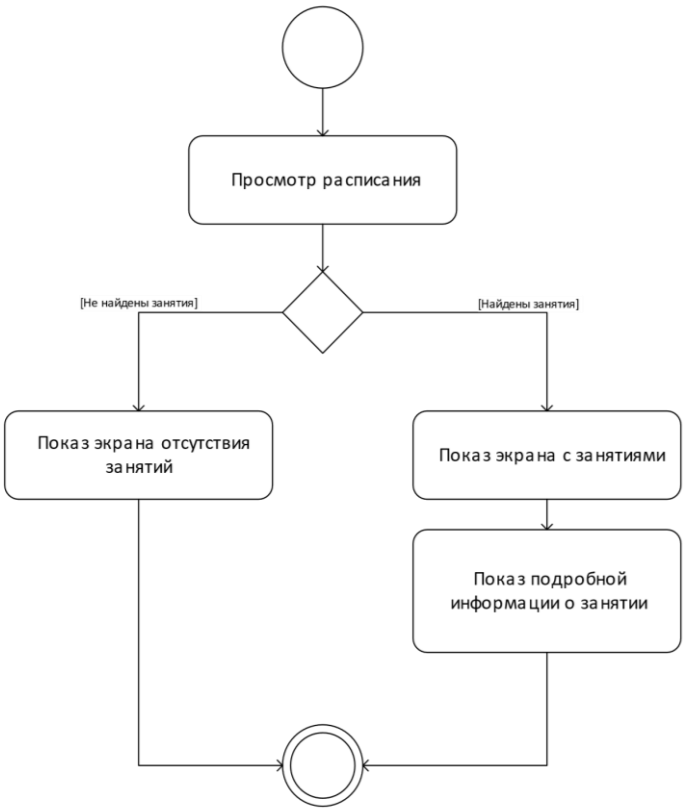

Рис. 9. Просмотр расписания.

На рисунке 9 описан процесс просмотра расписания. После авторизации или пропуска авторизации пользователь может просмотреть расписание выбранной группы и посмотреть подробную информацию о занятии.

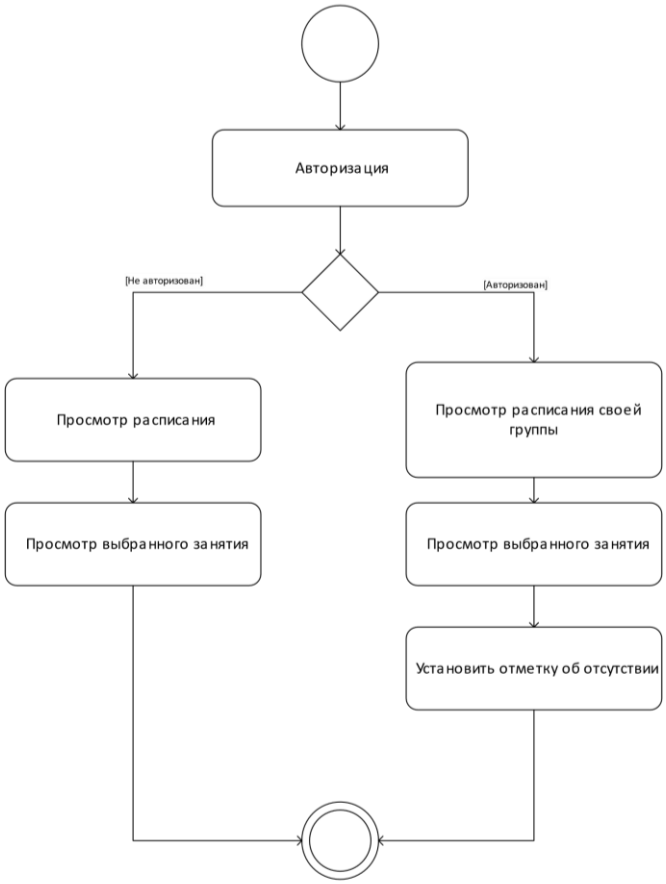

Рис. 10. Установить отметку об отсутствии.

Далее на рисунке 10 изображен процесс установки отметки об отсутствие студента на занятии. Пользователь должен быть авторизован, далее он видит расписание своей группы, открывает нужное занятие и может установить отметку об отсутствии.

2.3 Эскизное моделирование

Интерфейс приложения должен быть максимально просто и удобен для использования. Много лет дизайнеры стараются делать интерфейс красивым и легким в использовании. Для приложений на Андроид компания Google предложила собственную концепцию дизайеа интерфейса Material Design. Впервые представлен на конференции Google I/O 25 июня 2014 года. Идея

дизайна заключается в приложениях, которые открываются и сворачиваются как карточки, используя эффекты теней.

Такая концепция стала результатом изучения компанией работы с бумагой в реальности. Команда создала иконки приложений из бумаги. Они хотели увидеть взаимодействие света и тени с плоским, но реальным материалом.

2.4 Прототип приложения

Первым экраном приложения на рисунке 11 представлена авторизация, которую можно будет пропустить и перейти сразу к выбору группы и ее расписанию.

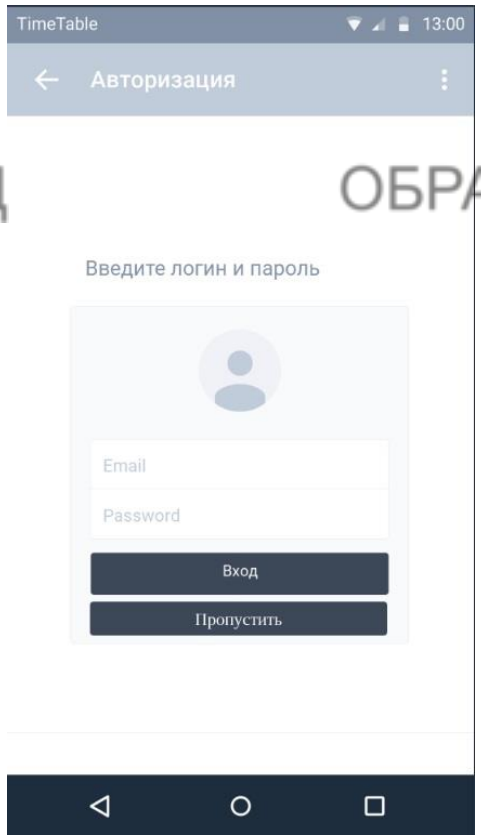

Рис. 11. Авторизация

Далее на рисунке 12 представлен прототип списка занятий группы. По нажатию на день происходит переход на список предметов.

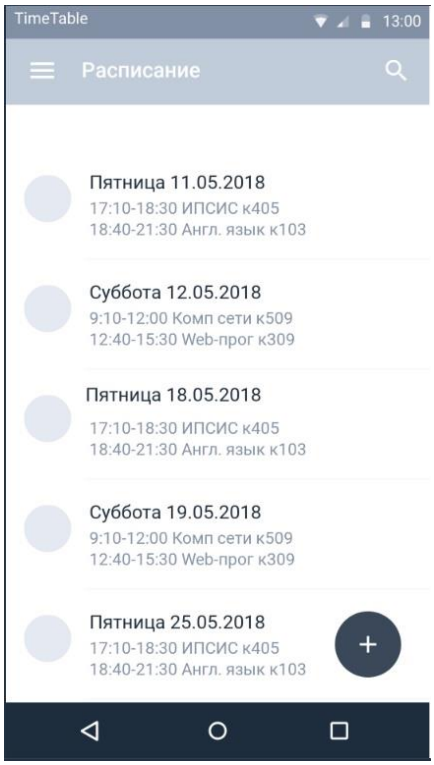

Рис. 12. Список предметов группы

По нажатию на предмет происходит переход к информации о занятии на рисунке 13, где отображается дата, время, преподаватель, корпус, кабинет и домашнее задание. Также на этом экране есть кнопка, устанавливающая отметку об отсутствии студента, если тот по какой-то причине не может посетить занятие.

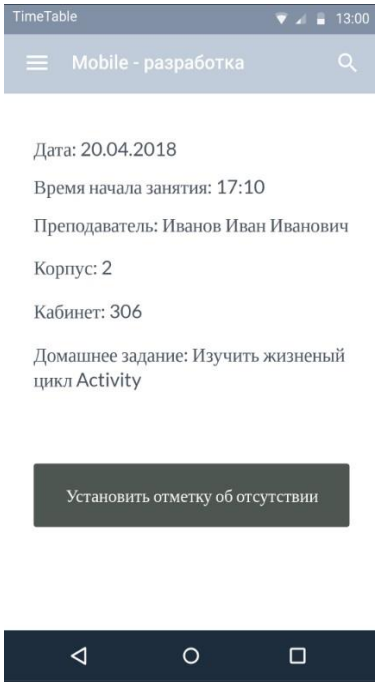

Рис. 13. Информация о занятии

Выводы

Одновременно с моделированием вариантов использования выявляются классы сущности, их атрибуты и взаимосвязи между ними, что является диаграммами классов. Диаграммы классов используются для построения логической модели системы, которая содержит лишь постоянные аспекты функционирования системы.

Выполнив анализ предметной области, были выявлены следующие классы-сущности: Преподаватель, Предмет, Студент, Группа, Аудитория, Чат, Сообщение, Пользователь, Занятие.

При первом запуске приложения система предлагает авторизоваться. Пользователь может пропустить авторизацию и просто просматривать расписание. Если пользователь авторизуется ему станут доступны функции общения по предметам между студентами и преподавателями.

После авторизации или пропуска авторизации пользователь может просмотреть расписание выбранной группы и посмотреть подробную информацию о занятии.

Глава 3. Разработка мобильного приложения «Расписание занятий для ООО «НИ»

3.1 Выбор операционной системы для внедрения приложения

Анализ существующих операционных систем

Сегодня в магазинах существует огромное разнообразие моделей мобильных телефонов. Большинство всех моделей работают на двух самых популярных операционных системах – это iOS и Android. До 80 % всех мобильных устройств в мире работают на этих операционных системах.

Хотя iOS и Android являются прямыми конкурентами, которые борются за охват большей аудитории. Каждая ОС представляет собой некий сформировавшийся «продукт» со своими достоинствами и недостатками. Для сравнения рассмотрим каждую операционную систему.

iOS – это мобильная ОС компании Apple, которая устанавливается на гаджеты собственного производства: iPhone, iPad, iPod Touch и другие. Во время анонса своего первого iPhone в 2007 г. Apple заявила,что в нем используется специальная версия Mac OS X, переработанная под мобильные версии.

Mac OS X – это проприетарная ОС производства Apple, построенная на базе платформы UNIX и в полной мере задействующая все возможности программного обеспечения.

Android – ОС для мобильных телефонов, планшетных компьютеров, наручных часов и других устройств. Первоначально Android задумывалась как система с открытым исходным кодом. Как только появлялась новая версия ОС, сразу же вся необходимая документация выкладывалась в общий доступ. Тем самым это облегчало для программистов создание новых программ и игр.

Операционная система Android основана на ядре Linux. Тип ядра представляет собой монолитное модифицированное ядро Linux. Монолитное ядро – на сегодняшний день наиболее распространенная архитектура ядер

операционных систем. Оно является старейшим принципом организации ОС и применяется в большинстве UNIX систем.

В таблице 10 представлено сравнение мобильных операционных систем по девяти показателям. Обобщая сравнительный анализ операционных систем iOS и Android можно сделать вывод, что выбор ОС необходимо осуществить с учетом тех задач, которые нужно решить в ходе разработки приложения.

Таблица 10

Сравнение мобильных операционных систем

Показатель	Android	iOS
1. Тип системы	Открытая	Закрытая
2. Защищенность системы от вирусов	Незащищенная	Защищенная
3. Связка модели телефона и программного обеспечения	Для разных моделей устройств	Строго для своих устройств
4. Автономность	Высокое энергопотребление	Грамотная работа с аккумулятором
5. Стабильность	Менее стабильна	Более стабильна
6. Настройки интерфейса	Более настраиваемый интерфейс	Мало настроек
7. Степень освоения	Сложнее в освоении	Легче в освоении
8. Скорость работы привводе-выводе	Работает медленнее	Работает быстрее
9. Потребность в оперативной памяти	Больше	Меньше

3.2 Анализ продаж мобильных устройств

Проанализируем продажи смартфонов. По данным портала TAdviser продажи смартфонов с 2022 по 2023 выросли в среднем на 10%4. В таблице 11 представлены лидеры продаж смартфонов за 2022 и 2023 года.

[4] [TAdviser] URL: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A1%D0%BC

Таблица 11

Лидеры российского рынка смартфонов

Производители	2й Квартал 2022 г.	2й Квартал 2023 г..
Samsung	31%	30%
Huawei	11%	29%
Apple	11%	10%
Bright & Quick	4%	7%
Xiaomi	4%	8%
Другие	39%	16%
Итого	100%	100%

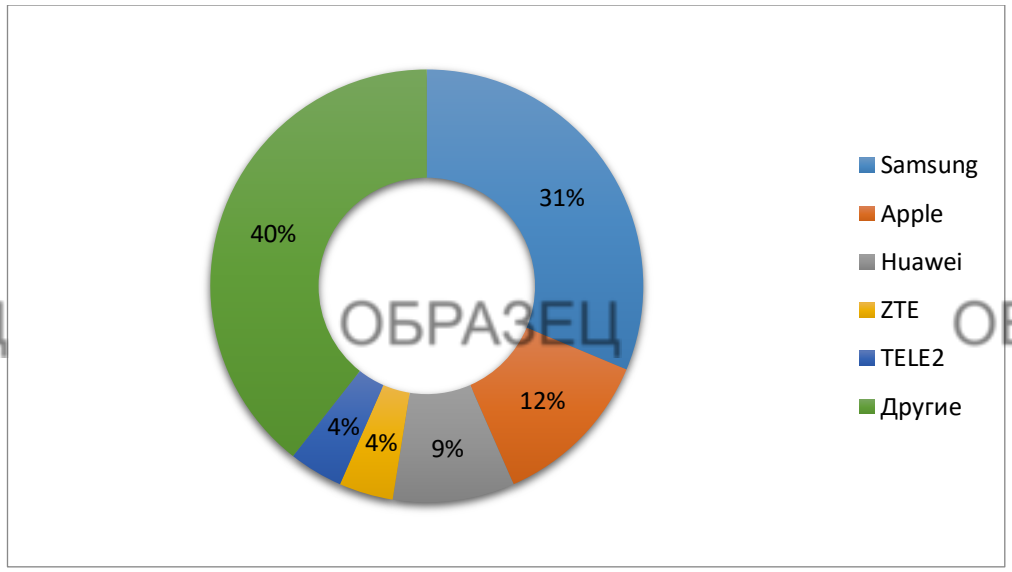

Рис. 14. Производителей смартфонов на российском рынке.

На рисунках 15, 16, 17 представлены распределение сил в сегментах международных, китайских и локальных брендов на российском рынке смартфонов, данные Counterpoint Technology Market Research5 за первый квартал 2023 г.

%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD%D1%8B_%28%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%80%D1%8B%D0%BD%D0%BE%D0%BA%29

[5] [Counterpoint Technology Market Research] URL: https://www.counterpointresearch.com/

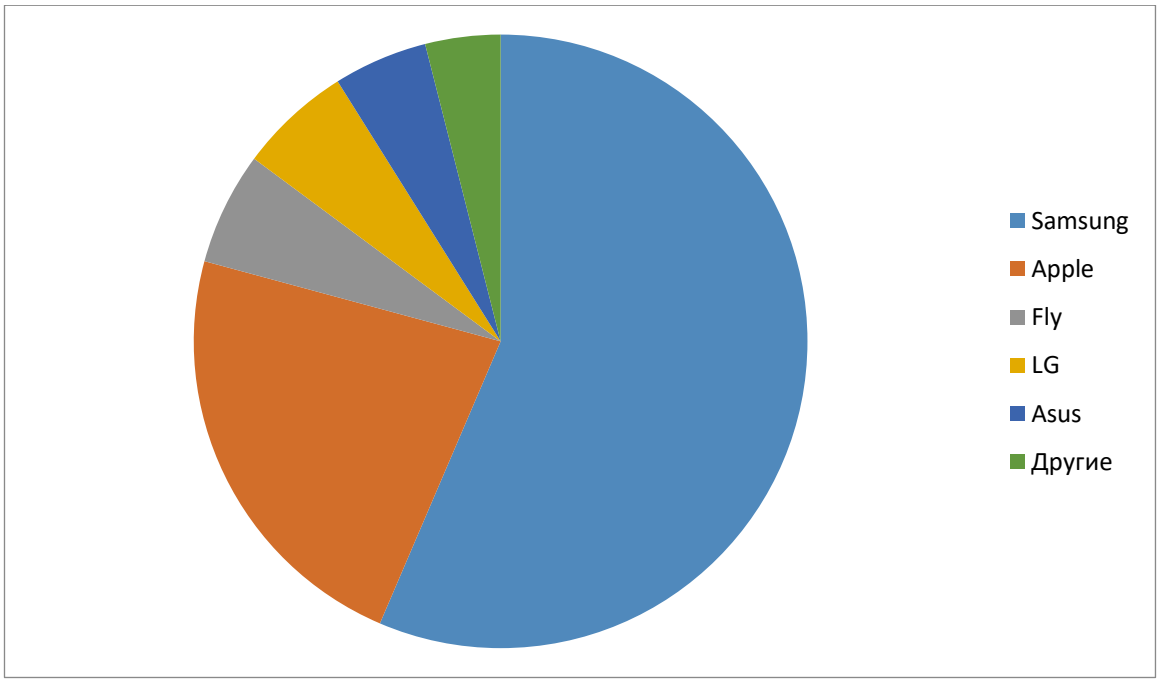

Рис.15. Международные бренды.

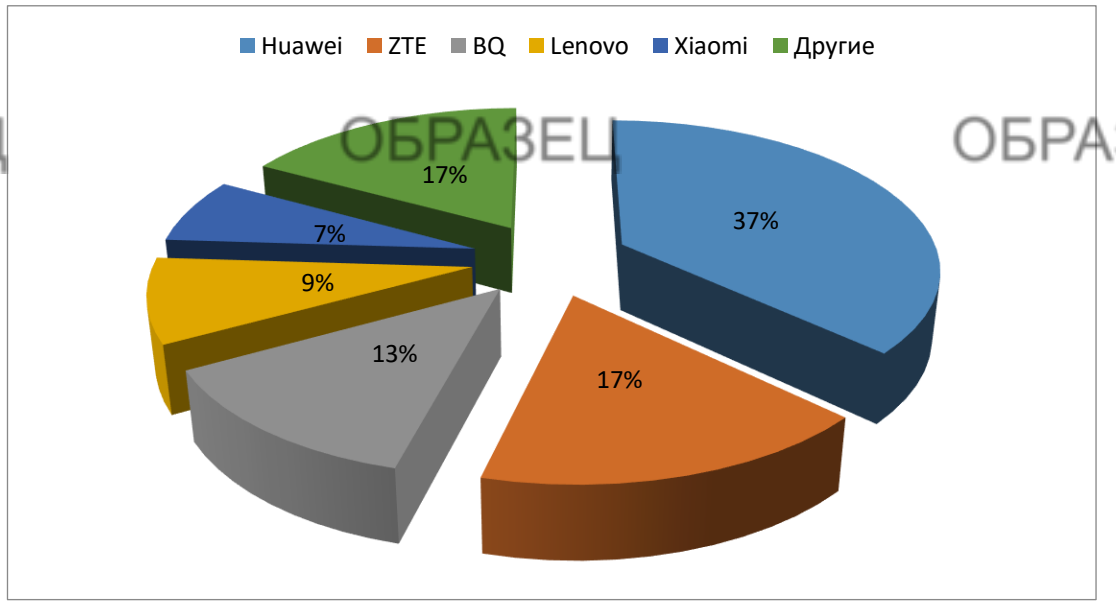

Рис. 16. Китайские бренды.

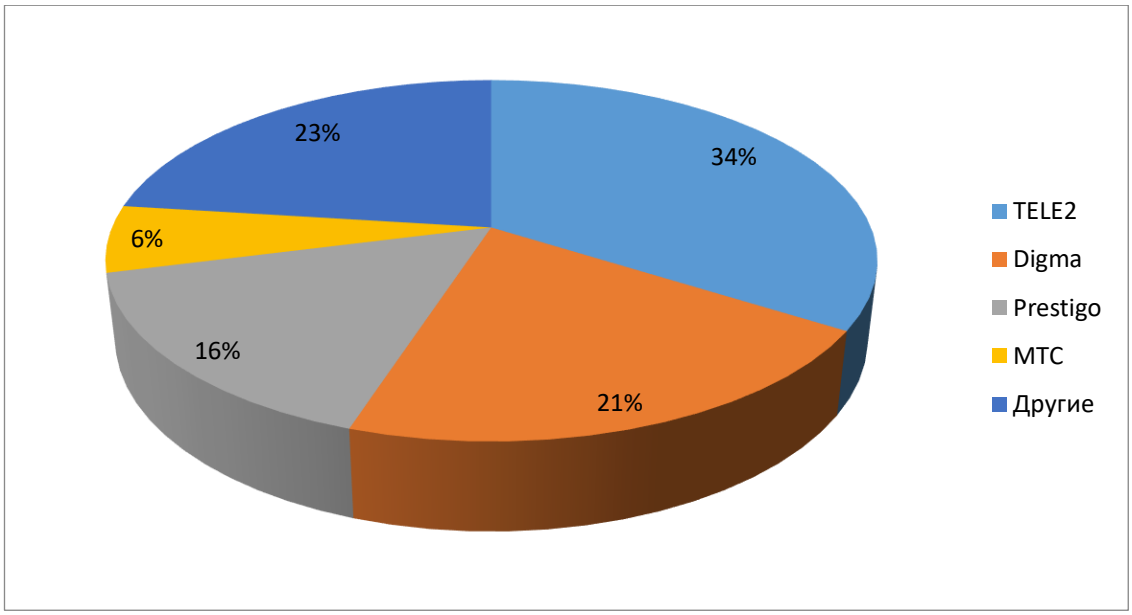

Рис.17. Локальные бренды

Мобильные устройства на платформе Андроид занимают почти 90% всего рынка в России. То есть таких устройств значительно больше.

Сложность разработки на обеих платформах существенно не различается.

Есть мнение, что пользователи iOS платежеспособнее, чем Андроид поэтому большинство проектов разрабатывается сначала на iOS, но разрабатываемая система не будут иметь платного функционала.

Таким образом, после проведенного анализа возможностей операционных систем и рынка мобильных устройств, системой для приложения выбран Android.

Выбор языка программирования

Языки программирования – это язык программистов, который используется для разработки программного обеспечения или других наборов инструкций и алгоритмов.

Языки программирования делятся на два типа: языки программирования низкого и высокого уровня. В нашем исследовании рассматриваются языки программирования высокого уровня, это такие языки как Java, C++, C#, Python, Pascal и др.

Основная черта языков высокого уровня — это введение конструкций, описывающих структуры данных и операции над ними, описания которых на другом низкоуровневом языке программирования длинные и сложные для понимания. Далее мы рассмотрим основные языки программирования.

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Приложения Java транслируются в специальный байт-код, благодаря чему приложения работают на любой компьютерной архитектуре с помощью виртуальной Java-машины.

Kotlin —типизированный язык программирования, работающий поверх JVM, разработанный компанией JetBrains. Компилируется в JavaScript и на другие платформы через инфраструктуру LLVM.

Хронология создания языка:

2010 год – начало разработки языка.

2011 год – язык представлен общественности.

Февраль 2012 года – открыт исходный код реализации языка и представлен плагин для IDEA.

Июнь 2012 года – поддержка Андроид.

Декабрь 2012 года – поддержка JAVA 7.

Февраль 2016 года – релиз-кандидат версия 1.0.

Май 2022 года - Google сообщила, что инструменты языка Kotlin, основанные на JetBrains IDE, будут включены в Android Studio 3.0 (официальный инструмент разработки для ОС Android).

Сравнение Kotlin и Java

Некоторые проблемы Java, решённые в Kotlin:

Нет сырых (raw) типов.

Массивы в Kotlin инвариантны.

Kotlin имеет правильные функциональные типы и поддерживает их использование вместо SAM-типов из Java.

Ссылки на null под контролем системой типов.

Вариативность на месте использования без подстановочных символов.

В Kotlin нет проверяемых исключений

Что есть в Java, но нет в Kotlin

Проверяемые исключения.

Примитивные типы, которые не являются классами.

Статичные члены.

Не приватные поля.

Подстановочные символы (маски, wildcards).

Что есть в Kotlin, но нет в Java

Производительные и контролируемые пользовательские структуры за счет лямбда-выражения и inline-функций.

Функции расширения.

Null безопасность.

Умные приведения.

Строковые шаблоны.

Свойства.

Первичный конструктор.

Делегирование на уровне языка.

Выведение типа для переменных и свойств.

Синглтоны на уровне языка.

Вариативность на уровне объявления и Проекции типов.

Интервалы.

Перегрузка операторов.

Вспомогательные объекты.

Классы данных.

Раздельные интерфейсы для изменяемых и неизменяемых коллекций.

Сопрограммы (корутины).

Сравнив Java и Kotlin можно прийти к выводу, что оба языка обладают достоинствами и недостатками. Java хорошо зарекомендовал себя, на нем написано множество приложений и не только для мобильных. Kotlin совсем новый язык, но уже набрал популярность среди разработчиков, а также он

решает множество задач, которые из-за совместимости старых приложений будет довольно сложно решить в Java. После анализа выбор был сделан в сторону Kotlin.

Анализ баз данных для мобильных приложений

SQLite – это встраиваемая библиотека, ее особенностями является движок базы, интерфейс в пределах одной библиотеки, возможность хранить всю информацию в одном файле. Дополнительно скорость работы обеспечивает высокоупорядоченная внутренняя архитектура и устранение необходимости в соединениях «сервер-клиент» и «клиент-сервер».

SQLite может работать в крошечном объёме выделяемой для неё памяти, гораздо меньшем, чем в других системах БД, благодаря использованию эффективной инфраструктуры. Это делает эту встраиваемую библиотеку удобным инструментом с возможностью использования практически в любых задачах.

Преимущества технологии:

Очень популярна.

Очень надежна.

Консольная утилита для работы с базами.

Открытые исходные коды.

Недостатки.

Нет хранимых процедур.

Нет встроенной поддержки UNICODE.

Не подходит для приложений, которые часто обращаются в базу.

SQLite поддерживает динамическое типизирование данных.

Возможные типы полей – Integer, Real;Text; Blob.

SharedPreferences —хранилище, которое используется для хранения настроек приложений. Данное хранилище относительно постоянное, поэтому пользователь может зайти в настройки приложения и очистить его данные, при этом отчистив данные и в хранилище.

Для работы с данными постоянного хранилища нам понадобится экземпляр класса SharedPreferences, который можно получить у любого объекта, унаследованного от класса android.content.Context (например, Activity или Service). У объектов этих классов (унаследованных от Context) есть метод getSharedPreferences.

Realm — кросс-платформенная база данных для мобильных устройств iOS и Android. Realm не только лучше и быстрее чем SQLite и Core Data, она значительно проще в использовании. Realm распространяется бесплатно, и не имеет ограничения на использование. За последнее десять лет не было представлено ничего нового в сфере мобильных базах данных, но Realm изменил ситуацию. Сейчас, для работы с мобильной базой данных, есть только SQLite и обертки над ним. Realm не является ORM, поэтому она проста в использовании и у нее свой собственный механизм персистентности для большей производительности и скорости выполнения.

Особенности Realm:

Проста в установке: это простая зависимая библиотека;

Скорость: быстрее, чем SQLite и CoreData;

Кросс-платформенная: простая масштабируемость и быстрая барота с большими обьемами данных;

Документация;

Надежность;

Стоимость: абсолютно бесплатна.

Сравнительный анализ баз данных показал, что лучшим выбором будет Realm. Он прост в освоении в отличии от SQLite и намного надежнее, чем SharedPreferences.

Node.js

Node.js — программная платформа. Разработана на движке V8, который транслирует JavaScript в машинный код. Движок V8 делает из JavaScript язык общего назначения.

Node.js позволяет JavaScript работать с устройствами ввода-вывода благодаря собственному API, который подключает сторонние библиотеки, реализованные на различных языках. В основном Node.js применяется на сервере, выполняя роль веб-сервера, но есть другие возможности такие как обычные приложения для Linux, Windows и macOS или программирование микроконтроллеров. Основой Node.js асинхронные и событийно-ориентированные выводы и программирование с неблокирующим вводом/выводом.

С помощью Node.js реализован веб-сервер для заполнения расписания и передачи данных для мобильного приложения по средствам HTTP протокола.

3.3 Анализ интегрированной среды разработки

Для продуктивной разработки Android-приложений, используются гибкие средства проектирования, разработки и оптимизированный программный код, поэтому важным этапом создания приложения является выбор эффективной интегрированной среды разработки (integrated development environments, IDE).

Android Studio — это среда разработки для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O.

Android Studio стала официальным средством разработки Android приложений. Среда разработки доступна для Windows, OS X и Linux. В мае 2022 года Google анонсировал язык Kotlin, используемый в Android Studio, который стал официальным языком программирования для платформы Android наряду с Java и C++.

Eclipse — интегрированная среда разработки модульных кроссплатформенных приложений.

Наиболее известные приложения на основе данной интегрированной среды — «Eclipse IDE» для разработки ПО на множестве языков.

На сегодняшний момент использование Android Studio будет более правильным решение так как его поддерживает Google, который владеет Android.

Библиотеки зависимости

NPM

Пакетом в Node.js называется один или несколько JavaScript-файлов, состоит из библиотеки или инструмента с дополнительным функционалом.

Node Package Manager или npm входит в стандартный пакет поставки Node.js и устанавливается автоматически. При использовании он устанавливает дополнительные библиотеки с сервера npm.

Bcrypt — метод создания ключа на основе адаптивных и криптографических технологий. Используется для защищенного хранения паролей. Bcryptjs - npm библиотека использующая bcrypt. Является наиболее быстрой системой хеширования паролей.

Dust - шаблонизатор разработанный компанией LinkedIn. Предназначен для работы с HTML страницами в асинхронном режиме на сервере и в браузере.

Express - веб-фреймворк для Node.js, имеющий в своем составе функции для мобильных и веб-приложений. В состав API библиотеки express входят все методы HTTP и промежуточные обработчики.

Node-postgres – библиотека для работы с СУБД PostgreSQL. Поддерживает обратные вызовы, обещания, асинхронность / ожидание, пул соединений, подготовленные операторы, курсоры, результаты потоковой.

Socket.IO - Node.js библиотека дает возможность выполнять обмен данными между клиентом и сервером в реальном времени. Библиотека использует протокол веб-сокетов, который поддерживается браузером по умолчанию.

Bootstrap - набор HTML и CSS инструментов и шаблонов разработанный компанией Twitter, используется для верстки и наиболее эффективного и срочного создания сайтов и веб-приложений. Bootstrap регулярно

обновляется, поэтому почти все его функции корректно работают в современных браузерах.

Описание основных методов и структур данных

Исходный код приложения представлен в Приложении 2. Далее будут описаны наиболее интересные части программного кода.

Создание приложения начинается с объявления библиотек зависимостей и создания сервера.

```
...
const app = express();
const server = require('http').Server(app);
...
```

Параметр app это основная навигация приложения отвечает, как будут обрабатываться события ввода адреса в строку браузера. Запуск приложения выполняется функцией listen первым параметром она принимает порт, на котором будет запущен сервер, а вторым функцию, которая выполнится после успешного запуска сервера.


```
server.listen(1515, function(){
 console.log('Server Started On Port 1515');
});
```

Функция входа на главную страницу. При выполнении данной функции происходит переход на главную страницу. Метод app.get выполняет http GET запрос, ‘/’ указывает по какому пути обращается браузер, checkAuth проверяет авторизацию пользователя, если эта функция успешно выполнится функция продолжит свое выполнение и покажет главную страницу, иначе перейдет на страницу авторизации.

```
app.get('/', checkAuth, async (request, response) => {
 try{
 const poolClient = await pool.connect()
 const dates = await poolClient.query(`
 SELECT date
```

```
FROM dates
GROUP BY date
`);
poolClient.release();


...
response.render('index', {weeks: weeksArray});

} catch(err){
console.log(err);
}
});
```

Внутри функции выполнятся запрос к базе данных в результате, которого программа получает дни, когда проходят занятия. Вызов происходит асинхронно поэтому программа не блокирует пользовательский потом и пользователь не видит никаких замедлений в работе программы. Так как вызов асинхронный сама функция помещена в конструкцию try-catch поэтому в случае ошибки программа не будет закрыта.

После успешного выполнения запроса программа визуализирует код в привычное для пользователя изображение с помощью команды response.render, где первый параметр это представление библиотеки dust.js которое необходимо показать, а второе параметры которые передаются представлению.

Dust файл подструктуре HTML страница с инъекциями параметров и функций. После передачи этих параметров в функции response.render они становятся доступны в index.dust. Так как параметр weeks это массив в dust файле он выполнит цикл со всеми своими элементами и построил список дней на странице, также параметры устанавливаются внутри кнопки, которая выполняет функцию перехода на расписание выбранного дня.

```
{#weeks}
 <div class="card">
 <div class="card-header">
 <h4>{friday_ru}</h4>
 <h4>{saturday_ru}</h4>
 </div>
 <div class="card-body">
 <form action="/day/2015/{friday}&{saturday}">
 <button
 class="btn btn-success"
 type="submit"
 data-friday="{friday}"
 data-saturday="{saturday}">
 Открыть
 </button>
 </form>
 </div>
 </div>
 {/weeks}
```


Функция checkAuth получает в качестве параметров request, response и next. В теле функции библиотека passport.js выполняет аутентификацию пользователя. Качестве формирования токена выступает библиотека jsonwebtoken.js создает и проверяет токен клиента. Далее идет условие если есть ошибки при проверке данных пользователя выполняется переход на страницу авторизации, иначе система передает токен в request и продолжает выполнение загрузки нужной страницы.

```
function checkAuth (req, res, next) {
```

```
passport.authenticate('jwt', { session: false }, (err, decryptToken, jwtError)
=> {
if(jwtError != void(0) || err != void(0)) return res.render('chat', { error: err ||
jwtError});
req.user = decryptToken;

next();
})(req, res, next);
}
```

Функция createUser создает нового пользователя. Она получает пользовательские данные указанные в форме ввода. Пароль кэшируется с помощью библиотеки bcrypt и хранится в базе в именинном виде.

```
async function createUser(username, password){
try {
const passwordHash = bcrypt.hashSync(password, 12);
...
} catch(err){
console.log(err);
}
});
```


При авторизации система ищет пользователя по имени, если находит в базе, по проверяет совпадает ли введенный пароль с тем что есть в базе. Если операция прошла успешна пользователю присваивается токен, помещая его в cookie и система продолжает загрузку страницы.

```
app.post('/login', async (req, res) => {
try {
let user = await findUser({$regex: _.escapeRegExp(req.body.username),
$options: "i"});
```

```
        if(user != null && bcrypt.compareSync(req.body.password, user.password))
{
        const token = createToken({id: user.id, username: user.name});
        res.cookie('token', token, {
        httpOnly: true
        });

        res.status(200).send({message: "Успешная Авторизация"});
        } else res.status(400).send({message: "Данные пользователя не
корректны"});
        } catch (e) {
        console.error("E, login,", e);
        res.status(500).send({message: "some error"});
        }
        });
```

Заключение

Выполнив анализ предметной области, были выявлены классы-сущности, составлена схема сущностей-классов. Спроектирована USE CASE диаграмма, выделены два актора Студент и Преподаватель, а также семь прецедентов. Сформированы диаграммы деятельности «Авторизации в приложении», «Просмотр расписания», «Установить отметку об отсутствии». Составлено техническое задание на разрабатываемую систему.

В результате анализа сред программирования были выбраны: целевая операционная, под которую было разработано приложение, IDE для разработки и база данных для хранения данных.

По результатам разработки было создано мобильное приложения расписания занятий с возможностью просмотра расписания своей группа, выборкой расписания по преподавателям, чата между преподавателями и группами студентов в рамках занятия для быстрого обмена информацией.

Лишь приложения делают операционные системы пригодной для работы, развлечения, просмотра веб-страниц и иного использования, благодаря чему телефон по праву считается маленьким карманным компьютером.

Из перспективы развития приложения можно выделить дальнейшее совершенствование приложения и добавление в него новых функциональных возможностей таких как передача файлов в чатах. Приложение может быть разработано для всех платформ, то есть кросс программы, или персонально для каждой системы. Как показывает практика, возможности индивидуального ПО гораздо шире, что делает целесообразнее устанавливать соответствующее обеспечение, а следовательно, и разрабатывать необходимые приложения. Также возможно внедрение автоматизированного составления расписания.

Список литературы

1. Аллан, А. Программирование для мобильных устройств на iOS: Профессиональная разработка приложений для iPhone, iPad, and iPod Touch / А. Аллан.. - СПб.: Питер, 2013. - 416 с.

2. Аникеев, С.В. Разработка приложений баз данных в Delphi: Самоучитель / С.В. Аникеев, А.В. Маркин. - М.: ДИАЛОГ-МИФИ, 2013. - 160 с.

3. Афоничкин, А.И. Разработка бизнес-приложений в экономике на базе MS Excel. / А.И. Афоничкин. - М.: Диалог-МИФИ, 2003. - 416 с.

4. Болотнов, А.М. Разработка программных приложений в среде BlackBox: Учебное пособие / А.М. Болотнов. - СПб.: Лань, 2018. - 144 с.

5. Брюс, Е.К. Windows Mobile. Разработка приложений для КПК / Е.К. Брюс. - М.: ДМК, 2011. - 352 с.

6. Васильев, А.Е. Микроконтроллеры. Разработка встраиваемых приложений / А.Е. Васильев. - СПб.: BHV, 2008. - 304 с.

7. Васильев, А.Е. Микроконтроллеры. Разработка встраиваимых приложений / А.Е. Васильев. - СПб.: BHV, 2012. - 304 с.

8. Веллинг, Л. Разработка Web-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон. - М.: Вильямс, 2013. - 848 с.

9. Веллинг, Л. Разработка Web-приложений с помощью PHP и MySQL / Л. Веллинг. - М.: Вильямс, 2006. - 880 с.

10. Гарнаев, А. Мастер Visual Basic.NET. Разработка приложений / А. Гарнаев. - СПб.: BHV, 2002. - 624 с.

11. Гарнаев, А. Visual Basic.NET: разработка приложений / А. Гарнаев. - СПб.: BHV, 2002. - 624 с.

12. Гецманн, П. Разработка приложений для Windows Phone. Архитектура, фреймворки, API / П. Гецманн. - СПб.: BHV, 2014. - 880 с.

13. Глейзер, Дж. Многопользовательские игры. Разработка сетевых приложений / Дж. Глейзер, С. Мадхав. - СПб.: Питер, 2019. - 152 с.

14. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. - М.: ДМК, 2014. - 272 с.

15. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. - М.: ДМК, 2016. - 272 с.

16. Дари, К. AJAX и PHP. Разработка динамических веб-приложений / К. Дари, Б. Бринзаре и др. - М.: Символ, 2015. - 336 с.

17. Дари, К. AJAX и PHP. Разработка динамических веб-приложений / К. Дари, Б. Бринзаре, Ф. Черчез-Тоза, М. Бусика. - СПб.: Символ-плюс, 2015. - 336 с.

18. Дари, К. AJAX и PHP. Разработка динамических веб-приложений / К. Дари, Б. Бринзаре. - М.: Символ-Плюс, 2007. - 336 с.

19. Джонсон, Г. Разработка клиентских веб-приложенийи на платформе. NET Framework: экзамен 70-528 / Г. Джонсон. - М.: Русская редакция, 2008. - 768 с.

20. Дронов, В.А. Windows 8: разработка Metro-приложений для мобильных устройств / В.А. Дронов. - СПб.: BHV, 2012. - 516 с.

21. Дронов, В.А. Windows 8: разработка Metro-приложений для мобильных устройств. / В.А. Дронов. - СПб.: BHV, 2013. - 528 с.

22. Дэйв, М. iOS 5 SDK. Разработка приложений для iPhone, iPad и iPod touch / М. Дэйв, Н. Джек. - М.: Вильямс, 2012. - 672 с.

23. Дэрси, Л. Разработка приложений для Android-устройств. Т. 1: Базовые принципы / Л. Дэрси, Ш. Кондер. - М.: Лори, 2014. - 402 с.

24. Есенин, С.А. DirectX и Delphi: разработка графических и мультимедийных приложений / С.А. Есенин. - СПб.: BHV, 2006. - 512 с.

25. Заяц, А.М. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и node.js: Учебное пособие / А.М. Заяц, Н.П. Васильев. - СПб.: Лань, 2019. - 120 с.

26. Здзиарски, Д. iPhone. Разработка приложений с открытым кодом / Д. Здзиарски. - СПб.: BHV, 2013. - 368 с.

27. Здзиарски, Д. iPhone SDK. Разработка приложений / Д. Здзиарски. - СПб.: BHV, 2013. - 512 с.

28. Здзиарски, Дж. iPhone. Разработка приложений с открытым кодом / Дж. Здзиарски. - СПб.: BHV, 2013. - 368 с.

29. Здзиарски, Дж. iPhone. Разработка приложений с открытым кодом / Дж. Здзиарски. - СПб.: BHV, 2009. - 368 с.