**РОСЖЕЛДОР**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

|  |  |  |  |
| --- | --- | --- | --- |
| **К защите:** |  |  | |
| **Заведующий кафедрой** | **Информационные** | |  |
| **технологии транспорта** | | |  |
|  | д-р техн. наук, проф. | |  |
|  |  | В. И. Хабаров | |
| *подпись* |  | *инициалы, фамилия* | |
|  |  |  | |
| *дата* |  |  | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тема:** | Разработка мобильного приложения «Расписание занятий» | | | | | |
|  |  | | | | | |
|  | |  | БР.БПИ.14.2022 |  |  |  |
|  | |  | *шифр документа* |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Выполнил** |  |  |  | **Руководитель** |
|  |  | А. А. Чурилов |  |  |  | канд. техн. наук, доц.  В. Г. Кобылянский |
| *подпись* |  | *инициалы, фамилия* |  | *подпись* |  | *инициалы, фамилия* |
|  |  |  |  |  |  |  |
| *дата* |  |  |  | *дата* |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Консультанты по разделам** |  |  |  |  |
| 1 |  |  |  |  |
|  |  |  |  |  |
| 2 |  |  |  |  |
|  |  |  |  |  |
| 3 |  |  |  |  |
|  |  |  |  |  |
| Нормоконтролер работы |  |  |  | ст. преп.  Т. А. Распопина |
|  |  | *подпись* |  |  |
|  |  |  |  |  |
|  |  | *дата* |  |  |

**2022 г.**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес-информатики

Кафедра: Информационные технологии транспорта

Направление: 09.03.03 «Прикладная информатика»

Профиль: Корпоративные информационные системы на транспорте

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д-р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| студенту | Чурилову Андрею Алексеевичу | | |
|  |  | | |
| 1. Тема «Разработка мобильного приложения «Расписание занятий» утверждена приказом № 203/с от «30» мая 2022 г. | | | |
| 2. Задание выдано «12» мая 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «20» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно-пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 2 | 13.05.2022 |
| Анализ предметной области | | 9 | 13.05.2022 |
| Проектирование информационной системы | | 26 | 28.05.2022 |
| Руководство пользователя | | 5 | 01.06.2022 |
| Заключение | | 1 | 02.06.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  слайдов | График  (сроки)  выполнения |
| Презентация PowerPoint | 15 | 07.06.2022 |

7. Консультанты по разделам

|  |  |  |
| --- | --- | --- |
| Наименование  раздела | Фамилия, И. О.  консультанта | Подпись консультанта,  дата выдачи задания |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | В. Г. Кобылянский |
|  | *(подпись, фамилия, И.О.)* |  |
| Задание к использованию принял |  | А. А. Чурилов |
|  | *(подпись студента)* |  |

УДК 004.41

**АННОТАЦИЯ**

В работе 51 страница, 36 рисунков, 1 таблица, 15 источников, 1 приложение.

Ключевые слова: *мобильное приложение, система, факультет, база данных, расписание*.

Предметная область – отображения расписания факультета и упрощение его просмотра. Приложение работает с базой данных, откуда берет информацию о расписании занятий и заметках. Мобильное приложение работает на платформе Android.

**ABSTRACT**

The work contains 51 pages, 36 figures, 1 table, 15 sources, 1 attachment.

Keywords: *a mobile application, a system, faculty, a database, schedule.*

Subject area: displaying the faculty schedule and simplifying its viewing. The application works with a database, from where it takes information about the schedule of classes and notes. The mobile application runs on the Android platform.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

ИС - информационная система.

ОС - операционная система.

БД - база данных.

Android - мобильная операционная система компании Google.

Visual Studio Xamarin – среда разработки для создания приложений для операционной системы Android.

C# - объектно-ориентированный язык программирования высокого уровня.

SQL (Structured Query Language) - декларативный язык программирования, применяемый для управления данными в реляционной базе данных.

API - описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

HTTP - протокол прикладного уровня передачи данных.

REST - архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

MVVM - шаблон проектирования архитектуры приложения.

БПИ - название групп прикладной информатики.

БИСТ - название групп информационных систем и технологий.

МЛ - название групп менеджмента.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc106152963)

[1 Анализ предметной области 8](#_Toc106152964)

[1.1 Общее положение 8](#_Toc106152965)

[1.2 Описание проблемы и постановка задачи 10](#_Toc106152966)

[1.3 Обзор существующих систем отображения расписания 11](#_Toc106152967)

[1.4 Обзор существующих систем отображения расписания 16](#_Toc106152968)

[1.5 Выводы к разделу 17](#_Toc106152969)

[2 Проектирование информационной системы 20](#_Toc106152970)

[2.1 Описание бизнес-процессов 20](#_Toc106152971)

[2.2 Общая архитектура системы 26](#_Toc106152972)

[2.3 Обоснование выбора среды разработки и языка программирования 28](#_Toc106152973)

[2.4 Проектирование мобильного приложения 32](#_Toc106152974)

[2.5 Описание интерфейса информационной системы 38](#_Toc106152975)

[2.6 Выводы к разделу 44](#_Toc106152976)

[3 Руководство пользователя 45](#_Toc106152977)

[Список использованных источников 50](#_Toc106152978)

[Приложение А Код программы](#_Toc106152978) 52

# **ВВЕДЕНИЕ**

В настоящее время автоматизация процессов является чуть ли не самой важной частью любого предприятия. Применение ЭВМ для различных процессов позволяет экономить огромное количество времени и способствует достижению высокой производительности труда. Благодаря высококачественной автоматизации, можно облегчить работу предприятия и производства.

Основной целью автоматизации является: снижение большого количества затрат по времени, труду и материальным ресурсам, ведение и контроль отчетов, поддержание данных в надежном состоянии; очевидное большое количество ошибок и опечаток при проведении выборки необходимых данных и их подготовки к различным документам.

В Новосибирске имеется множество учебных заведений. Как правило, большинство из них используют информационные технологии для упрощения различных процессов происходящих в учебном заведении между студентами и преподавателями. Из этого можно сделать вывод, что для большинства учебных заведений будет практично иметь свою информационную систему, с помощью которой можно будет упростить взаимодействие внутри вуза.

Целью данной ВКР является создание информационной системы, которая в свою очередь должна посодействовать в упрощении отслеживания расписания и актуальной информации о предметах, с помощью хранения и изменения данных для всех видов занятий, которые проводятся в высшем учебном заведении, с помощью мобильного приложения, на основании представленной информации о времени проведении занятий, а также добавление заметок к каждому предмету. Данная система будет работать в рамках определенного факультета и выполнять функции основной площадки для отслеживания занятий и просмотр актуальных заметок на каждом предмете.

Для достижения поставленной цели необходимо выполнить ряд основных задач:

- изучить предметную область;

- сделать анализ рынка по схожей тематике;

- построить модели бизнес-процессов предметной области;

- выбрать и обосновать инструменты разработки приложения;

- разработать информационную систему для факультета учебного заведения;

- настроить и протестировать информационную систему;

- разработать руководство пользователя.

Данная работа посвящена разработке информационной системы.

Практическая значимость выпускной квалификационной работы заключается в том, что разработанная информационная система «Расписание факультета» позволит значительно упростить процесс отслеживания расписания, сэкономить время студента и преподавателя, а также даст возможность облегчить задачи информационно-управленческого характера.

**1 Анализ предметной области**

**1.1 Общее положение**

Объектом автоматизации является просмотр расписания факультета образовательного учреждения «ФБИ». Главной целью данного факультета является обучение студентов по выбранному ими направлению «БПИ», «БИСТ», «МЛ».

В Учебном заведении имеется бюро расписания, в котором разрабатывается расписание для каждого факультета на каждый отдельный семестр. После этого, бюро расписания передает данные о составленном расписании каждому факультету. Расписание составляется по всем требованиям, которые будут требоваться от конкретного факультета.

Обучение на факультете можно разделить на два вида занятий: теория (прослушивание лекций в аудитории) и практика (применение полученных знаний в процессе выполнения лабораторных работ). Для обучения могут использоваться разные аудитории и корпусы учебного заведения. После того, как студент защитит выпускную квалификационную работу, ему выдаётся диплом. Критерии ВКР могут измениться с прибавлением новых правил в соответствии с потребностями факультета, или появлением новых государственных стандартов.

У факультетов имеется свои кафедры, к которым относятся определенные преподаватели и аудитории, которые оснащены специальным оборудованием, с помощью которого можно оттачивать навыки, полученные во время лекционных занятий. К такому оборудованию относятся: компьютеры, маршрутизаторы, роутеры и много другое. В каждой аудитории имеется компьютер, доска, проектор, парты со стульями или скамьями, стенды с полезной информацией и многие другие вещи для обучения.

Всё это очень важно для факультета, потому что оно даёт надлежащую конкуренцию по обучению среди факультетов и вузов, предоставляемых на рынке учебных заведений.

Основными задачами факультета являются:

- обеспечение высокой конкурентоспособной позиции в отечественной образовательной и научной среде в области подготовки высококвалифицированных специалистов по техническим специальностям и направлениям подготовки;

- ориентация на текущие потребности национальной экономики с учетом мировых тенденций;

- повышение качества учебного процесса на основе внедрения инновационных методов обучения и современных дистанционных, телекоммуникационных и компьютерных технологий в образовательный процесс;

- развитие научно-исследовательской работы студентов и аспирантов;

- обеспечение высокого уровня востребованности на рынке труда выпускников факультета информационно-технического сервиса.

Для обеспечения поставленных задач, факультет осуществляет следующие функции:

- разработать и реализовать планы по привлечению абитуриентов на образовательные программы факультета;

- провести анализ перечня направлений и специальностей подготовки, указанных в лицензии университета на предмет актуальности их реализации;

- разработать и реализовать основные образовательные программы, направленные на получение прикладных квалификаций;

- использовать индивидуальный адресный подход в организации работы со студентами;

- разработка новых методов обучения для повышения качества общего образования студентов;

- обеспечение студентов необходимым оборудованием;

- контроль программы, удовлетворяющий современные стандарты обучения студентов.

**1.2 Описание проблемы и постановка задачи**

Выпускная квалификационная работа направлена на создание прототипа информационной системы для факультета, для того чтобы упростить процесс просмотра расписания. Эта проблема является актуальной потому, что в настоящее время, процесс отслеживания расписания проблемно осуществить с помощью мобильного телефона, для студента или преподавателя. Лист с расписанием вмещает пары на весь семестр, на котором невозможно удобно и оперативно просмотреть занятия, из-за его громоздкости. Пример отображения листа с расписанием представлен на рисунке 1.1.

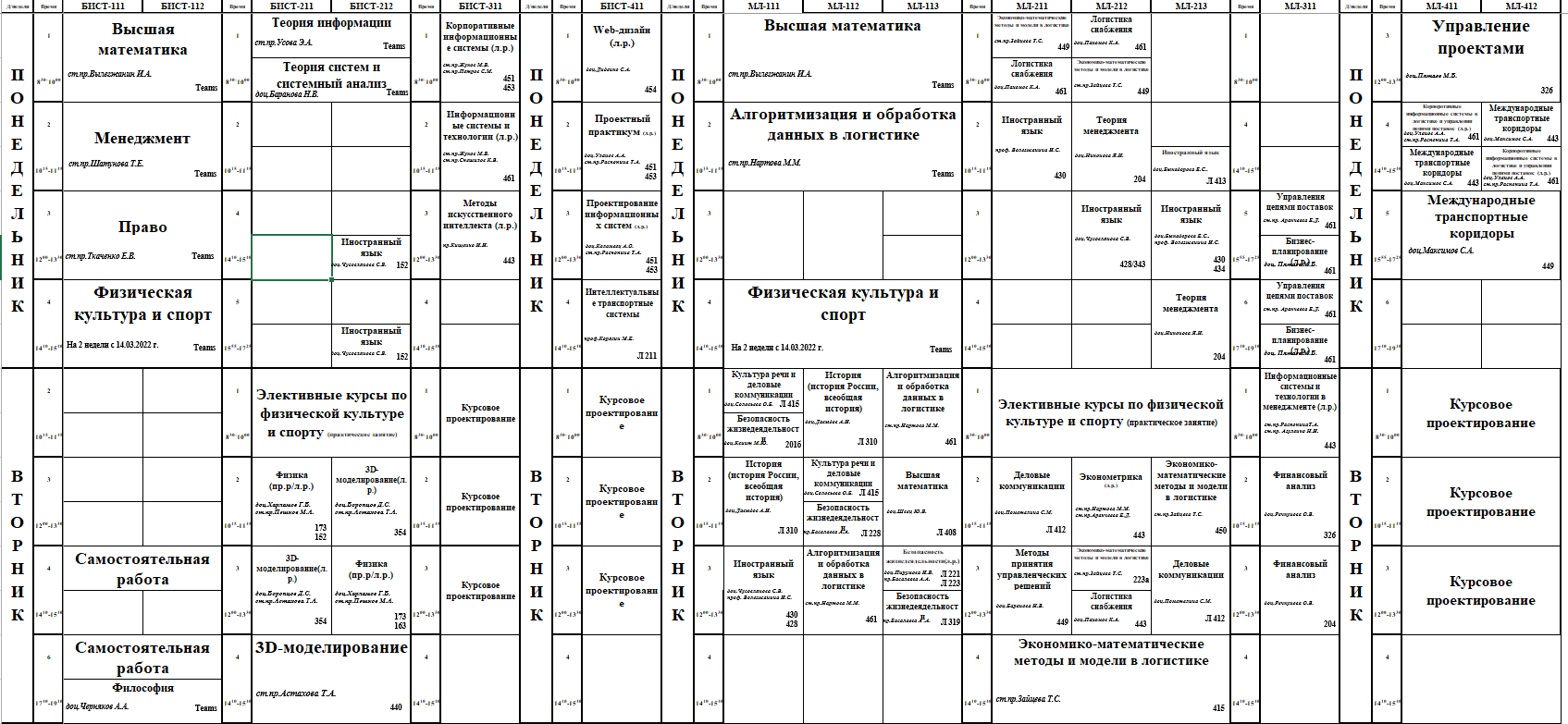


Рисунок 1.1 - Пример расписания

Если рассматривать с точки зрения затраченного времени, то этот способ является не рациональным, так как каждый раз, когда нужно узнать расписание или уточнить, в какой аудитории идет занятие, необходимо искать группу или предмет в большой таблице, что может занять очень продолжительное количество времени.

Рассмотрим алгоритм просмотра расписания студентом или преподавателем:

- студенту или преподавателю, следует потратить время, для того чтобы зайти на сайт, найти свой факультет и там скачать файл с расписанием;

- после чего необходимо найти свою группу в большой таблице, а также нужный день и занятие.

Этот алгоритм не комфортен для постоянного использования в качестве просмотра занятий и информации о предметах, также сам файл может потеряться на телефоне, либо в само расписание могут внести правки, и если их не запомнить, то студент или преподаватель может путаться.

Следовательно, можно сделать вывод о необходимости приложения, в котором будут выполнены следующие фрагменты:

- с помощью мобильного приложения, пользователь всегда будет знать, где просмотреть расписание;

- облегчение просмотра расписание;

- контроль и быстрое обновление со стороны факультета;

- возможность оставлять заметки на предметах.

**1.3 Обзор существующих систем отображения расписания**

В настоящее время проблема составления расписания в вузе актуальна как никогда. Это связано с появлением новых направлений, различных видов аудиторий для лекционных, практических и лабораторных занятий, большим количеством учебных групп. Справиться с такой нагрузкой методистам становится сложнее с каждым днем. Не единожды были предприняты попытки переложить решение этой проблемы на плечи компьютера, что привело к созданию специальных программ для составления расписания занятий в вузе. Был проведен обзор основных программ, используемых для составления расписания, рассмотрена спецификация каждой из них.

Их можно разделить по их назначению. Одни предназначены для составления только школьного расписания, другие - для создания расписания для вуза, третьи совмещают в себе обе спецификации, предоставляя пользователю возможность составить расписание с учётом особенностей конкретного учебного заведения.

Основными программами для составления расписания в вузе в настоящее время являются: «Ректор-вуз 1.6», «Университет 3.2.0.711», «Расписание занятий 2.1», «Расписание ПРО», «AVTOR-2000», «Аналит: расписание, авторасписание». В ходе написания работы был произведен анализ основных программ для составления расписания, который производился по следующим критериям: выполнение задач составления расписания, необходимость закупки дополнительных программ, платформа, стоимость лицензии на один компьютер, удобство и простота интерфейса.

По результатам проведённого анализа можно сделать следующие выводы. Программа «Расписание занятий 2.1» не отвечает всем критериям, так как выполнена для операционной системы DOS. Система «Расписание ПРО» не является удачной, так как её область функционирования достаточно узка. К недостаткам программы «Университет 3.2.0.711» можно отнести проблематичность установки и достаточно высокую стоимость. «AVTOR-2000» не прошёл отбор, так как его цена неприемлема для среднего вуза. Программа «Аналит: расписание» реализована для 1с8 и требует от пользователя достаточно серьезного уровня подготовки. Программа «Ректор-вуз» для составления расписания занятий в системе высшего профессионального образования версии 1.6 по результатам анализа занимает лидирующую позицию.

Эта программа обладает следующими преимуществами:

- удобный и понятный даже неопытному пользователю интерфейс;

- планирование распределения нагрузок по неделям (причём расписание по разным неделям может быть различным);

- расписание занятий составляется без «окон»;

- в графике работы преподавателя можно запретить первые или вторые пары (не все преподаватели могут приехать рано);

- в программе есть понятие «сложность предмета» и возможность установки «приоритета» на определённые дисциплины (лекция по базовому, сложному предмету в конце дня - напрасно потраченное время, так как способность восприятия слушателей в конце дня крайне низка);

- можно ограничить число занятий в день у преподавателя 12 часами (согласно Трудовому кодексу РФ);

- составление отчётов по распределению нагрузок кафедр и вуза по видам;

- автоматический и ручной режимы составления расписания занятий.

К сожалению, программа не очень удобна в использовании для заочной формы обучения и для вузов, периодичность расписания в которых не одна неделя, а несколько.

Остановимся на работе программы «Ректор-вуз» более подробно. Она состоит из четырёх разделов: «Списки», «Нагрузки», «Расписание» и «Замены». Раздел «Списки» служит для ввода, редактирования и печати списков кафедр, специальностей, групп, дисциплин, аудиторий, преподавателей и видов занятий, как показано на рисунке 1.2.

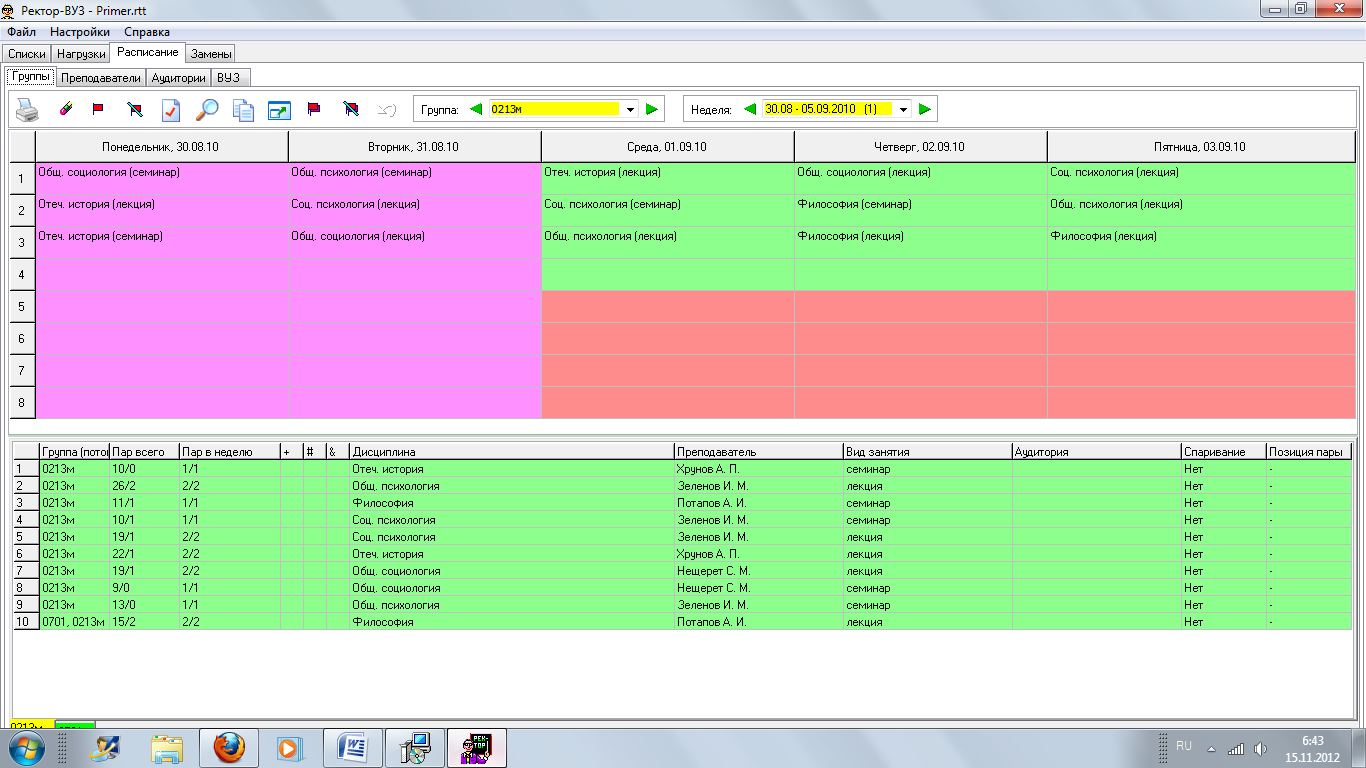


Рисунок 1.2 - Интерфейс программы

Раздел «Нагрузки» используется для ввода, редактирования и печати учебных планов по каждой специальности, нагрузок преподавателей, графиков распределения часов по неделям в пределах семестра, отчётов по загрузке преподавателей, кафедр и вуза в целом представлено на рисунке 1.3.

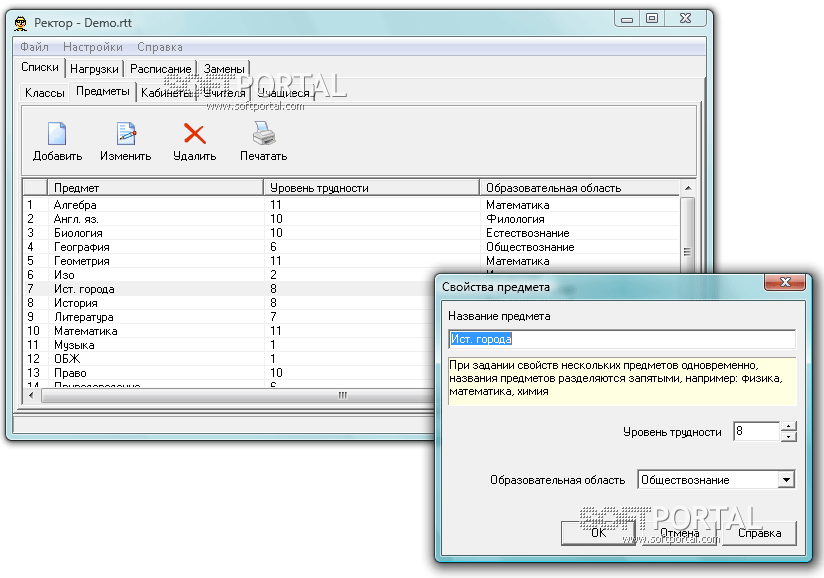


Рисунок 1.3 - Интерфейс раздела «Нагрузки»

Раздел «Расписание» предназначен для составления расписания по группам, преподавателям, аудиториям и вузу в целом, пример показан на рисунке 1.4, но, к сожалению, в программе нет возможности закрепить за преподавателем или группой определённую аудиторию.

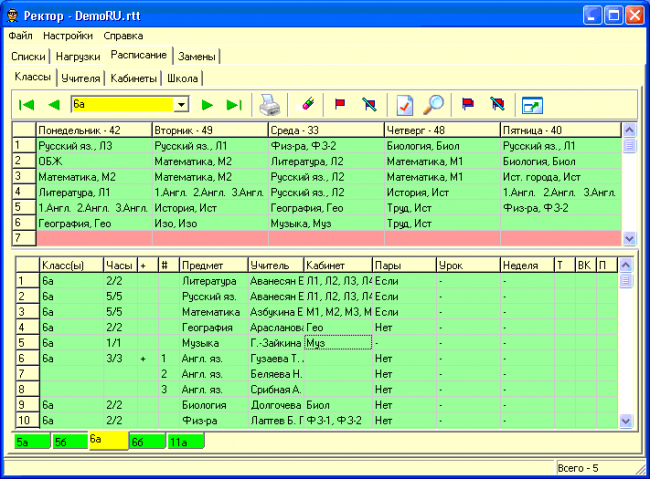


Рисунок 1.4 - Интерфейс раздела «Расписание»

Расписание занятий можно составлять в автоматическом, ручном или комбинированном режимах, при этом переходить от одного рабочего режима к другому можно в любой момент времени. При составлении расписания в автоматическом режиме программа учитывает все сформулированные требования к расписанию. При составлении расписания в ручном режиме программа подсказывает возможные варианты расстановки уроков выбранного преподавателя, возможные варианты заполнения пустых клеток в расписании группы, следит за количеством мест в аудиториях.

Готовое расписание занятий одной, всех или нескольких групп и преподавателей можно сохранить в форматах Microsoft Word, Excel или HTML. Программа «Ректор-вуз» работает на любом компьютере с операционной системой Windows ХР, Windows Vista или Windows 7. Распространение программы условно-бесплатное, однако демоверсия программы «Ректор-вуз», предоставляя возможность составить в ней расписание, не позволяет сохранять его. Лицензия на один компьютер стоит на сегодняшний день 7 579 рублей.

При составлении расписания в программе «Ректор-вуз» у пользователя не возникает никаких сложностей. Работать с ней достаточно удобно. Непростой задачей является проблема контроля ошибок в расписании (случается, что по ошибке назначены занятия в одной и той же аудитории, в одно и то же время у групп различных специальностей и курсов, у одного преподавателя, но в разных группах). Эта проблема в рамках программы «Ректор-вуз» решается с лёгкостью. Она позволяет при помощи специальных функций отслеживать занятость аудиторий, кроме того, при наличии совпадения представляет список свободных аудиторий.

Безусловно, программа «Ректор-вуз» не может полностью самостоятельно составить расписание в вузе, ведь необходимо учитывать огромное количество различных нештатных ситуаций. Однако можно сказать с уверенностью, что программа «Ректор-вуз» существенно облегчает процесс составления расписания, экономит время, а также следит за корректным составлением расписания, не допуская ошибок, по причине человеческого фактора (усталость).

Процесс составления расписания задача не простая, поэтому перед нами стоит задача отображения на мобильных устройствах.

**1.4 Обзор существующих систем отображения расписания**

Примерами решений в области упрощения просмотра расписания является программное обеспечение компании «Кампус».

«Кампус» - является приложением, которое направленно в первую очередь на удобное отображение расписания на мобильном устройстве и упрощение взаимодействия с расписанием, а также оно тесно интегрировано с базой данных сервиса Кампус.

Интерфейс приложения представлен на рисунках 1.5, 1.6.

Единственная функция этого приложения — это отображение расписания на мобильном устройстве. Минусом этого приложения является список учебных заведений, который сильно ограничен.

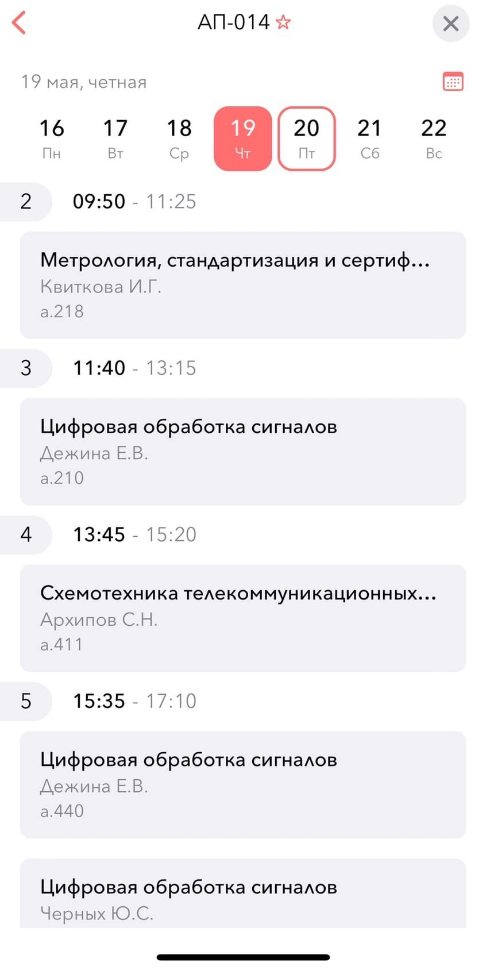


Рисунок 1.5 - Интерфейс мобильного приложения

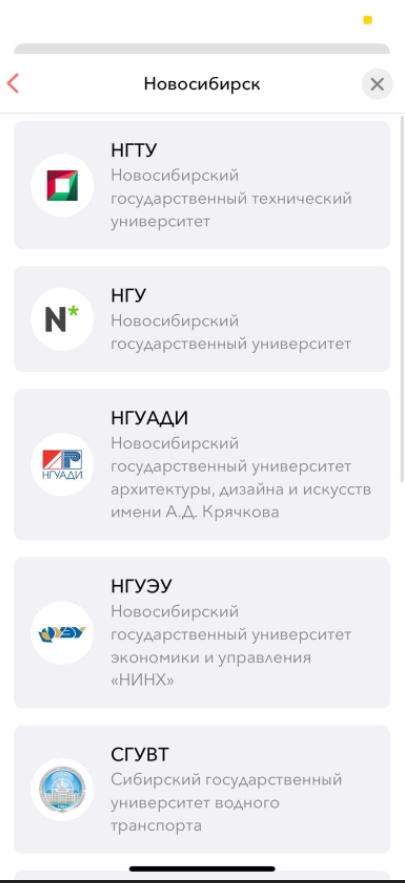


Рисунок 1.6 - Интерфейс при выборе учебного заведения

**1.5 Выводы к разделу**

В первом разделе данной выпускной квалификационной работе:

- проведен анализ предметной области;

- выявлены функции, требующие автоматизации посредством реализации приложения «Расписание»;

- проведен краткий обзор существующих систем автоматизации.

В результате анализа аналогов, было принято решение разработать собственный продукт, так как закрытый исходный код большинства готовых продуктов не дает возможности самостоятельной доработки и расширения функционала системы, что может привести к дополнительным финансовым и временным затратам в процессе внедрения обновлений.

Также данный проект был разделен на две части, одна из которых это разработка базы данных, а вторая, это создание мобильного приложения для отображения информации из этой базы данных. Наличие недостатков объясняет необходимость разработки собственного приложения.

К преимуществам создания собственной системы можно отнести:

- открытый исходный код позволит проводить модификацию, расширение и усложнение системы в зависимости от потребности факультета;

- оперативное обновление расписания, так как приложение принадлежит факультету;

- возможность добавление заметок к предметам.

Приложение с расписанием для студентов и преподавателей - утилита для смартфонов, предназначенная для просмотра расписания занятий для конечных пользователей, включает в себя следующие функциональные возможности:

- просмотр расписания;

- выбор группы;

- редактирование расписания со стороны администратора;

- отображение ссылок на официальный сайт и социальные сети университета;

- возможность добавления объявления в приложение.

Основными целями разработки Системы являются:

- упрощение взаимодействия с расписанием для студентов и преподавателей;

- повышение удобства информирования студентов о изменении расписания.

Основная функция Системы - обеспечение процесса дистанционного просмотра расписания для пользователя, с использованием приложения. Система должна обеспечивать выполнение перечисленных ниже функций (функциональные характеристики):

- круглосуточная доступность и работоспособность системы;

- возможность хранения расписания групп и его отображения по запросу пользователя;

- возможность отображения заметок предмета пользователям.

К нефункциональным требованиям относятся следующие пункты.

Требования к надежности:

- должен быть разработан механизм и регламент полного резервного копирования системы;

- максимальное время остановки системы для проведения технологических операций не более 2 часа в сутки.

Требования к объему хранимой информации:

- система должна хранить всю информацию о расписании групп активных пользователей в течение актуальности расписания.

Требования к производительности:

- количество одновременно работающих пользователей: до 1000 человек;

- время подготовки к работе с момента запуска приложения до запуска отображения расписания не должно превышать 5 секунд;

- время отклика Системы на действия пользователя не должно превышать 2 секунд;

- максимальная продолжительность вычислений должна быть не более 7 секунд.

Требования к численности и квалификации персонала. Для эксплуатации cистемы определены следующие роли:

- пользователь;

- системный администратор.

Минимальные системные требования к клиентской части:

- операционная система: Android 9.0;

- сеть: 5 Мбит/сек;

- процессор: 1 ГГц;

- свободное место: 40 МБ;

- оперативная память: 1 ГБ.

Требования к информационной и программной совместимости:

- система должна быть совместима с любыми операционными системами семейства Android от версии 9.0 и выше.

**2 Проектирование информационной системы**

Проектирование информационных систем - это упорядоченная совокупность методологий и средств создания или модернизации информационных систем.

Перед непосредственной разработкой программного продукта необходимо спроектировать каждый этап, предусмотреть требуемую функциональность системы и степень адаптации к изменяющимся условиям ее использования .

Информационная система должна состоять из трёх частей: базы данных, сервера и связанного с ним мобильного приложения. Таким образом рассматривать также нужно три программных продукта.

## 2.1 Описание бизнес-процессов

Моделирование бизнес-процессов является одним из методов улучшения качества и эффективности работы организации. В основе этого метода лежит описание процесса через различные элементы (действия, данные, события, материалы и пр.) присущие процессу. Как правило, моделирование бизнес-процессов описывает логическую взаимосвязь всех элементов процесса от его начала до завершения в рамках организации. В более сложных ситуациях моделирование может включать в себя внешние по отношению к организации процессы или системы [1], [2].

Моделирование бизнес-процессов позволяет понять работу и провести анализ организации. Это достигается за счет того, что модели могут быть составлены по различным аспектам и уровням управления. В больших организациях моделирование бизнес-процессов выполняется более подробно чем в малых, что связано с большим количеством кросс-функциональных связей.

Обычно для моделирования бизнес-процессов применяются различные компьютерные средства и программное обеспечение. Это облегчает управление моделями, отслеживание в них изменений и позволяет сократить время анализа.

Моделирование бизнес-процессов преследует несколько целей:

- это описания процессов, которое можно проследить, от начала, до завершения;

- моделирование бизнес-процессов задает правила выполнения процессов, то есть то, каким образом они должны быть выполнены. Если следовать установленным в моделях правилам, руководящим указаниям или требованиям, то можно достичь желаемой производительности процессов;

- установление взаимосвязей в процессах для выстраивания связи между процессами и требованиями, которые они должны выполнять.

Моделирование бизнес-процессов, как правило, включает в себя выполнение нескольких последовательных стадий.

Состав стадий, которые включает в себя моделирование бизнес‑процессов, следующий:

- выявление процессов и построение исходной модели «как есть». Для того чтобы улучшить процесс, необходимо понимать, как он работает в данный момент. На этой стадии определяются границы процесса, выявляются его ключевые элементы, собираются данные о работе процесса. В результате создается исходная модель процесса «как есть». Эта модель не всегда адекватно отражает работу процесса, поэтому модель этой стадии можно назвать «первым драфтом» или исходной моделью «как есть»;

- пересмотр, анализ и уточнение исходной модели. На этой стадии выявляются противоречия и дублирование действий в процессе, определяются ограничения процесса, взаимосвязи процесса, устанавливается необходимость изменения процесса. В результате формируется окончательный вариант модели «как есть»;

- разработка модели «как должно быть». После анализа существующей ситуации, необходимо определить желаемое состояние процесса. Это желаемое состояние представляется в модели «как должно быть». Такая модель показывает, как процесс должен выглядеть в будущем, включая все необходимые улучшения. В ходе этой стадии моделирования бизнес‑процессов и разрабатываются такие модели;

- тестирование и применение модели «как должно быть». Эта стадия моделирования связана с внедрением разработанной модели в практику деятельности организации. Модель бизнес-процесса проходит апробацию, и в нее вносятся необходимые изменения;

- улучшение модели «как должно быть». Моделирование бизнес‑процессов не ограничивается только созданием модели «как должно быть». Каждый из процессов по ходу работы продолжает изменяться и совершенствоваться, поэтому модели процессов должны регулярно пересматриваться и улучшаться. Эта стадия моделирования связана с постоянным улучшением процессов и улучшением модели бизнес-процессов.

*Диаграмма вариантов использования.*

Прецедент - возможность моделируемой системы (часть ее функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы - описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- четко отделить систему от ее окружения;

- определить действующих лиц (актеров), их взаимодействие с системой и ожидаемую функциональность системы;

- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Диаграмма вариантов использования состоит из актеров, для которых система производит действие и собственно действия Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case - овалом.

На рисунках 2.1 и 2.2 представлены диаграммы вариантов использования для администратора базы данных, диаграмма функции пользователя.

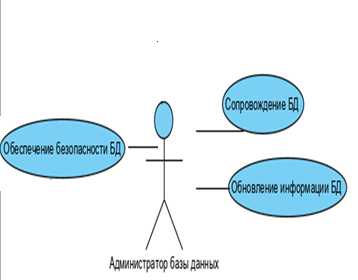


Рисунок 2.1 - Функции администратора базы данных

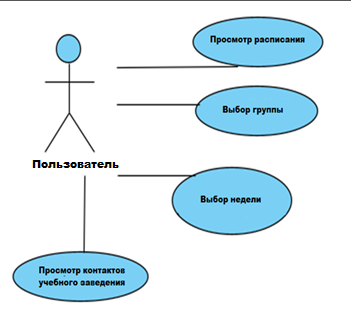


Рисунок 2.2 - Функции пользователя

*Диаграмма деятельности.*

Диаграмма деятельности - еще одна важная диаграмма в UML, описывающая динамические аспекты системы. Диаграмма деятельность - это, по сути, блок-схема, представляющая поток от одного действия к другому. Деятельность может быть описана как работа системы [3]. Поток управления передается от одной операции к другой. Этот поток может быть последовательным, разветвленным или параллельным. Диаграммы действий касаются всех типов управления потоком с использованием различных элементов, таких как fork, join, и многие другие.

Деятельность - это особая операция системы. Диаграммы действий используются не только для визуализации динамической природы системы, но они также используются для построения исполняемой системы с использование прямого и обратного проектирования. Единственная недостающая вещь на диаграмме активности — это часть сообщения. Он не показывает поток сообщений от одного действия к другому. Диаграмма деятельности иногда рассматривается как блок-схема. Хотя диаграммы выглядят как блок-схема, это не так. Он показывает разные потоки, такие как параллельный, разветвленный, параллельный и одиночный. Диаграмма активности подходит для моделирования потока активности системы. Приложение может иметь несколько систем. Диаграмма деятельности также охватывает эти системы и описывает поток от одной системы к другой. Это конкретное использование не доступно на других диаграммах. Этими системами могут быть базы данных, внешние очереди или любая другая система. На рисунках 2.3-2.5 представлены диаграммы деятельности, на которых представлен процесс просмотра расписания студентом и заполнение расписания администратором базы данных. Как можно видеть на картинке, если пользователь выполняет какое-то действие, то система незамедлительно реагирует на это.



Рисунок 2.3 - Диаграмма деятельности процесса «Просмотр расписания»

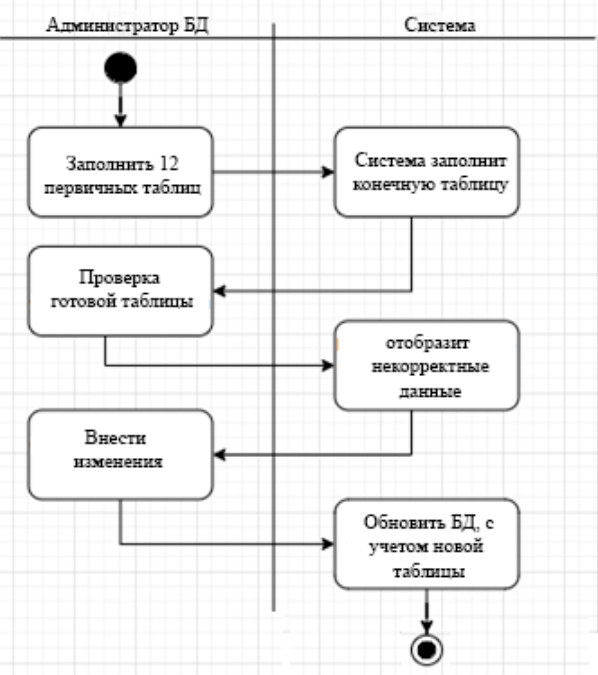


Рисунок 2.4 - Диаграмма деятельности процесса «Ввод расписания»



Рисунок 2.5 - Диаграмма деятельности процесса «Редактирование»

В текущем подразделе были описаны бизнес-процессы, которые будут использоваться в прототипе мобильного приложения.

## 2.2 Общая архитектура системы

Система должна работать с помощью API, этот термин расшифровывается как Application Programming Interface, что в переводе на русский значит «Программный Интерфейс Приложения». Простыми словами, интерфейс – это «прослойка» между приложением А и приложением Б. В ней происходят процессы, которые позволяют двум программам обмениваться информацией и выполнять функции, связанные с обеими сторонами, скрывая «внутреннее строение» программ.

Такой подход позволяет наладить взаимодействие между несколькими утилитами, не задумываясь о том, как они устроены, какая программная логика ими движет и каким образом обрабатываются передаваемые данные [4]. Такая система будет иметь вид клиент-сервер.

Используется программный архитектурный стиль REST API, который означает «передача репрезентативного состояния», то есть сервер передает клиенту представление о состоянии запрошенного ресурса после вызова API.

То есть REST API взаимодействует с помощью HTTP запросов с методами GET,POST,PUT и DELETE, для управления данными. C помощью REST API мы отправляем запрос от одного программного обеспечения, а затем второе программное обеспечение повторно отправляет ресурсы как JSON, XML или HTML.

Графическое представление как оно работает представлено на приведенном ниже рисунке 2.6.

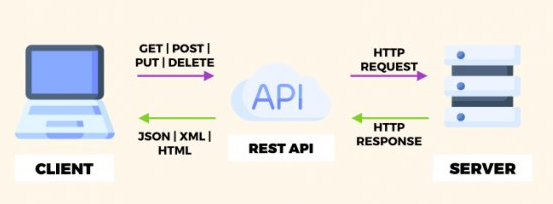


Рисунок 2.6 - Графическое представление REST API

В этой информационной системе сервер с мобильным приложением общаются с помощью XML данных. XML хранит данные в текстовом формате. Это обеспечивает независимый от программного и аппаратного обеспечения способ хранения, транспортировки и обмена данными. XML также облегчает расширение или обновление до новых операционных систем, новых приложений или новых браузеров без потери данных. Для того чтобы мобильное приложение функционировало ему нужно подключиться с указанием IP адреса сервера или к его доменному имени, но для конечного пользователя это должно быть вне поля его зрения.

Архитектура данного прототипа мобильного приложения реализована с помощью локальной базы данных и интерфейса приложения, из чего и состоит мобильное приложение, которое функционирует на смартфоне. как показано на рисунке 2.7.

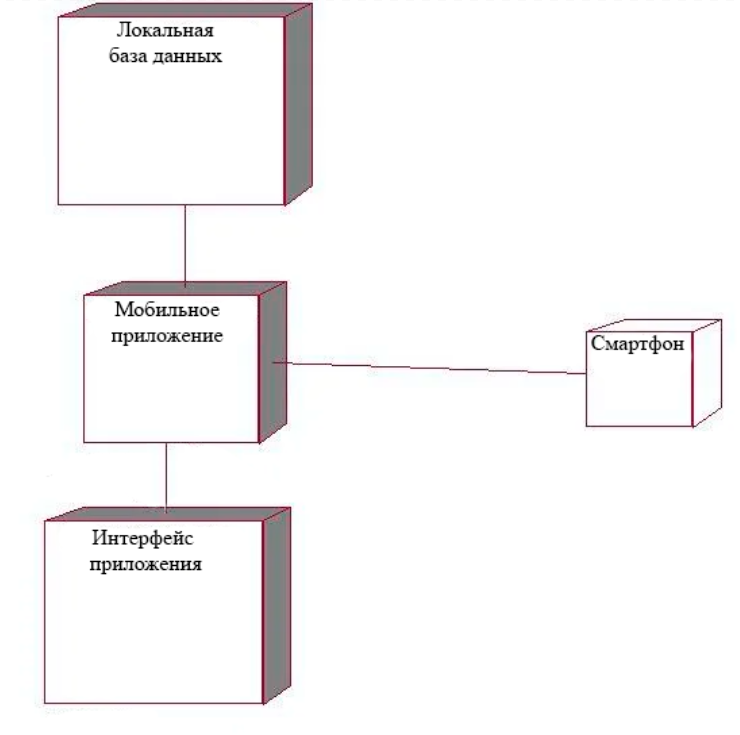


Рисунок 2.7 - Архитектура мобильного приложения

В прототипе нашего мобильного приложения серверная часть реализована не будет по некоторым причинам. Были рассмотрены несколько вариантов бесплатных серверов для хостинга, такие как Hostinger и ONOS (1&1). Это одни из самых популярных хостинг-провайдеров, но из-за ситуации в мире они прекратили предоставлять услуги клиентам из России. Были рассмотрены Российские хостинг-провайдеры, но за их услуги нужно платить немалую сумму, так как для обслуживания серверов необходимы различные оборудования, аренды стоек для размещения оборудования в дата-центрах и прочие затраты. Конечно, у отечественных провайдеров есть бесплатные тарифы, но при поиске бесплатного хостинга важно понимать, что у таких тарифов нет стабильности в работе и есть определенные ограничения [5].

**2.3 Обоснование выбора среды разработки и языка программирования**

Для создания информационной системы были выбраны две среды программирования: Visual Studio Xamarin. Для стабильного функционирования программы необходим компьютер, с объёмом оперативной памяти не менее 4 Гб, а также процессор с частотой не менее 600 мГц. Для того чтобы решить поставленную задачу необходимо использовать функциональную, эффективную и удобную платформу для разработки. В качестве такой платформы была выбрана среда Visual Studio.

Xamarin - это платформа с исходным кодом, который является открытым, прежде всего она предназначена для создания приложений отвечающих современным стандартам производительности, который был задан на рынке для iOS, Android и Windows с .NET. Платформа Xamarin представляет из себя высокий уровень абстракции, который обеспечивает управление взаимодействием между кодом базовой платформы и между общим кодом. Xamarin выполняется в управляемой среде, которая реализует такие возможности, как выделение памяти и сборка мусора.

Благодаря xamarin в среднем 90 % кода приложения может использоваться без изменений на разных платформах. С помощью этого шаблона разработчик может написать всю бизнес-логику на одном языке (или использовать существующий код приложения), но при этом получить характеристики производительности, оформление и поведение, характерные для каждой соответствующей платформы [6].

Приложения xamarin можно писать на ПК или Mac и компилировать в собственные пакеты приложений, например в файлы с расширением .apk для Android или .ipa для iOS.

Платформа Xamarin ориентирована на разработчиков, перед которыми стоят следующие задачи:

- совместное использование кода, тестов и бизнес-логики на различных платформах;

- описание кроссплатформенных приложений на языке C# в Visual Studio.

С помощью средств оценки производительности определяется состояние файла с пакетом прикладных программ. С помощью инструмента для визуализации памяти разработчик узнает, когда его приложение будет использовать слишком много оперативной памяти и когда произойдет «сборка мусора». Инструменты для анализа батареи показывают, какая нагрузка приходится на устройство.

C# - современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# - объектно-ориентированный, ориентированный на компоненты язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# это объектно-ориентированный язык [7].

*Преимущества и недостатки языка C#.*

Данный язык является объектно-ориентированным, соответственно подход к программированию в нем такой-же. Для программиста это означает, что абстрактные конструкции будут описываться на основе предметной области, а после этого необходимо между ними реализовать взаимодействие. Такой подход пользуется огромной популярностью, так как можно не держать всю информацию в голове, а легок работать по принципу, который в точности напоминает черный ящик.

У «шарпа» выделяют много преимуществ:

- поддержка подавляющего большинства продуктов Microsoft;

- бесплатность ряда инструментов для небольших компаний и некоторых индивидуальных разработчиков — Visual Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и другие;

- для строчного типа данных можем задать фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как вы всегда знаете точно, с чем вы имеете дело;

- автоматическая «сборка мусора» это значит, что нам в большинстве случаев не придётся заботиться об освобождении памяти. Языковая среда CLR сама вызовет сборщик мусора и очистит память;

- большое количество синтактических конструкций, которые никак не влияют на работу программы, но делают использование языка удобнее для человека — специальных конструкций, разработанных для понимания и написания кода. Они не имеют значения при компиляции;

- низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков;

- с помощью Xamarin на C# можно писать программы и приложения для таких операционных систем, как iOS, Android, MacOS и Linux;

- сегодня в любом регионе России имеется немало вакантных мест на должность C#-программиста.

Но есть у C# и некоторые недостатки:

- приоритетная ориентированность на платформу Windows;

- язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании покупка лицензионной версии этого языка обойдётся в круглую сумму.

В конечно итоге была сформирована таблица сравнения наиболее важных особенностей двух языков, а также оценены их минусы и плюсы относительно друг друга.

Мы сравниваем тут такие характеристики, как производительность, лаконичность языка, количество библиотек, обратная совместимость, многопоточность, простота в использовании, быстрое нахождение и исправление ошибок.

Сравнительная таблица языка C# и Java представлена в таблице 2.1.

Таблица 2.1 - Сравнение C# и Java

|  |  |  |
| --- | --- | --- |
| Сравнительные характеристики | C# | Java |
| Производительность | Средняя | Высокая |
| Лаконичность | Высокая | Средняя |
| Количество библиотек | Высокая | Очень высокая |
| Обратная совместимость | Высокая | Высокая |
| Многопоточность | Есть | Есть |
| Простота в использовании | Высокая | Высокая |
| Быстрое нахождение и исправление ошибок | Высокая | Средняя |

## 2.4 Проектирование мобильного приложения

Разработка велась в специальной среде для разработки мобильных приложений Visual Studio Xamarin. Пример как выглядит окно данной среды, представлено на рисунке 2.8. Одной из основных функций мобильного приложения является оставление заметок на предмете.

Интегрированная среда разработки Visual Studio - это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений.

Помимо стандартного редактора и отладчика, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса написания программного кода и совместной разработки одной программы.

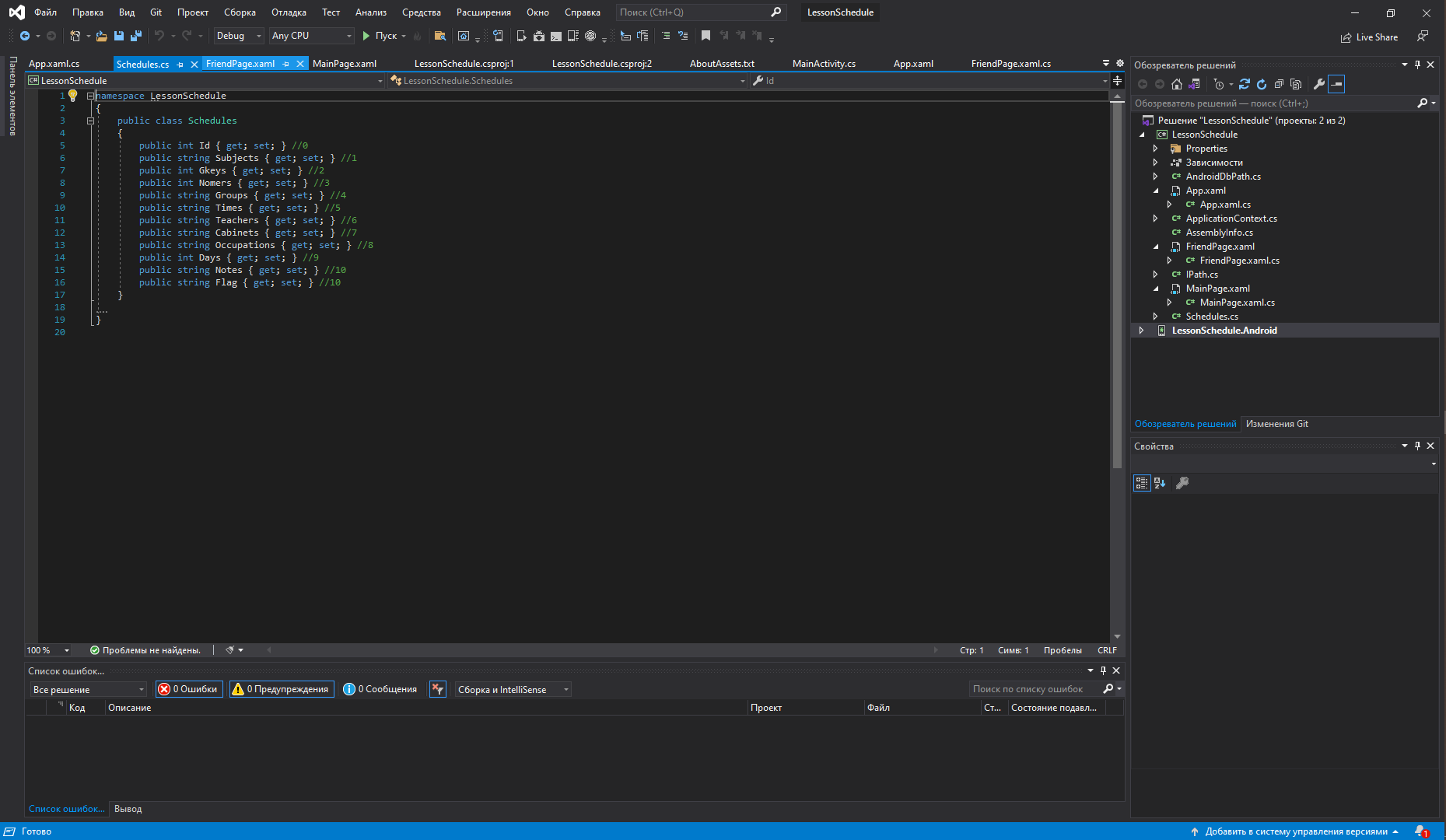


Рисунок 2.8 - Окно среды разработки

MVVM (Model-View-ViewModel) – паттерн, который позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения. Этот паттерн был представлен Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF. И хотя сейчас данный паттерн вышел за пределы WPF и применяется в самых различных технологиях, в том числе при разработке под Android, iOS, тем не менее WPF является довольно показательной технологией, которая раскрывает возможности данного паттерна [8].

На рисунке 2.9 представлено, что MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

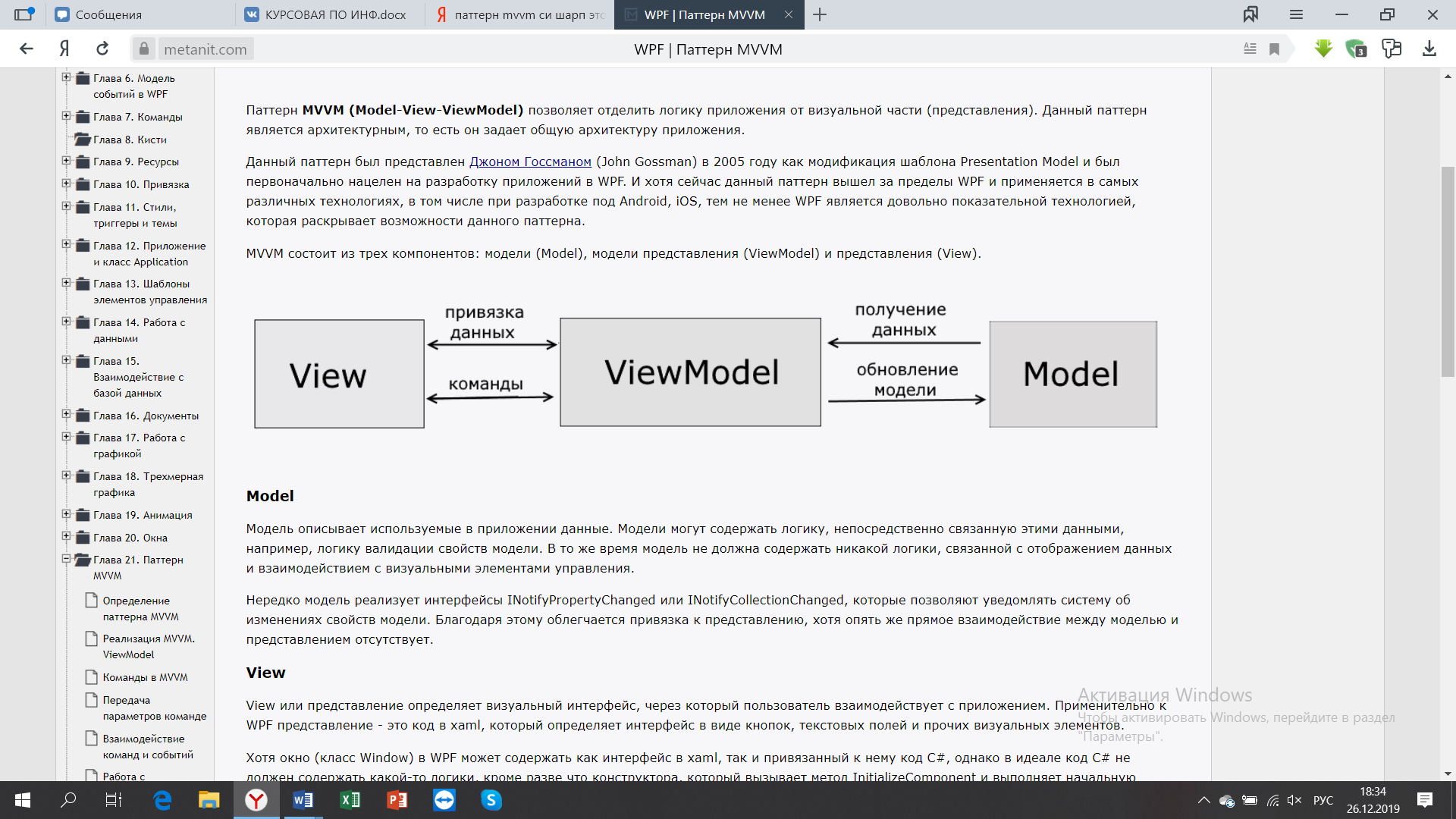


Рисунок 2.9 - Основные компоненты Model - View - ViewModel

*View и ViewModel отвечают за отображение данных.*

Model содержит бизнес-логику приложения и связанные с ней данные. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления [9].

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также ViewModel определяет логику по обновлению данных в модели. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд. Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента.

ORM (Object-Relational Mapping) - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии в настоящее время [10].

Задача ORM: обеспечить работу с данными в терминах классов, а не таблиц данных, и, напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД. Необходимо также обеспечить интерфейс для CRUD-операций над данными. В общем, необходимо избавиться от необходимости писать SQL-код для взаимодействия в СУБД.

Одной из реализаций ORM является Entity Framework, которая также использована в работе.

ADO.NET Entity Framework (EF) - объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM [11].

Изначально с самой первой версии Entity Framework поддерживал подход Database First, который позволял по готовой базе данных сгенерировать модель edmx. Затем эта модель использовалась для подключения к базе данных. Позже был добавлен подход Model First. Он позволял создать вручную с помощью визуального редактора модель edmx, и по ней создать базу данных. Начиная с 5.0 предпочтительным подходом становится Code First. Его суть - сначала пишется код модели на C#, а затем по нему генерируется база данных. При этом модель edmx уже не используется. В работе используется подход Code First.

В программе создан класс «Schedules», который хранит данные об ID предмета, группы и их ключи, кабинет, время, ФИО преподавателя, номер пары, день недели, заметки, и информацию о том оставлена ли заметка в данный день. Он представлен на рисунке 2.10.

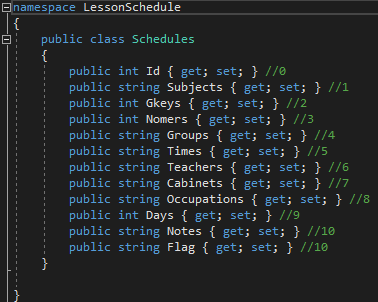


Рисунок 2.10 - Класс «Schedules»

Далее прописали XAML код, создали строки для ввода данных в БД, затем создали кнопки добавления, и удаления записей БД. Код представлен на рисунке 2.11.

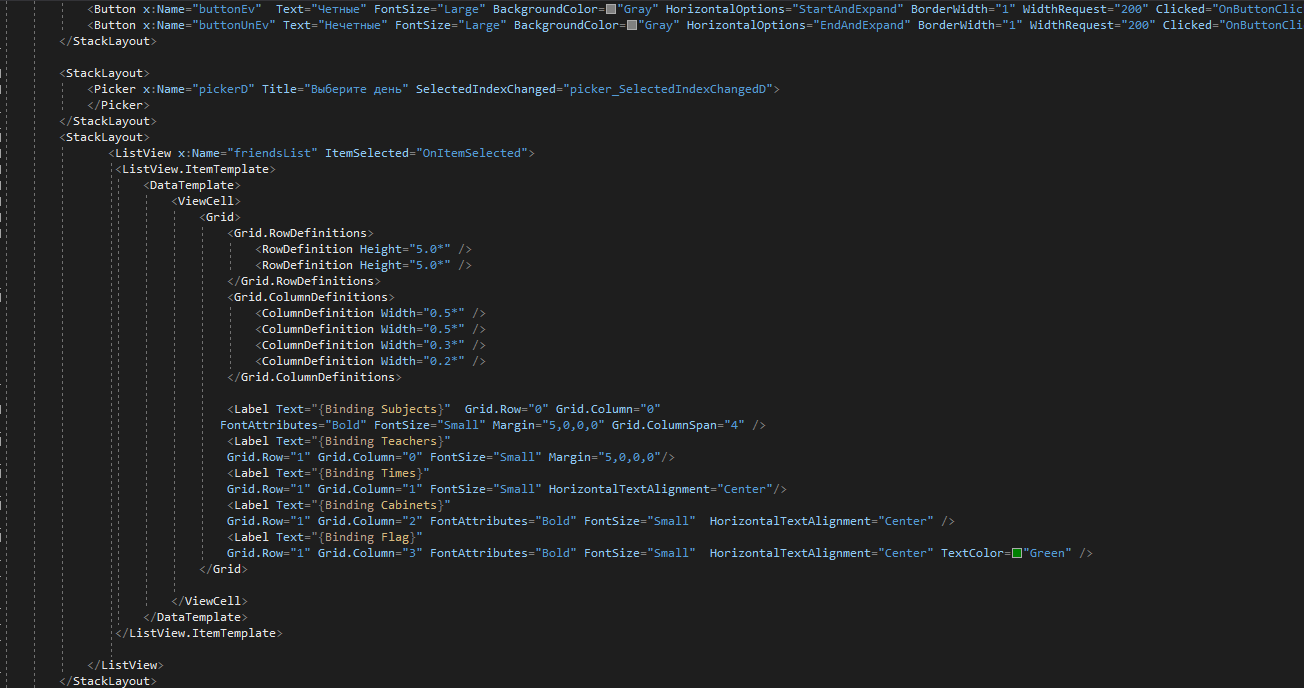


Рисунок 2.11 - Код в MainPage.xaml

Затем прописали в классе MainPage.xaml.cs обработчик нажатия кнопки добавления записи, удаления, изменения записей в бд и метод отображения записей в CollectionView.

Код класса MainPage.xaml.cs представлен на рисунках 2.12-2.13.

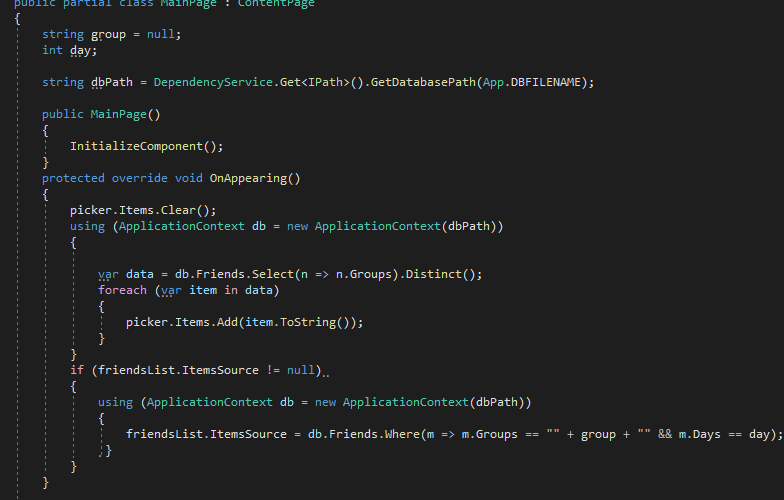


Рисунок 2.12 - Класс MainPage.xaml.cs



Рисунок 2.13 - Класс MainPage.xaml.cs обработчик нажатия кнопки

## 2.5 Описание интерфейса информационной системы

Для связи базы данных с приложением был создан определенный набор представлений и процедур, которые позволяют реализовать дружелюбный интерфейс.

Разработка прототипа интерфейса мобильного приложения проводилась в Visual Studio Xamarin, в разработке использовались различные элементы управления.

Стартовое окно приложения, в котором есть возможность выбрать группу, неделю и день.

WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен System.Windows. Используемые управляемые технологии, такие как ASP.NET и Windows Forms, основные принципы программирования с помощью WPF многим знакомы: создается экземпляры классов, задаются свойства, вызываются методы и обрабатываются события.

WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий: свойства зависимостей и перенаправленные события [12].

WPF позволяет разрабатывать приложения, используя как *разметку*, так и код программной части. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) - для реализации его поведения. Такое разделение внешнего вида и поведения имеет ряд преимуществ:

- затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение;

- повышается эффективность разработки, так как дизайнеры, занимающиеся внешним видом приложения, могут работать параллельно с разработчиками, реализующими поведение приложения;

- глобализация и локализация приложений WPF упрощена.

XAML - это язык разметки на основе XML, который служит для определения внешнего вида приложения в декларативной форме. Обычно он используется для создания окон, страниц и пользовательских элементов управления, а также их заполнения элементами управления, фигурами и графическими элементами [13].

Так как язык XAML основан на XML, создаваемый с его помощью пользовательский интерфейс образует иерархию вложенных элементов, известную как дерево элементов. Дерево элементов обеспечивает логичный и интуитивно понятный способ создания пользовательских интерфейсов и управления ими [14].

Чтобы как-то взаимодействовать с пользователем, получать от пользователя ввод с клавиатуры или мыши и использовать введенные данные в программе, необходимы элементы управления. WPF предлагает богатый стандартный набор элементов для этой задачи [15].

Все изначально доступные элементы управления могут быть условно разделены на несколько подгрупп:

- элементы управления содержимым, например кнопки (Button), метки (Label);

- специальные контейнеры, которые содержат другие элементы, но в отличие от элементов Grid или Canvas не являются контейнерами компоновки - ScrollViewer,GroupBox;

- декораторы, чье предназначение создание определенного фона вокруг вложенных элементов, например, Border или Viewbox;

- элементы управления списками, например, ListBox, ComboBox;

- текстовые элементы управления, например, TextBox, RichTextBox;

- элементы, основанные на диапазонах значений, например, ProgressBar, Slider;

- элементы для работ с датами, например, DatePicker и Calendar;

- остальные элементы управления, которые не вошли в предыдущие подгруппы, например, Image.

Все элементы управления наследуются от общего класса System.Window.Controls.Control и имеют ряд общих свойств. А общая иерархия элементов управления представлена на рисунке 2.14.

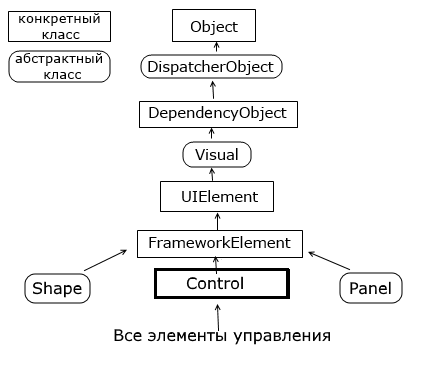


Рисунок 2.14 - Общая иерархия элементов управления

Создали новый WPF проект, добавили элементы управления из панели инструментов на MainWindow, такие как: Rectangle, Button, TextBlock, и посмотреть их описание в XAML, как показано на рисунке 2.15.

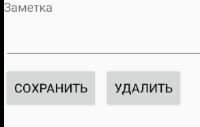
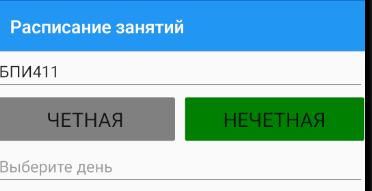


Рисунок 2.15 - Элементы управления

Изменить свойства элементов (со свойств по умолчанию на свои), а также начальный экран приложения представлены на рисунке 2.16-2.17.

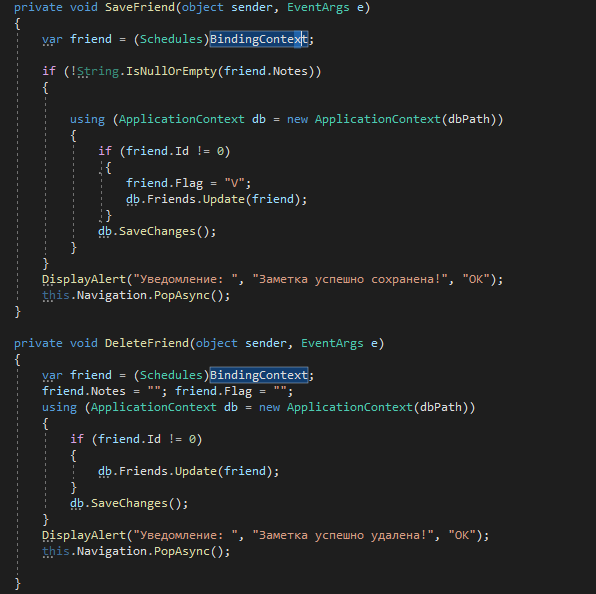


Рисунок 2.16 - добавление действия при нажатии на элементы

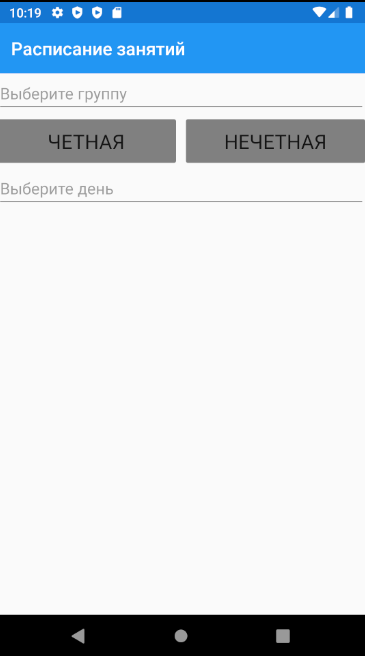


Рисунок 2.17 - Запуск приложения

Далее добавили обработчик событий Click в элемент Button. Добавить в обработчик событий Click код, который добавляет в элементы TextBlock уже имеющийся текст, как показано на рисунке 2.18. Также в разметку XAML добавили код, который позволяет при наведении на кнопку менять ее цвет, фрагмент кода представлен на рисунке 2.19, а работа приложения представлена на рисунках 2.20-2.21. Полный код программы представлен в приложении А.

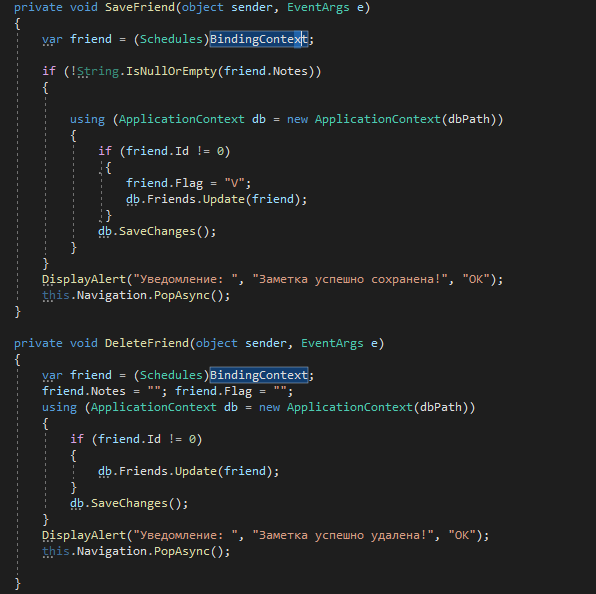


Рисунок 2.18 - Код для обработчика событий Click

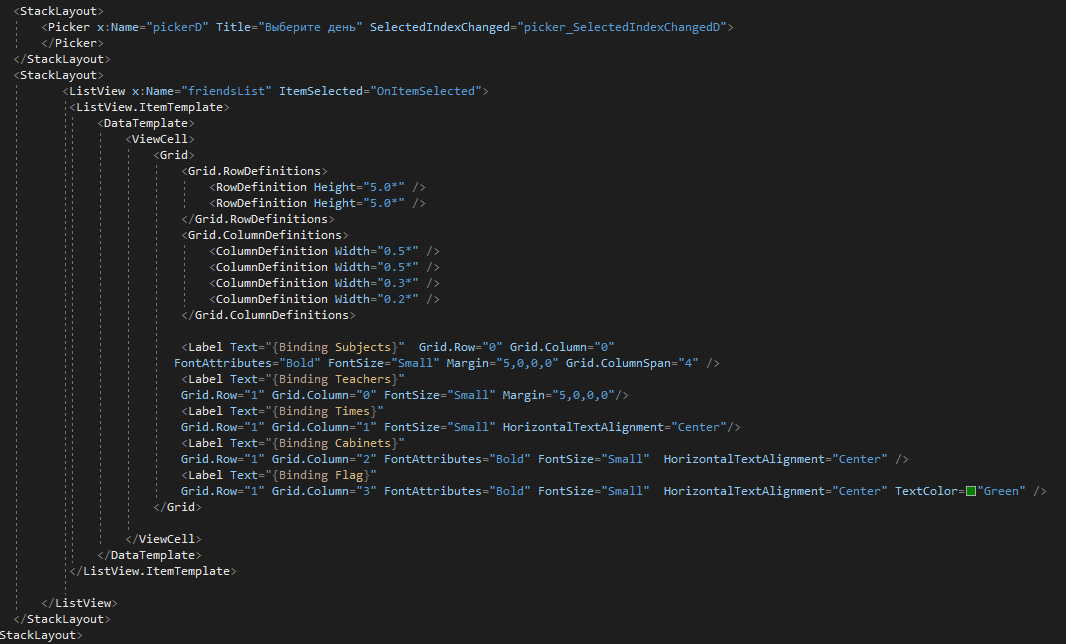


Рисунок 2.19 - Код в разметке XAML для изменения цвета кнопки

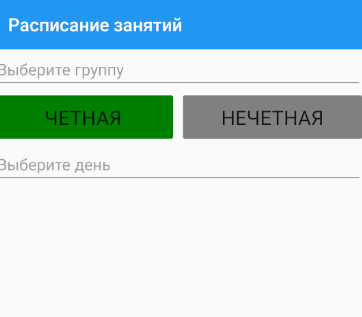
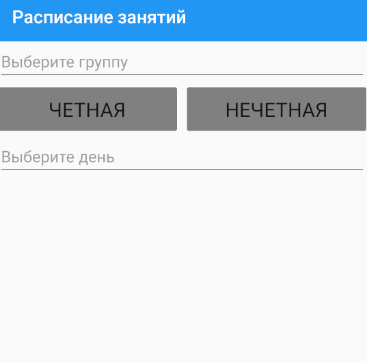


Рисунок 2.20 - Изменение цвета кнопки

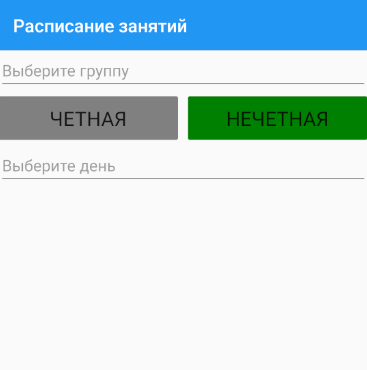


Рисунок 2.21 - Нажатие на кнопку «нечетная»

Затем подготовили наброски ключевых состояний разрабатываемой ИС, в данном случае - «Расписание предметов для мобильных устройств факультета ФБИ». Данное приложение будет состоять из трех основных экранов - экран четной недели, экран нечетной недели, экран заметки на предмете. Данные экраны представлены на рисунках 2.22 - 2.23.

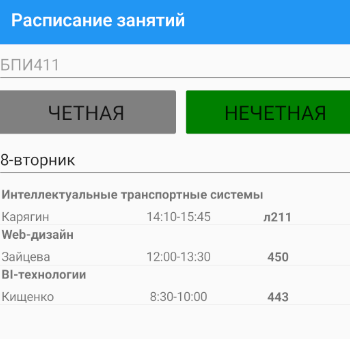


Рисунок 2.22 - Экран недели

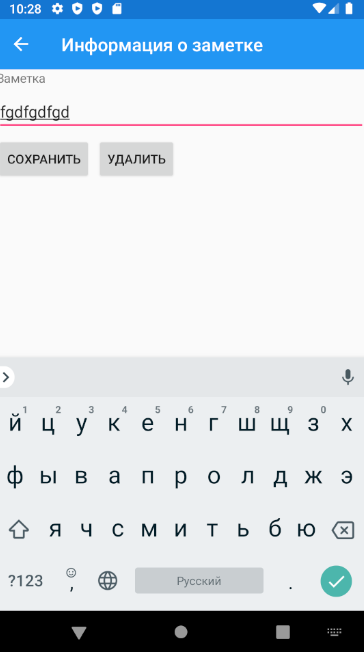


Рисунок 2.23 - Экран заметок

Далее опишем очередность и связи между экранами.

При входе в приложение открывается главный экран. Далее, нужно выбрать неделю, выбрать группу и день недели. Следующий экран – экран заметок. Отсюда мы можем перейти к нужному нам экрану, нажав на соответствующие пункты меню в нижней части экрана.

# **2.6** **Выводы к разделу**

В текущем разделе были подробно описаны обоснования выбора среды разработки, языка программирования и СУБД, а как же проектирование ИС.

**3** **Руководство пользователя**

Готовый продукт представляет собой мобильное приложение. Этот продукт позволит оптимизировать тайм-менеджмент студентов и преподавателей при помощи непрерывной связи с пользователем и возможностью постоянно просмотреть расписание.

Мобильное приложение должно автоматизировать следующие бизнес‑процессы организации:

- информирование пользователя о изменении расписания;

- предоставление информации о появлении новой заметки.

Для просмотра расписания и заметок требуется установить разработанное нами приложение. Установка осуществляется путем скачивания APK файла.

Для работы с приложением можно использовать интерфейс. Процесс взаимодействия представлен на рисунке 3.1.

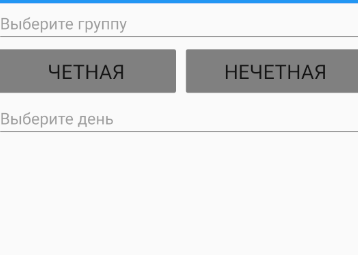


Рисунок 3.1 - Взаимодействие с приложением

Приложение имеет главное меню, которое включает в себя следующие основные пункты:

- смена недели;

- выбор дня;

- оставление заметки.

Во вкладке «Выбор группы» отображаются Группы:

- БПИ411;

- БИСТ411;

- МЛ411 и так далее.

Пункты меню представлены на рисунке 3.2.

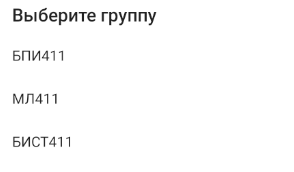


Рисунок 3.2 - Пункты меню приложения

При переходе по пункту меню «Выбор дня недели» пользователь может узнать информацию об днях недели. Результат представлен на рисунке 3.3.



Рисунок 3.3 - Выбор пункта меню «Выбор дня недели»

Возможность переключения недели (четная, нечетная) через пункт меню. При выборе недели, она выделяется зеленым цветом, что помогает пользователю понимать какая неделя выбрана сейчас. Результат представлен на рисунках 3.4.

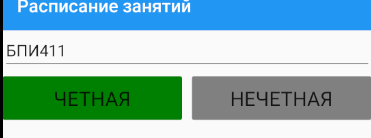


Рисунок 3.4 - Выбор пункта меню «Факультеты и специальности»

При заполнении всех пунктов меню выводится расписание, представленные на рисунке 3.5.

При нажатии на предмет открывается окно, которое позволяет оставить заметку. Результат представлен на рисунке 3.6.

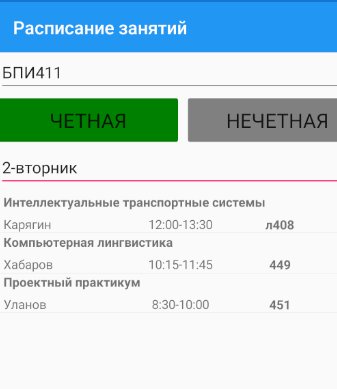


Рисунок 3.5 - Вывод расписания после заполнения всех пунктов меню

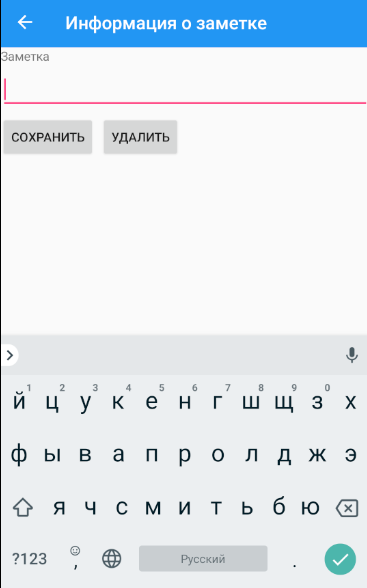


Рисунок 3.6 - Новое окно с возможностью оставить заметку

При возврате назад на против предмета будет стоять галочка, если заметка есть. Результат представлен на рисунке 3.7.

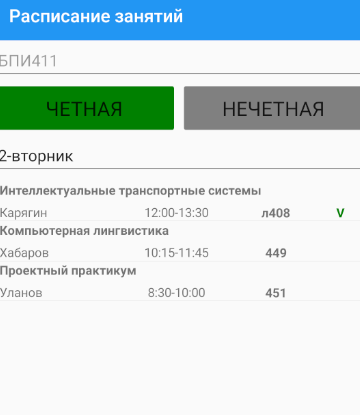


Рисунок 3.7 - Галочка на против предмета с оставленной заметкой

При отсутствии ответа от БД необходимо заново открыть приложение. Данная операция прервет связь с БД. Далее вы сможете вновь взаимодействовать с БД, через интерфейс приложения.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы «Разработка мобильного приложения «расписанием для факультета» был описан процесс реализации системы от анализа предметной области до руководства пользователя конечным продуктом.

В процессе работы было сделано:

- проведено сравнение программных аналогов;

- проведен анализ деятельности компании;

- исследованы процессы в компании (как есть);

- смоделированы бизнес-процессы;

- разработана информационная система;

- описано руководство пользователя.

Все поставленные задачи были выполнены в полном объеме, а цель работы была достигнута.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Моделирование бизнес-процессов в business studio // businessarchitecture [Электронный ресурс]. – URL: https://businessarchitecture.ru/modeling-business-studio (Дата обращения 13.05.2022).
2. Бизнес-процессы // KPMS [Электронный ресурс]. – URL: https://www.kpms.ru/Automatization/BPM.htm (Дата обращения 13.05.2022).
3. UML- диаграммы// Сoderlessons [Электронный ресурс]. – URL: https://coderlessons.com/tutorials/akademicheskii/uchit-uml/uml-diagrammy-deiatelnosti (Дата обращения 16.05.2022).
4. API // Timeweb [Электронный ресурс]. – URL https://timeweb.com/ru/community/articles/chto-takoe-api (Дата обращения 17.05.2022).
5. Хостинг // Habr [Электронный ресурс]. – URL https://habr.com/ru/company/timeweb/blog/655547/ (Дата обращения 17.05.2022).
6. Что такое Xamarin // microsoft [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/xamarin/get-started/what-is-xamarin (Дата обращения 18.05.2022).
7. Обзор языка C# // microsoft [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp (Дата обращения 18.05.2022).
8. Паттерн MVVM // metanit.com [Электронный ресурс]. – URL: https://metanit.com/sharp/wpf/22.1.php (Дата обращения 12.05.2022).
9. MVVM // medium [Электронный ресурс]. – URL: https://medium.com/clean-code-channel/mvvm-d01c432217af (Дата обращения 12.05.2022).
10. ORM // wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/ORM (Дата обращения 13.05.2022).
11. ADO.NET Entity Framework // wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/ADO.NET\_Entity\_Framework (Дата обращения 15.05.2022).
12. WPF что это такое // kupibaby [Электронный ресурс]. – URL: https://kupibaby.ru/znachenie/windows-presentation-foundation-chto-etotakoe.html (Дата обращения 12.05.2022).
13. Руководство по классическим приложениям (WPF .NET) // microsoft [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0 (Дата обращения 15.05.2022).
14. Язык программирования C# // geekbrains [Электронный ресурс]. – URL: https://gb.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke (Дата обращения 16.05.2022).
15. Элементы управления C# // metanit [Электронный ресурс]. – URL: https://metanit.com/sharp/wpf/5.1.php (Дата обращения 16.05.2022).

##### ПРИЛОЖЕНИЕ A

**Код программы**

namespace LessonSchedule

{

public class Schedules

{

public int Id { get; set; } //0

public string Subjects { get; set; } //1

public int Gkeys { get; set; } //2

public int Nomers { get; set; } //3

public string Groups { get; set; } //4

public string Times { get; set; } //5

public string Teachers { get; set; } //6

public string Cabinets { get; set; } //7

public string Occupations { get; set; } //8

public int Days { get; set; } //9

public string Notes { get; set; } //10

public string Flag { get; set; } //10

}

public partial class MainPage : ContentPage

{

string group = null;

int day;

string dbPath = DependencyService.Get<IPath>().GetDatabasePath(App.DBFILENAME);

public MainPage()

{

InitializeComponent();

}

protected override void OnAppearing()

{

picker.Items.Clear();

using (ApplicationContext db = new ApplicationContext(dbPath))

{

var data = db.Friends.Select(n => n.Groups).Distinct();

foreach (var item in data)

{

picker.Items.Add(item.ToString());

}

}

if (friendsList.ItemsSource != null)

{

using (ApplicationContext db = new ApplicationContext(dbPath))

{

friendsList.ItemsSource = db.Friends.Where(m => m.Groups == "" + group + "" && m.Days == day);

}

}

}

private void picker\_SelectedIndexChanged(object sender, EventArgs e)

{

if (picker.SelectedIndex != -1)

group = picker.Items[picker.SelectedIndex].ToString();

else

picker.Title = group;

}

private void picker\_SelectedIndexChangedD(object sender, EventArgs e)

{

if (pickerD.SelectedIndex != -1)

{

try

{

string days = pickerD.Items[pickerD.SelectedIndex].ToString();

string[] subs = days.Split('-');

day = Convert.ToInt32(subs[0]);

}

catch (Exception ex) { DisplayAlert("Ошибка: ", ex.Message.ToString(), "OK"); }

finally

{

using (ApplicationContext db = new ApplicationContext(dbPath))

{

friendsList.ItemsSource = db.Friends.Where(m => m.Groups == "" + group + "" && m.Days == day);

}

}

}

}

private void OnButtonClickedEven(object sender, System.EventArgs e)

{

buttonUnEv.BackgroundColor = Color.Gray;

buttonEv = (Button)sender;

buttonEv.BackgroundColor = Color.Green;

pickerD.Items.Clear(); pickerD.SelectedIndex = -1;

using (ApplicationContext db = new ApplicationContext(dbPath))

{

var data = db.Friends.OrderBy(p => p.Days).Where(m => m.Groups == "" + group + "" && m.Days >= 1 && m.Days <= 6).Select(n => n.Days).Distinct();

foreach (var item in data)

{

string days = StringDays(item);

pickerD.Items.Add(item.ToString()+"-"+days);

}

}

}

private string StringDays(int day)

{

string days = "";

switch (day)

{

case 1:

case 7:

days = "понедельник";

break;

case 2:

case 8:

days = "вторник";

break;

case 3:

case 9:

days = "среда";

break;

case 4:

case 10:

days = "четверг";

break;

case 5:

case 11:

days = "пятница";

break;

case 6:

case 12:

days = "суббота";

break;

}

return days;

}

private void OnButtonClickedUnEven(object sender, System.EventArgs e)

{

buttonEv.BackgroundColor = Color.Gray;

buttonUnEv = (Button)sender;

buttonUnEv.BackgroundColor = Color.Green;

pickerD.Items.Clear(); pickerD.SelectedIndex = -1;

using (ApplicationContext db = new ApplicationContext(dbPath))

{

var data = db.Friends.OrderBy(p=>p.Days).Where(m => m.Groups == "" + group + "" && m.Days >= 7 && m.Days <= 12).Select(n => n.Days).Distinct();

foreach (var item in data)

{

string days = StringDays(item);

pickerD.Items.Add(item.ToString() + "-" + days);

}

}

}

// обработка нажатия элемента в списке

private async void OnItemSelected(object sender, SelectedItemChangedEventArgs e)

{

Schedules selectedFriend = (Schedules)e.SelectedItem;

FriendPage friendPage = new FriendPage();

friendPage.BindingContext = selectedFriend;

await Navigation.PushAsync(friendPage);

}

}

public interface IPath

{

string GetDatabasePath(string filename);

}

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class FriendPage : ContentPage

{

string dbPath;

public FriendPage()

{

InitializeComponent();

dbPath = DependencyService.Get<IPath>().GetDatabasePath(App.DBFILENAME);

}

private void SaveFriend(object sender, EventArgs e)

{

var friend = (Schedules)BindingContext;

if (!String.IsNullOrEmpty(friend.Notes))

{

using (ApplicationContext db = new ApplicationContext(dbPath))

{

if (friend.Id != 0)

{

friend.Flag = "V";

db.Friends.Update(friend);

}

db.SaveChanges();

}

}

DisplayAlert("Уведомление: ", "Заметка успешно сохранена!", "OK");

this.Navigation.PopAsync();

}

private void DeleteFriend(object sender, EventArgs e)

{

var friend = (Schedules)BindingContext;

friend.Notes = ""; friend.Flag = "";

using (ApplicationContext db = new ApplicationContext(dbPath))

{

if (friend.Id != 0)

{

db.Friends.Update(friend);

}

db.SaveChanges();

}

DisplayAlert("Уведомление: ", "Заметка успешно удалена!", "OK");

this.Navigation.PopAsync();

}

}

public class ApplicationContext : DbContext

{

private string \_databasePath;

public DbSet<Schedules> Friends { get; set; }

public ApplicationContext(string databasePath)

{

\_databasePath = databasePath;

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite($"Filename={\_databasePath}");

}

}

public partial class App : Application

{

public const string DBFILENAME = "Schedulesapp.db";

public App()

{

InitializeComponent();

string dbPath = DependencyService.Get<IPath>().GetDatabasePath(DBFILENAME);

using (var db = new ApplicationContext(dbPath))

{

db.Database.EnsureCreated();

}}}