АКАДЕМИЯ МАРКЕТИНГА И СОЦИАЛЬНО-ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ – ИМСИТ (г. Краснодар)

Факультет информатики и вычислительной техники

Кафедра математики и вычислительной техники

Допустить к защите

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_ Н.С. Нестерова

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Разработка системы электронного расписания для учебно-методического управления образовательных организаций» (по материалам ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса» (г. Краснодар), г. Краснодар)

Направление подготовки 230100.62 «Информатика и

вычислительная техника»

Работу выполнил студент

5 г. заочной формы обучения,

группы 14-ЗИВТ-СПО-01

Скляр Дмитрий Эдуардович

Научный руководитель, к.т.н., доцент Н. С. Нестерова

Краснодар

2018

АКАДЕМИЯ МАРКЕТИНГА И СОЦИАЛЬНО-ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ — ИМСИТ (г. Краснодар)

Факультет информатики и вычислительной техники

Кафедра математики и вычислительной техники

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_г.

**ЗАДАНИЕ**

на выпускную квалификационную работу

Студенту Скляр Дмитрию Эдуардовичу

Тема выпускной квалификационной работы:

«Разработка системы электронного расписания для учебно-методического управления образовательных организаций» (по материалам ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса» (г. Краснодар), г. Краснодар)

Утверждена приказом ректора от « 22 » июля 2014 г. №1001

Целевая установка: улучшение качества образовательного процесса, повышение интереса студентов к изучаемой дисциплине.

Основные вопросы, подлежащие разработке: анализ предметной области, создание интерактивного и мультимедийного контента, разработка программного кода.

Основная литература:

1. Тжаскалик Тадеуш - Введение в исследование операций с применением компьютера. - М.: Горячая Линия-Телеком, 2010. — 310 с.

2. В. В. Фаронов - Delphi. Программирование на языке высокого уровня. Учебник для вузов. – Спб.: Питер, 2012. — 348с.

Срок представления законченной работы « 20 » декабря 2014 г.

Дата выдачи задания: «2» сентября 2014г.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Нестерова Н.С. /

(подпись) (Ф.И.О.)

Задание получил: «2» сентября 2014г.

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Скляр Д.Э. /

(подпись) (Ф.И.О.)

РЕФЕРАТ

Выпускная квалификационная работа 0с., 0рис., 4 табл., 25 источников, 1 прил.

ЧУ ПОО КТУИС, РАСПИСАНИЕ, БАЗА ДАННЫХ, СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ, C#, ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ, АВТОМАТИЗИРОВАННАЯ СИСТЕМА.

Объектом исследования является подразделение учебно-методического управления образовательной организации среднего профессионального образования.

Цель работы состоит в автоматизации процесса составления расписания занятий.

К полученным результатам относится разработанная система электронного расписания. Она позволяет автоматизировать работу подразделения учебно-методического управления образовательной организации, предоставив ему автоматически составленное расписание занятий образовательной организации, с последующим, при необходимости, его модифицированием, для достижения исключительно необходимого результата, вывод необходимых отчётов.

Новизна результатов заключается в разработке эффективной системы создания электронного расписания для организации среднего профессионального образования в условиях федеральных государственных образовательных стандартов.

Внедрение результатов предусмотрено в ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса».

Эффективность работы характеризуется автоматизацией работы, скоростью получения необходимого результата, упрощением, за счёт исключения второстепенных данных.

Определения, обозначения и сокращения

ОО – образовательная организация

БД – база данных

СУБД – система управления базами данных

ЧУ ПОО «КТУИС» – Частное учреждение – профессиональная образовательная организация «Краснодарский техникум управления, информатизации и сервиса»

UI – User Interface (интерфейс пользователя)

UX - User Experience (дословно «пользовательский опыт»)

АСУ – автоматизированная система управления

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc503992910)

[1 Исследование предметной области 2](#_Toc503992911)

[1.1 ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса» 2](#_Toc503992912)

[1.2 Обоснование актуальности составления электронного расписания занятий 2](#_Toc503992913)

[2 Обзор существующих продуктов 2](#_Toc503992914)

[2.1 «Экспресс-расписание Колледж» 2](#_Toc503992915)

[2.2 «Ректор-Колледж» 2](#_Toc503992916)

[2.3 Конфигурация «1С: Автоматизированное составление расписания. Колледж» 2](#_Toc503992917)

[3 Требования к программному продукту 2](#_Toc503992918)

[4 Техническое задание на разработку 2](#_Toc503992919)

[4.1 Введение 2](#_Toc503992920)

[4.1.1 Наименование программы 2](#_Toc503992921)

[4.1.2 Краткая характеристика области применения 2](#_Toc503992922)

[4.2 Назначения разработки 2](#_Toc503992923)

[4.2.1 Функциональное назначение 2](#_Toc503992924)

[4.2.2 Эксплуатационное назначение 2](#_Toc503992925)

[4.3 Требования к программе или программному изделию 2](#_Toc503992926)

[4.3.1 Требования к функциональным характеристикам 2](#_Toc503992927)

[4.3.2 Требования к надёжности 2](#_Toc503992928)

[4.3.3 Условия эксплуатации 2](#_Toc503992929)

[4.3.4 Требования к составу и параметрам технических средств 2](#_Toc503992930)

[4.3.5 Требования к информационной и программной совместимости 2](#_Toc503992931)

[4.4 Требования к программной документации 2](#_Toc503992932)

[4.4.1 Предварительный состав программной документации 2](#_Toc503992933)

[4.6 Порядок контроля и приемки 2](#_Toc503992934)

[5 Модель предметной области 2](#_Toc503992935)

[6 Выбор метода и инструментов разработки 2](#_Toc503992936)

[6.1 Выбор среды программирования 2](#_Toc503992937)

[6.2 Выбор языка программирования 2](#_Toc503992938)

[7 Реализация разработки 2](#_Toc503992939)

[7.1 Построение моделей данных 2](#_Toc503992940)

[7.2 Разработка интерфейса пользователя 2](#_Toc503992941)

[7.3 Испытания 2](#_Toc503992942)

[8 Разработка проектной документации 2](#_Toc503992943)

[8.1 Руководство пользователя 2](#_Toc503992944)

[9 Оценка эффективности программного продукта 2](#_Toc503992945)

[10 Мероприятия по охране труда 2](#_Toc503992946)

[10.1 Общие требования безопасности при работе за компьютером 2](#_Toc503992947)

[10.2 Безопасность перед началом работы 2](#_Toc503992948)

[10.3 Безопасность во время работы за компьютером 2](#_Toc503992949)

[10.4 Безопасность при возникновении аварийной ситуации 2](#_Toc503992950)

[10.5 Безопасность по окончании работы 2](#_Toc503992951)

[ЗАКЛЮЧЕНИЕ 2](#_Toc503992952)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 2](#_Toc503992953)

[ПРИЛОЖЕНИЯ А 2](#_Toc503992954)

ВВЕДЕНИЕ

В современном информационном обществе значительное внимание отводится информатизации в сфере образования. Информатизация образования — это процесс обеспечения системы образования теорией и практикой разработки и использования новых информационных технологий, ориентированных на реализацию целей обучения и воспитания. Процесс информатизации образования невозможен без оснащения учреждений образования и органов управления образованием аппаратными и программными средствами информационных технологий.

Расписание учебных занятий является основным календарным планом системы планирования учебно-воспитательной работы образовательной организации, от которой зависит своевременное и качественное выполнение учебных планов и образовательных программ.

В настоящее время задача правильного формирования расписания учебных занятий является значимой частью в управлении учебным процессом. Таким образом, проблема автоматизации составления расписания учебных занятий в учебных заведениях является одной из преимущественно значимых проблем в организации учебного процесса. От успешности составленного расписания зависит:

* качество знаний;
* результативность обучения;
* удобство обучения студентов и работы преподавателей.

Автоматизация процесса составления расписания занятий позволит значительно сократить временные затраты на составление расписаний и облегчит работу сотрудникам учебного отдела образовательного учреждения.

Проблему составления расписания следует воспринимать, не только как трудоёмкий процесс, объект автоматизации с использованием ЭВМ, требующий особого внимания, но и как механизм оптимального управления.

Поскольку при составлении расписания возникает большое количество факторов, практически невозможно учесть их все. Интересы участников учебного процесса достаточно многообразны, поэтому задача, которой является составление расписания занятий учебного процесса, является многокритериальной с варьирующимся множеством факторов, влияющих на его конечный результат.

Решение таких задач, как правильно, осуществляется в два этапа:

* получение оптимального варианта, с точки зрения используемых критерием;
* последующая доработка человеком, с целью учёта неформализованных факторов.

Расписание занятий составляет учебная часть под руководством заместителя директора по учебной работе. В основе составления расписания лежат учебные планы, наличие материально-технической базы, количество групп и преподавателей.

На первом этапе задачи, при составлении расписания можно выделить шесть атрибутов:

* группа, в которой проводятся занятия;
* преподаватель, который проводит занятия;
* количество часов на изучение предмета;
* предмет, по которому проводится занятие;
* аудитория;
* календарное время проведения занятия.

Данные атрибуты должны учитываться при разработке системы электронного расписания.

Целью данной работы является разработка программного продукта, позволяющего составить расписание учебных занятий, учитывая множественное количество факторов, при необходимости иметь возможность внесения изменений в конечный результат, предоставленный программным продуктом. Автоматизация рабочего места сотрудника учебно-методического управления, за счёт ускорение процесса составления расписания, предоставления необходимых для работы отчётов.

Для достижения данной цели потребуется решить ряд задач: проанализировать предметную область – процесс составления расписания занятий в образовательной организации; выработать требования к программному продукту и построить модель предметной области; на основании требований и модели, выбрать подходящее для решения задачи языки программирования и инструменты; реализовать разработку и составить необходимую документацию.

Объектом исследования работы является учебно-методическое управление ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса» в обязанности которого входит составление расписания учебных занятий.

Предмет исследования – процесс составления расписания.

Структурно работа состоит из введения, десяти глав и заключения.

В первой главе изучена система электронного расписания занятий и её влияние на учебный процесс.

Во второй главе выполнен критический обзор программных продуктов, предназначенных для решений названной задачи.

В главах третьей и четвёртой главах приведены требования к программному обеспечению и составлено техническое задание на разработку.

В пятой главе выявлены основные сущности предметной области и связи между ними, построена непротиворечивая модель предметной области.

Выбор инструментов, методологий и языков программирования для реализации различных частей разработки описан в главе шесть.

Главы седьмой и восьмой главах содержится описание реализации некоторых проектных документов: руководство пользователя, руководство по установке.

В главе девять оценено повышение производительности пользователя при использовании разработанного программного обеспечения и выполнено сравнение разработанного продукта с наиболее близким аналогом.

1 Исследование предметной области

1.1 ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса»

Частное учреждение – профессиональная образовательная организация «Краснодарский техникум управления информатизации и сервиса» [1] был учрежден в 1998 году по решению собрания учредителей Института маркетинга и социально-информационных технологий как негосударственное образовательное учреждение «Финансово-экономический колледж ИМСИТ» с целью создания системы непрерывного профессионального образования.

В 2002 году колледж был переименован в Негосударственное образовательное учреждение среднего профессионального образования «Колледж управления, информатики и сервиса Академии маркетинга и социально-информационных технологий» (сокращенное наименование «НОУ СПО КУИС Академии ИМСИТ»).

В настоящее время в связи с принятием Федерального закона от 29.12.2014 №273-ФЗ «Об образовании в Российской Федерации» колледж был переименован в Частное учреждение — профессиональная образовательная организация «Краснодарский техникум управления информатизации и сервиса».

Как и в любом учебном заведении, в условиях реализации ФГОС СПО [2] при модульной разработке учебного процесса рекомендуется использовать метод построения занятий по типу – семестровое, то есть планирования отдельно каждой недели семестра. Семестровое расписание реализует следующие задачи:

* установить оптимальную последовательность в изучении дисциплин, профессиональных модулей без учёта кратности часов количеству недель в семестре;
* рационально планировать в течение семестра проведение зачетов, экзаменом, контрольных работ, практических и лабораторных занятий, учебных и производственных практик, а также сдачу курсовых работ.

1.2 Обоснование актуальности составления электронного расписания занятий

Для большинства моделей теории расписаний, получение оптимального расписания является трудноразрешимой задачей, а решение на практике показывают ещё большую сложность, так как данные решения должны удовлетворять многочисленным, а зачастую и конфликтующим между собою, условиям, как организационного, так и психологического характера. Выходом из данного положения является отказ от подхода, когда пригодным считается идеальное решение, так как на практике добиться его к сожалению, невозможно.

Качественный и количественный рост сферы среднего профессионального образования требует нового подхода к решению задач в учебно-методической, научной и хозяйственной деятельности. Этот подход с недавних пор находит свою реализацию в применение современных средств вычислительной техники и математических методов в управлении профессиональным образованием.

В современном мире всё большее распространение получают различные системы автоматизации процессов профессиональной деятельности, которые всего выполнялись вручную. Например, системы приятия решений в маркетинге, экспертные системы, системы тестирования, прогнозирующие системы в различных областях науки, техники и торговли. К таким же процессам относится и составление расписания занятий, которое до сих пор во многих учебных заведениях разрабатывается вручную на основе многолетнего опыта. Современные IT-технологии предлагают средства, позволяющие наилучшим образом организовать и автоматизировать любой процесс, в том числе и учебный.

Планирование расписания учебных занятий [3] – это задача комбинаторного характера, особенностью которой является огромная размерность и наличие большого числа ограничений в сложной форме. Фактический, в настоящее время, не существует универсальных методов, которые могли бы решить такие задачи. Прямое применение математической (классической) теории расписания к задаче составления учебных занятий не представляется возможным. Тем не менее, есть ряд эвристических и переборных методов, которые поддаются программированию.

Существует мнение, что опытный специалист отвечающий за составление расписание сможет составить его таким образом, что оно будет отвечать всем интересам учебного процесса и научно-методической жизни образовательной организации. Однако с этим нельзя согласиться. Ручное решение данной задачи требует больших затрат времени, высокой квалификации специалиста, в тоже время результат такого решения часто получается далёк от оптимального. После ввода исходной информации требуется её согласование, в то время как на этапе анализа этих данных, может сложится ситуация, когда получение требуемого расписания может быть невозможна. Во время разработки расписания также может возникнуть тупиковая ситуация. Все эти факторы могут потребовать изменения исходных данных и ослабления ограничений, и в данной ситуации без человека не обойтись.

Без внесения изменений в расписание составленной авторизированной системой, данное расписание не будет иметь практической ценности. В этом плане очень важной является поддержка данного процесса автоматизированными методами и процедурами. Основное преимущество в автоматизации составления расписания занятий заключается в устранение массы рутинной работы, такой как:

* поиск вариантов внесения очередных данных в расписание;
* проверка выполнения требований;
* поиск случайных ошибок в готовом расписание;
* оформление расписания на бумаге в виде различных таблиц (для обучающихся, для преподавателей, групп, покабинетного);
* уменьшение затрат времени связанных с составлением расписания занятий.

Компьютер в данном случае выступает инструментом, существенно увеличивающим возможности человека в данном рабочем процессе, так как человек не в состоянии переработать и проанализировать такое же большое количество вариантов и факторов, влияющих на них, как компьютер.

В последние годы осуществляются множественные попытки совершенствования учебно-методических и научных процессов в образовательной организации путём построения алгоритмов оптимизации задач планирования учебной работы с использованием вычислительной техники и использования программного обеспечения от Корпорации Microsoft, в данном случае Microsoft Excel. Анализ состояния этих разработок позволяет сделать следующие выводы:

1. Разработка и внедрение образовательными организациями автоматизированных систем управления осуществляется в инициативном, свободном порядке и эти работы, как правило, направлены на решение конкретных проблем. Разобщенность групп исследователей, разработчиков программах продуктов привело к большому множеству систем, направленных на разработку расписания занятий, рассчитанных на обслуживание только конкретной образовательной организации.
2. Многие системы возлагают на специалиста отвечающего за разработку расписания всю ответственность за учёт реальных требований. В частности, учёт требований преподавателей, ограничения на количество проводимых занятий в день, в неделю, а также остальных факторов, влияющих на оптимальное расписание занятий – все эти и многие другие рутинные задачи в таких системах приходится решать специалисту самостоятельно, чаще всего методами перебора, что конечно же сказывается на качестве и временных затрат.
3. Имеющиеся программы имеют весьма неудобный интерфейс для ввода исходных данных и редактирования полученного результата.

При разработке алгоритмов автоматизированной системы управления составлением расписания занятий остро стоит проблема в создании универсальных алгоритмов, учитывающих специфику условий всех факторов и каждой конкретной задачи. Такие алгоритмы должны быть достаточно гибкими, то есть без существенного их изменения можно было бы включать и исключать, а также добавлять новые требования для системы разработки расписания. Однако попытки решать такую задачу каким-либо одним универсальным алгоритмом на данный момент не представляется возможным. Алгоритмы, решающие широкий класс задач, не в состоянии дать той эффективности, которую обеспечивают конкретные, адаптированные с учётом конкретных условий алгоритмы.

Для систем разработки расписания занятий характера сильная зависимость от специфики конкретной образовательной организации уже на уровне математических моделей и исходных данных, что в значительной мере затрудняет процесс внедрения и использования типовых систем. Систему, созданную в одной образовательной организации, обычно без изменений и каких-либо доработок невозможно эффективно использовать в другой. К тому же многие системы создавались достаточно давно и не могут затронуть весь спектр изменений в образовательном процессе изменившийся в связи с нововведениями в образовательные стандарты РФ, поэтому не в состоянии эффективно решать поставленные перед ними задачи.

На данный момент в исследуемой образовательной организации используются процессы составления расписания, которые можно представить, как полуавтоматические, так как большую часть этих процессов специалист выполняет в ручном режиме. Данный процесс можно сопоставить следующему алгоритму действий: происходит обработка большого количества входной информации, что способствует внушительном затратам времени, а также не исключает человеческий фактор, после анализа этих данных, и составления, так называемых набросков, данные вводятся в заранее подготовленные таблицы Microsoft Excel, после чего происходит вывод на печать.

2 Обзор существующих продуктов

К настоящему времени разработан ряд программных продуктов выполняющие автоматизацию процесса разработки расписания занятий образовательной организации.

Из ряда разработанных программных продуктов стоит выделить наиболее популярные и которые в той или иной мере заслужили признания в учебных процессах образовательных организаций:

* «Экспресс-расписание Колледж»;
* «Ректор-Колледж»;
* Конфигурация «1С: Автоматизированное составление расписания. Колледж».

2.1 «Экспресс-расписание Колледж»

Экспресс-расписание Колледж [4] – программа для автоматизации составления расписания учебных занятий в колледжах, техникумах и профессиональных лицеях. Программа автоматически составляет основное расписание, позволяет вести ежедневные изменения расписания, учет выполненных часов, формирует разнообразные отчёты.

Программный продукт имеет набор возможностей, а именно:

* автоматическое составление основного расписание на 1 (2, 3 или 4) недели;
* ведение текущего расписания и возможность ежедневной замены уроков, преподавателей, аудиторий;
* составление расписания для нескольких смен, а также потоков с разными сроками начала и окончания обучения;
* работа с подгруппами;
* планирование отсутствия и занятости учителей, аудиторий, групп;
* печать основного и текущего расписания, листа замены;
* выборка занятий для ведения журнала;
* учет норм СанПиН.

Также стоит отметить что данный программный продукт зарегистрирован в Министерстве образования, получил рекомендации Департамента образования, и внедрён более чем в 450 учебных заведений страны.

К его плюсам можно отнести большой функционал, отзывчивую техническую поддержку, обновления, большое количество обучающих видеоуроков.

К минусам данного программного продукта стоит отнести сложность интеграции продукта в учебный процесс, удовлетворительное, с точки зрения восприятия, юзабилити, сложность и неудобство в управление.

Цена программного продукта составляет 7990 рублей, что приемлемо для учебных заведений.

2.2 «Ректор-Колледж»

«Ректор-Колледж» [5] – программа предназначена для составления расписания занятий в колледжах. Программа состоит из четырёх разделов: «Списки», «Нагрузки», «Расписание» и «Замены».

Программный продукт имеет набор возможностей, а именно:

* составление расписание для любого учебного заведения;
* ручной и автоматический режим работы;
* расписание занятий можно сохранить в форматах Microsoft Word, Excel или HTML;
* лёгкость (работа практический на любом персональном компьютере под управлением системы Windows старше XP);

К его плюсам можно отнести лёгкость и простоту освоение, небольшие требовательность к характеристикам персонального компьютера.

К минусам можно отнести меленький функционал, не затрагивающий, как заявлено в описание, всех аспектов учебного процесса, из этого следует, что данный программный продукт не подходит для любого учебного заведения, отсутствие обновлений и технической поддержки.

Представленный программный продукт не поддерживается с 2012 года, но из-за своей простоты и лёгкости всё же используется в некоторых образовательных организация. Данный программный продукт можно купить по цене от 7000 рублей.

2.3 Конфигурация «1С: Автоматизированное составление расписания. Колледж»

«1С: Автоматизированное составление расписания. Колледж» [6] относится к самостоятельному программному продукту, который обладает возможностью решать такие задачи, как: автоматизированное составление учебного расписания, оперативное управление кабинетами в общеобразовательных учреждениях среднего профессионального образования.

Программный продукт имеет набор возможностей, а именно:

* удобная форма «шахматка» для быстрой ручной модификации расписания перетаскиванием;
* составление расписаний в разрезе;
* семестров;
* отделений;
* сценариев («пессимистичный», оптимистичный);
* составление нескольких расписаний и выбор лучшего.
* учет пожеланий и возможностей преподавателей, групп студентов, помещений;
* загрузка справочников и учебных планов из Excel;
* оперативное изменение расписаний и составление «текущих» расписаний;
* компоновка расписаний по отделениям в расписания по колледжу;
* разграничение доступа подокументно.

Положительным является большой функционал, отличная техническая поддержка, обновления, простая интеграция в учебный процесс.

Недостатки – высокая цена (22 000 рублей) и сложность в обучении и эксплуатации.

3 Требования к программному продукту

Требуется обеспечить возможность ввода и редактирования необходимых для составления расписания занятий данных. Выходными данными должны служить необходимые для работы учебно-методического управления образовательной организации отчёты, а также расписания занятий.

Входные данные следующие:

* наименование специальности;
* название группы;
* наименование кабинета;
* название дисциплины;
* фамилия имя отчество преподавателя;
* время работы преподавателя;
* сводная выписка из учебного плана.

Выходные данные следующие:

* расписание занятий;
* сводная выписка;
* дисциплины преподавателя;
* список преподавателей и дисциплины, которые они ведут;
* время работы преподавателя;
* список групп для каждой специальности.

Интерфейс пользователя для данного программного продукта имеет большую значимость. Должен быть реализован удобный, понятный и ненагруженный интерфейс пользователя. Пользователю должна быть понятна цепочка действий для получения необходимого результата.

Крайне желательно обеспечить возможность вывода расписания занятий в программный продукт Microsoft Excel.

4 Техническое задание на разработку

4.1 Введение

4.1.1 Наименование программы

Наименование программы – «Системы электронного расписания для учебно-методического управления образовательных организаций». Далее по тексту «система».

4.1.2 Краткая характеристика области применения

Система предназначена к применению в ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса».

4.2 Назначения разработки

4.2.1 Функциональное назначение

Функциональным назначением системы является предоставление пользователю возможности внесения данных, с последующим, при необходимости, их модифицированием и редактированием с целью получения оптимального, расписанием занятий, а также выполнение необходимых отчётов для учебно-методического управления.

4.2.2 Эксплуатационное назначение

Конечными пользователями программы должны являться сотрудники ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса».

4.3 Требования к программе или программному изделию

4.3.1 Требования к функциональным характеристикам

Система должна обеспечить следующие функции:

* ввод необходимых для составления расписания данных, их редактирование и модификацию;
* вывод на печать оптимального расписания занятий, а также необходимых для работы учебно-методической части отчётов.

Входные данные должны быть организованы в виде ручного ввода в предоставленные системой поля и таблицы.

Система должна обеспечить удобный ввод, редактирование и модифицирование входных данных, а именно:

* наименования специальностей;
* названия групп;
* наименования кабинетов;
* названия дисциплин;
* фамилия имя отчество преподавателя;
* время работы преподавателя;
* сводную выписку из учебного плана;

Система должна обеспечить вывод данных по средствам отчётов в формате \*.pdf и \*.elxs, а именно:

* расписания занятий;
* сводной выписки;
* дисциплин преподавателя;
* списка преподавателей и дисциплины, которые они ведут;
* времени работы преподавателей;
* списка групп для каждой специальности.

Требования к временным характеристикам системы не предъявляются.

4.3.2 Требования к надёжности

Надежное (устойчивое) функционирование системы должно быть обеспеченно выполнением совокупности организационно-технически х мероприятий, перечень которых приведен ниже:

1. организацией бесперебойного питания технических средств;
2. использованием лицензионного программного обеспечения;
3. регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
4. регулярным выполнением требования ГОСТ 51188-98 «Защита информации. Испытания программных средств на наличие компьютерных вирусов».

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 5 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы системы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3.3 Условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Программа не требует проведения каких-либо видов обслуживания.

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц: системный администратор и конечный пользователь программы – оператор.

Системный администратор должен иметь высшее профильное образование и сертификаты компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить:

1. задача поддержания работоспособности технических средств;
2. задачи установки (инсталляции) и поддержания работоспособности системных программных средств – операционной системы;
3. задача установки (инсталляции) программы.

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

4.3.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ) [7], включающий в себя:

* процессор Intel Pentium или аналогичный с тактовой частотой не менее 1,8 ГГц;
* оперативную память объёмом не менее 1 ГиБ;
* жёсткий диск или аналогичное запоминающее устройство с объёмом свободной памяти не менее 1 ГиБ.

4.3.5 Требования к информационной и программной совместимости

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения не предъявляются.

Требования к исходным кодам и языкам программирования не предъявляются.

Системные программные средства, используемые программой, должны быть предоставлены лицензионной локализованной версией операционной системы семейства Microsoft Windows, не ниже XP SP3, а также набор компонентов Microsoft Access Database Engine 2016 Redistributable.

Данный набор компонентов облегчает передачу данных между существующими файлами Microsoft Office, такими как Microsoft Office Access (\*.mdb и \*.accdb файл) файлами и Microsoft Office программы Excel (\*.xls,\*.xlsx и \*.xlsb) файлами и другими источниками данных, таких как Microsoft SQL Server. Также в состав набора входит поддержка подключение к существующим текстовым файлам. ODBC и OLEDB драйверы [8].

Требования к защите информации и программ не предъявляются.

4.4 Требования к программной документации

4.4.1 Предварительный состав программной документации

Состав программной документации должен включать в себя:

* техническое задание;
* руководство по установке;
* руководство пользователя.

4.6 Порядок контроля и приемки

Контроль и приёмка разработки осуществляются на основе испытаний контрольно-отладочных примеров. При этом проверяется выполнение всех функций программы.

5 Модель предметной области

В процессе разработки системы стоит выделить два основных объекта которые будут взаимодействовать друг с другом. Один объект будет заниматься работой с данными, составлением расписания и всевозможными отчётами. Второй объект в свою очередь будет использовать полученные отчёты, в информационном плане или в рабочем. Вариант использования программного продукта представлен на рисунке 1.

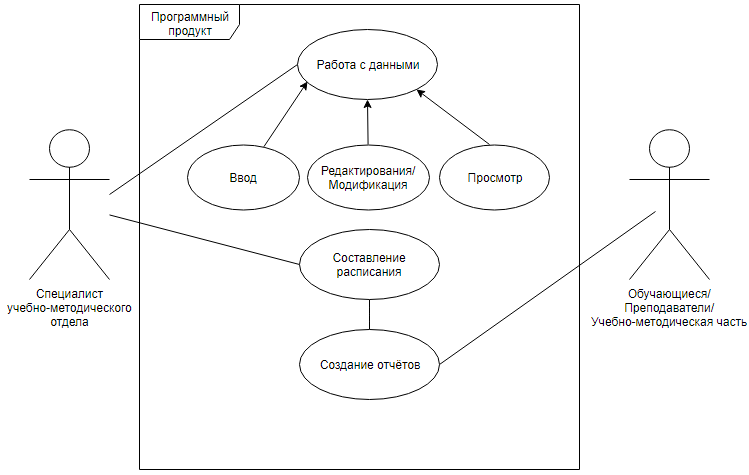


Рисунок 1 – Варианты использования программного продукта

Центральными сущностями предметной области являются расписание занятий, аудиторный фонд, дисциплины, преподаватели, специальности, а также рабочая нагрузка. Главные сущности представлены на рисунке 2 в виде диаграммы классов. Данная диаграмма служит для представления статической структуры модели система, различными взаимосвязями между отдельными сущностями предметно области, такими, как объекты и подсистемы, а также описания их структуры, типов и отношений.

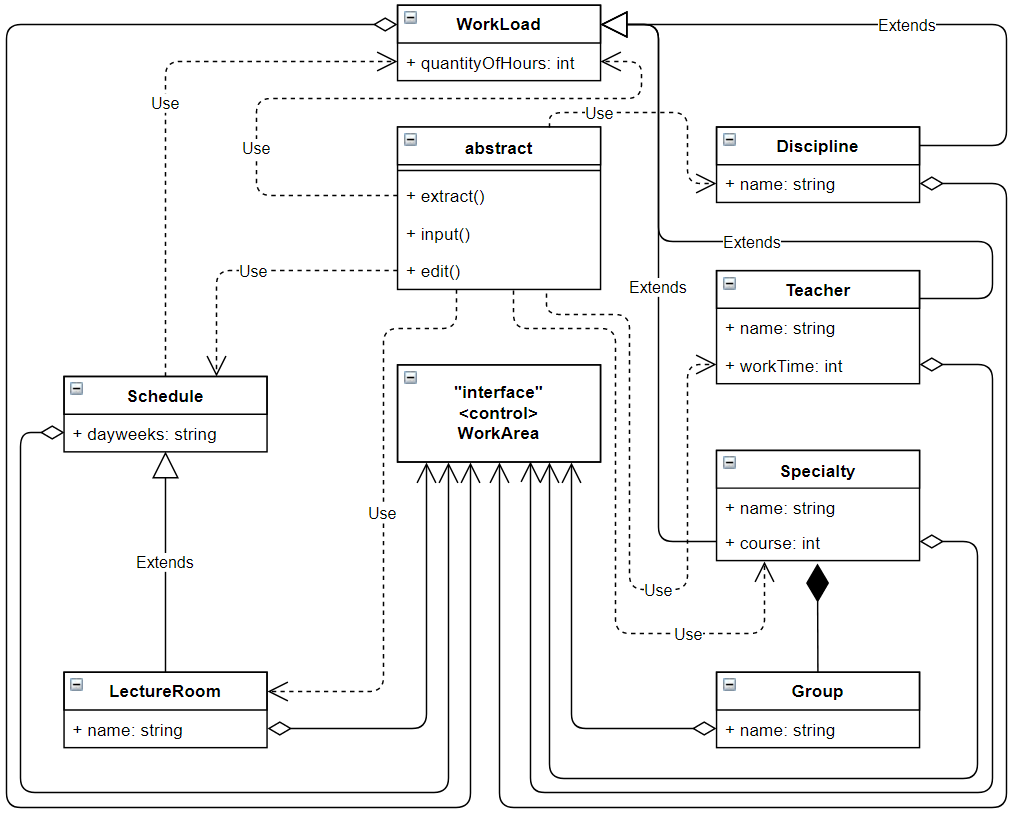


Рисунок 2 – Диаграмма классов основных сущностей

Предполагается в системе существование таких классов как: «Discipline» (Дисциплины), «Teacher» (Преподаватели), «Specialty» (Специальности) в которой, в свою очередь агрегатной ассоциацией относится класс «Group» (группы), «WorkLoad» (Рабочая нагрузка), «Shedule» (Расписание занятий), «LectureRoom» (аудиторный фонд), «abstract» и WorkArea (используемый интерфейс). Класс «abstract» является абстрактным: он имеет несколько методов необходимых для оптимальной работы и связан с классами «Discipline», «Teacher», «Specialty», «Group», «LectureRoom», «Shedule», «WorkLoad» отношением наследования. Класс «WorkArea» указывает на то, что данный класс отвечает за координацию действий, а также за интерфейс, предоставленный пользователю.

Классы «Teacher», «Specialty», «Group» должны предоставить данные для класса «WorkLoad», в свою очередь класс «WorkLoad» связан отношением наследования с классом «Schedule», которое на основе всех входных данных генерирует расписание, предоставляя его через интерфейс «WorkArea».

Динамические аспекты поведения системы отображаются на диаграммах кооперации и последовательности. На диаграмме кооперации поведение системы описывается на уровне отдельных объектов, которые обмениваются между собою сообщениями, чтобы достичь необходимой цели или реализовать некий вариант использования. Диаграмма кооперации представлена на рисунке 3.

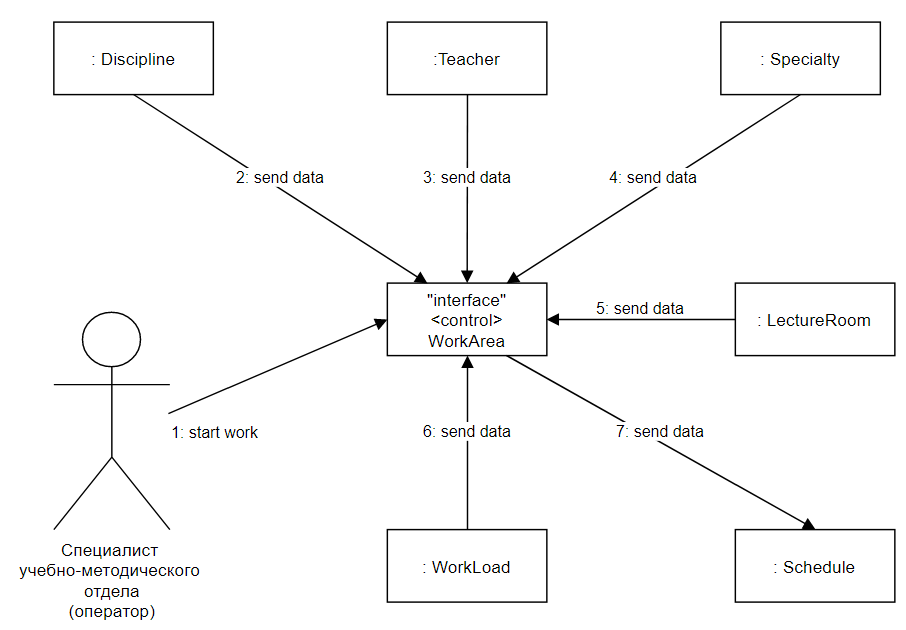


Рисунок 3 – Диаграмма кооперации

С помощью диаграмм последовательности можно описать полный процесс взаимодействий как своеобразный временной график «жизни» всей совокупности объектов, взаимодействующих между собою для реализации варианта использования системы, достижения цели или выполнения какой-либо задачи. Диаграмма последовательности представлена на рисунке 4.

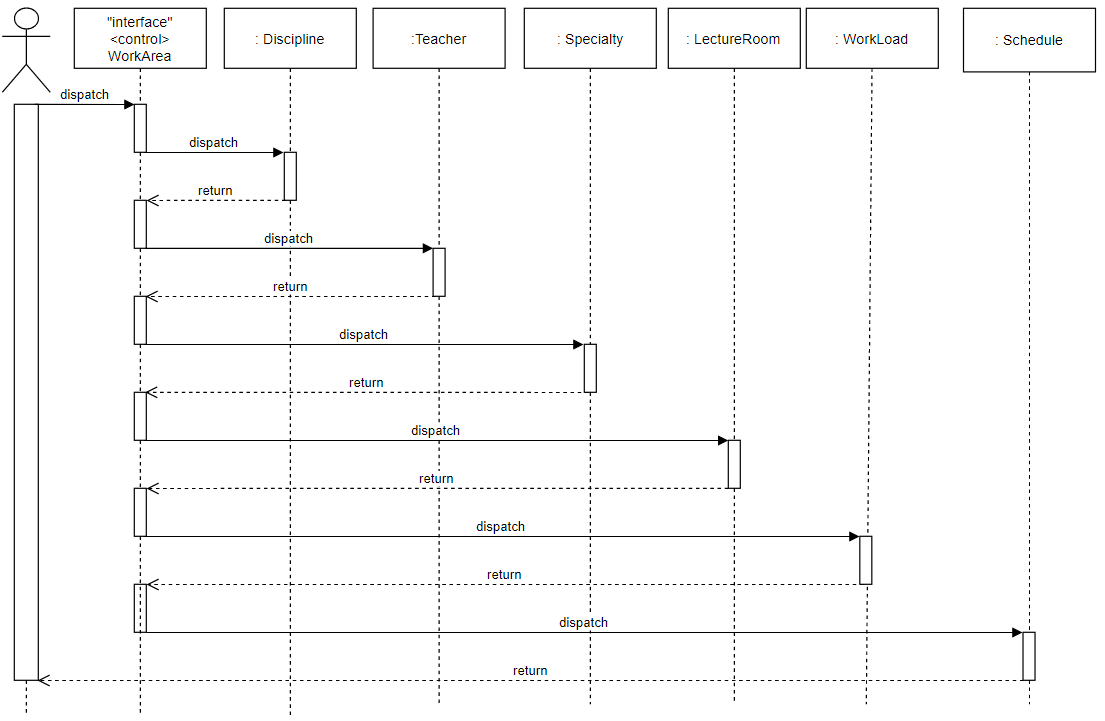


Рисунок 4 – Диаграмма последовательности

На диаграмме видно, что инициатором взаимодействия является Специалист учебно-методического отдела (оператор), который обладает фокусом управления на протяжении всей работы системы, постоянно контролируя её деятельность. С объектов «Discipline», «Teacher», «Specialty», «LectureRoom», «WorkLoad» собирается необходимая информация путем перехода фокуса управления от управляющего окна к соответствующему объекту и обратно. Когда необходима информация собрана, создаётся объект «Schedule» получающий временный фокус управления.

Далее требуется описать процесс изменения состояния системы или её подсистемы при реализации всех вариантов использования. При этом изменения состояний отдельных элементов системы может быть вызвано внешними воздействиями со стороны других элементов или извне системы. Именно для описания реакции системы на подобные внешние воздействия потребуется использовать диаграмму состояний. Данная диаграмма состояний представлена на рисунке 5.

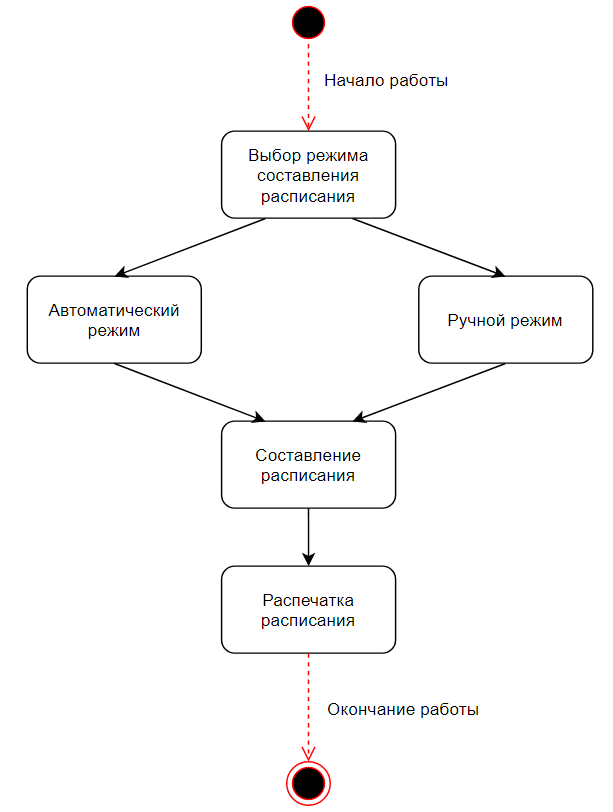


Рисунок 5 – Диаграмма состояний.

Для моделирования процесса выполнения операций используется так называемая диаграмма деятельности [9]. Каждое состояние на этой диаграмме соответствует выполнению некоторой элементарной операции, а переход в следующее состояние выполняется только после завершения операций в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а указывающими стрелками – переходы от одного состояния в другому. Таким образом, диаграмму деятельности можно считать частным случаем диаграмм состояний. Данная диаграмма деятельности представлена на рисунке 6.

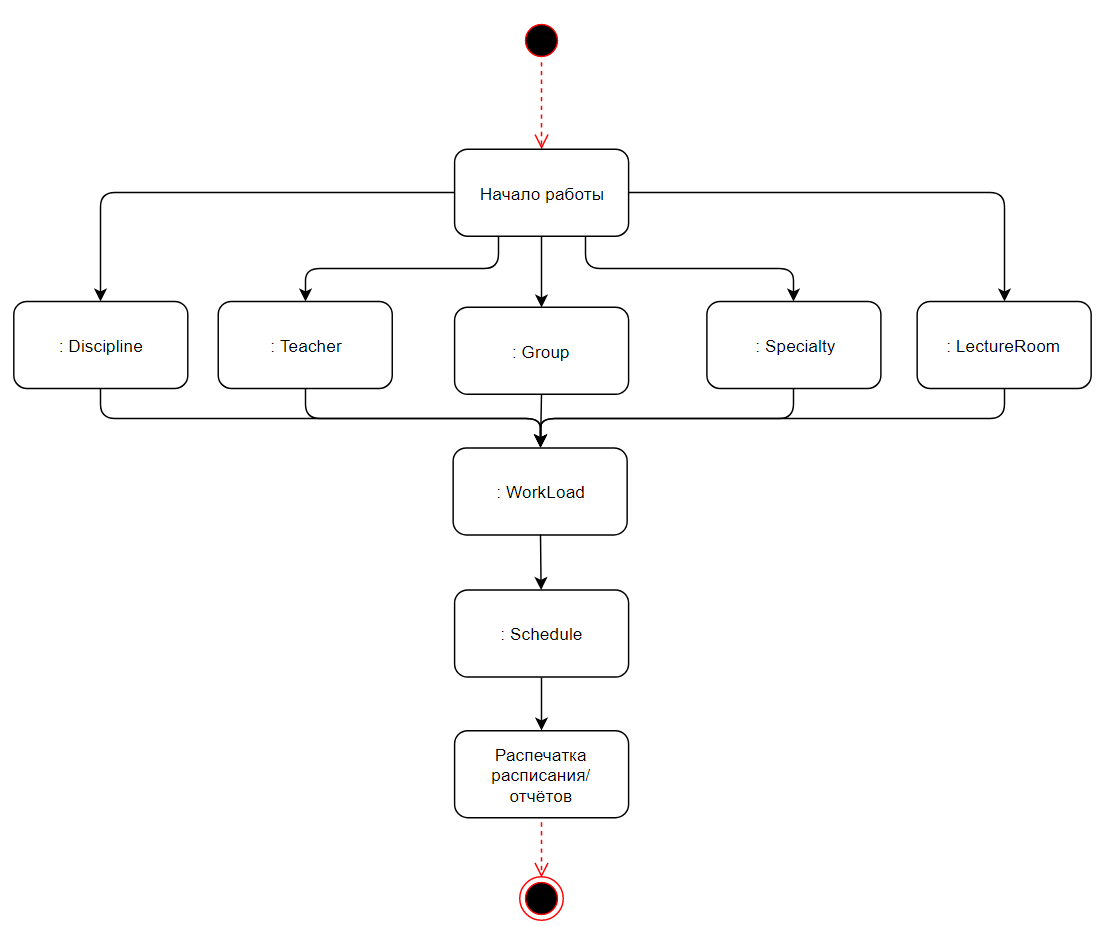


Рисунок 6 – Диаграмма деятельности.

Для создания конкретной физической системы необходимо некоторым образом реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно – физического представление модели. Для физического представления моделей систем будет использована так называемая диаграмма компонентов, которая позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. Данная диаграммы, как правило, строится непосредственно перед компьютерной реализацией систем. Диаграмма компонентов представлена на рисунке 7.

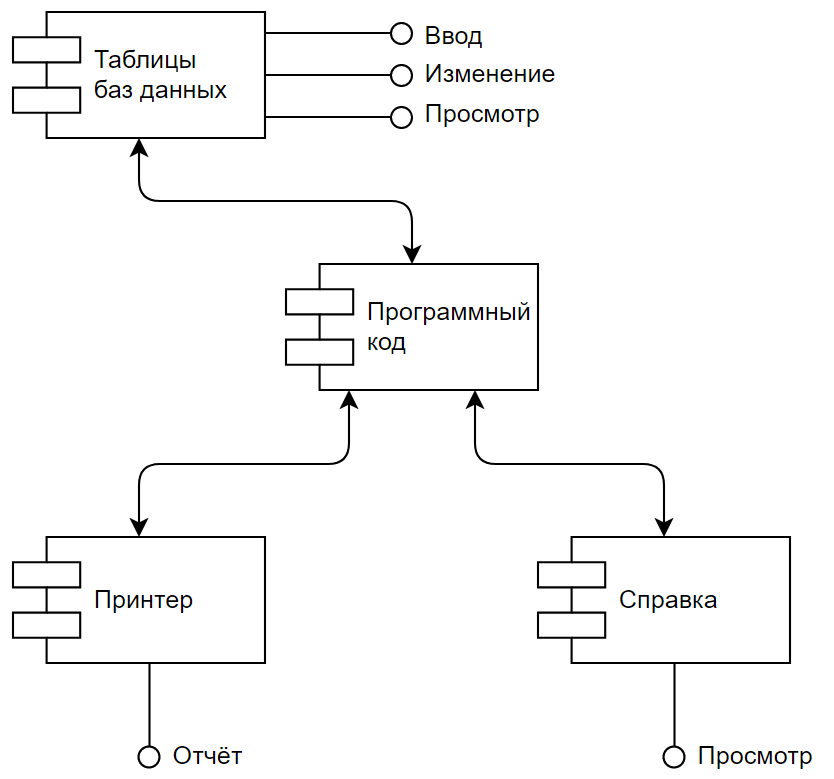


Рисунок 7 - Диаграмма компонентов.

6 Выбор метода и инструментов разработки

6.1 Выбор среды программирования

Microsoft Visual Studio [10] — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных, а именно можно создавать:

* приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
* веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
* приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect и "Интернета вещей";
* игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.

По умолчанию Visual Studio обеспечивает поддержку C#, C и C++, JavaScript, F# и Visual Basic. Visual Studio хорошо работает и интегрируется со сторонними приложениями, например Unity и Apache Cordova, с помощью расширений «Набор средств Visual Studio для Unity» и инструментов Visual Studio для Apache Cordova соответственно. Вы также можете самостоятельно расширить Visual Studio, создав собственные инструменты для выполнения специализированных задач [11].

База данных [12] – которая представляется в объективной форме, называется совокупностью самостоятельных материалов, упорядоченная таким образом, чтобы эти хранящиеся материалы можно было найти и обработать с помощью компьютера или другого вычислительного устройства. Большое количество разработчиков указывают на серьезную часто встречающуюся ошибку, которая состоит в ошибочном применении названия «база данных» вместо термина «система управления базами данных», что указывают на большую необходимость разграничения данных понятий.

На сегодняшний день основные направления развития современной информационной технологии составляют базы данных (БД) и системы управления базами данных (СУБД), роль которых как совместного средства хранения, обработки и доступа к огромным объемам упорядоченных данных постоянно увеличивается. В данном случае существенным фактом является постоянное увеличение объемов хранимой информации, хранимой в базе данных, что в свою очередь влечет за собой требование увеличения производительных мощностей таких систем.

Стремительно увеличивается таким же образом в различных применениях спрос на удаленный доступ к хранимой информации. Данный факт особенно проявляется при организации процесса логического манипулирования информации в системах баз данных, основой которых являются созданные современные экспертные системы. Стремительное развитие необходимости использования баз данных выдвигает ряд требований к системе управления базами данных: поддержка огромного числа типов представляемых данных и манипуляций над ними; естественные и эффективные размещения в БД различных отношений между объектами рассматриваемых предметных областей; осуществление поддержки верности данных и создание дедуктивных БД; обеспечение целостности информации хранимой в БД в большом диапазоне различных предметных областей и операционных систем; осуществление удобного управления распределенными БД, включение неоднородных баз данных; заметное повышение надежности функционирования БД.

Традиционные базы данных являются пассивными [13]. Информация часто помещается в базу данных пользователем или программным продуктом. Выборка объектов осуществляется опять-таки под воздействием внешних источников. Таким же образом под воздействием какого-либо внешнего источника информация меняет место своего нахождения в базе данных (например, переносится из исходной таблицы в другую). Бизнес-правила, используемые в содержимом базы данных (например, правила обновления определенного элемента данных, следующие за этим действия над предлагаемыми элементами как результат такого обновления), также управляются другим внешним источником. По-другому говоря, базы данных в традиционном представлении не являются активным элементов в информационных системах и вместо этого они выполняют организационную роль, которая направлена на обеспечение хранения данных. В последние годы эта роль все увеличивается, и важность концепции активных баз данных растет с каждым годом

Microsoft Office Access [14] или просто Microsoft Access — реляционная система управления базами данных (СУБД) корпорации Microsoft. Входит в состав пакета Microsoft Office. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Основные компоненты MS Access:

* построитель таблиц;
* построитель экранных форм;
* построитель SQL-запросов (язык SQL в MS Access не соответствует стандарту ANSI);
* построитель отчётов, выводимых на печать.

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и БД практически «с нуля» или написать оболочку для внешней БД.

Microsoft Jet Database Engine которая используется в качестве движка базы данных MS Access, является файл-серверной СУБД и потому применима лишь к приложениям, работающим с небольшими объёмами данных и при небольшом числе пользователей, одновременно работающих с этим данными. Непосредственно в Access отсутствует ряд механизмов, необходимых в многопользовательских базах данных, таких, например, как триггеры.

Microsoft Access Database Engine 2016 Redistributable - набор компонентов облегчает передачу данных между существующими файлами Microsoft Office, такими как Microsoft Office Access (\*.mdb и \*.accdb файл) файлами и Microsoft Office программы Excel (\*.xls,\*.xlsx и \*.xlsb) файлами и другими источниками данных, таких как Microsoft SQL Server. Также в состав набора входит поддержка подключение к существующим текстовым файлам. ODBC и OLEDB драйверы.

Встроенные средства взаимодействия MS Access со внешними СУБД с использованием интерфейса ODBC снимают ограничения, присущие Microsoft Jet Database Engine [15]. Инструменты MS Access, которые позволяют реализовать такое взаимодействие, называются «связанные таблицы» (связь с таблицей СУБД) и «запросы к серверу» (запрос на диалекте SQL, который «понимает» СУБД).

На данный момент актуальной версией Microsoft Access является версия 2015 года: Microsoft Access 2016 (из офисного пакета приложений Microsoft Office 2016).

В результате анализа модели данных была выбрана СУБД MS Access 2016 и IDE Microsoft Visual Studio 2017, а также объектная модель ADO.NET [16] реализующее непосредственный доступ к данным.

6.2 Выбор языка программирования

Последнее время С и С++ являются наиболее используемыми языками для разработки коммерческих и бизнес приложений. Эти языки устраивают многих разработчиков, но в действительности не обеспечивают должной продуктивности разработки. К примеру, процесс написания приложения на С++ зачастую занимает значительно больше времени, чем разработка эквивалентного приложения, скажем, на Visual Basic. Сейчас существуют языки, увеличивающие продуктивность разработки за счет потери в гибкости, которая так привычна и необходима программистам на С/С++. Подобные решения являются весьма неудобными для разработчиков и зачастую предлагают значительно меньшие возможности. Эти языки также не ориентированы на взаимодействие с появляющимися сегодня системами и очень часто они не соответствуют существующей практике программирования для Web. Многие разработчики хотели бы использовать современный язык, который позволял бы писать, читать и сопровождать программы с простотой Visual Basic и в то же время давал мощь и гибкость C++, обеспечивал доступ ко всем функциональным возможностям системы, взаимодействовал бы с существующими программами и легко работал с возникающими Web стандартами [17].

Учитывая все подобные пожелания, Microsoft разработала новый язык - C#. В него входит много полезных особенностей - простота, объектная ориентированность, типовая защищенность, "сборка мусора", поддержка совместимости версий и многое другое. Данные возможности позволяют быстро и легко разрабатывать приложения, особенно COM+ приложения и Web сервисы. При создании C#, его авторы учитывали достижения многих других языков программирования: C++, C, Java, SmallTalk, Delphi, Visual Basic и т.д. Надо заметить что по причине того, что C# разрабатывался с чистого листа, у его авторов была возможность, оставить в прошлом все неудобные и неприятные особенности (существующие, как правило, для обратной совместимости), любого из предшествующих ему языков. В результате получился действительно простой, удобный и современный язык, по мощности не уступающий С++, но существенно повышающий продуктивность разработок.

Очень часто можно проследить такую связь - чем более язык защищен и устойчив к ошибкам, тем меньше производительность программ, написанных на нем. К примеру, рассмотрим две крайности - очевидно это Assembler и Java. В первом случае вы можете добиться фантастической быстроты своей программы, но вам придется очень долго заставлять ее работать правильно не на вашем компьютере. В случае же с Java - вы получаете защищенность, независимость от платформы, но, к сожалению, скорость вашей программы вряд ли совместима со сложившимся представлением о скорости, например, какого-либо отдельного клиентского.

В C#, как в несомненно современном языке, также существуют характерные особенности для обхода возможных ошибок. Например, помимо упомянутой выше "сборки мусора", там все переменные автоматически инициализируются средой и обладают типовой защищенностью, что позволяет избежать неопределенных ситуаций в случае, если программист забудет инициализировать переменную в объекте или попытается произвести недопустимое преобразование типов. Также в C# были предприняты меры для исключения ошибок при обновлении программного обеспечения. Изменение кода, в такой ситуации, может непредсказуемо изменить суть самой программы. Чтобы помочь разработчикам бороться с этой проблемой C# включает в себя поддержку совместимости версий (vesioning). В частности, в отличии от C++ и Java, если метод класса был изменен, это должно быть специально оговорено. Это позволяет обойти ошибки в коде и обеспечить гибкую совместимость версий. Также новой особенностью является native поддержка интерфейсов и наследования интерфейсов. Данные возможности позволяют разрабатывать сложные системы и развивать их со временем.

В C# была унифицирована система типов, теперь вы можете рассматривать каждый тип как объект. Несмотря на то, используете вы класс, структуру, массив или встроенный тип, вы можете обращаться к нему как к объекту. Объекты собраны в пространства имен (namespaces), которые позволяют программно обращаться к чему-либо. Это значит что вместо списка включаемых файлов заголовков в своей программе вы должны написать какие пространства имен, для доступа к объектам и классам внутри них, вы хотите использовать. В C# выражение using позволяет вам не писать каждый раз название пространства имен, когда вы используете класс из него. Например, пространство имен System содержит несколько классов, в том числе и Console. И вы можете писать либо название пространства имен перед каждым обращением к классу, либо использовать using как это было показано в примере выше.

Важной и отличительной от С++ особенностью C# является его простота. К примеру, всегда ли вы помните, когда пишите на С++, где нужно использовать "->", где "::", а где "."? Даже если нет, то компилятор всегда поправляет вас в случае ошибки. Это говорит лишь о том, что в действительности можно обойтись только одним оператором, а компилятор сам будет распознавать его значение. Так в C#, оператор"->" используется очень ограничено (в unsafe блоках, о которых речь пойдет ниже), оператор "::" вообще не существует. Практически всегда вы используете только оператор "." и вам больше не нужно стоять перед выбором.

Еще один пример. При написании программ на C/С++ вам приходилось думать не только о типах данных, но и о их размере в конкретной реализации. В C# все упрощено - теперь символ Unicode называется просто char (а не wchar\_t, как в С++) и 64-битное целое теперь - long (а не \_\_int64). Также в C# нет знаковых и беззнаковых символьных типов. [18]

В C#, также как и в Visual Basic после каждого выражения case в блоке switch подразумевается break. И более не будет происходить странных вещей если вы забыли поставить этот break. Однако если вы действительно хотите чтобы после одного выражения case программа перешла к следующему вы можете переписать свою программу с использованием, например, оператора goto.

Многим программистам было не так легко во время изучения C++ полностью освоиться с механизмом ссылок и указателей. В C# нет указателей. В действительности нетривиальность указателей соответствовала их полезности. Например, порой, трудно себе представить программирование без указателей на функции. В соответствии с этим в C# присутствуют Delegates - как прямой аналог указателя на функцию, но их отличает типовая защищенность, безопасность и полное соответствие концепциям объектно-ориентированного программирования.

Хотелось бы подчеркнуть современное удобство C#. Когда вы начнете работу с C#, а, надеюсь, это произойдет как можно скорее, вы увидите, что довольно большое значение в нем имеют пространства имен. Уже сейчас, на основе первого примера, вы можете судить об этом - ведь все файлы заголовков заменены именно пространством имен. Так в C#, помимо просто выражения using, предоставляется еще одна очень удобная возможность - использование дополнительного имени (alias) пространства имен или класса.

Современность C# проявляется и в новых шагах к облегчению процесса отладки программы. Традиционным средством для отладки программ на стадии разработки в C++ является маркировка обширных частей кода директивами #ifdef и т.д. В C#, используя атрибуты, ориентированные на условные слова, вы можете куда быстрее писать отлаживаемый код.

В наше время, когда усиливается связь между миром коммерции и миром разработки программного обеспечения, и корпорации тратят много усилий на планирование бизнеса, ощущается необходимость в соответствии абстрактных бизнес процессов их программным реализациям. К сожалению, большинство языков реально не имеют прямого пути для связи бизнес логики и кода. Например, сегодня многие программисты комментируют свои программы для объяснения того, какие классы реализуют какой-либо абстрактный бизнес объект. C# позволяет использовать типизированные, расширяемые метаданные, которые могут быть прикреплены к объекту. Архитектурой проекта могут определяться локальные атрибуты, которые будут связанны с любыми элементами языка - классами, интерфейсами и т.д. Разработчик может программно проверить атрибуты какого-либо элемента. Это существенно упрощает работу, к примеру, вместо того чтобы писать автоматизированный инструмент, который будет проверять каждый класс или интерфейс, на то, является ли он действительно частью абстрактного бизнес объекта, можно просто воспользоваться сообщениями основанными на определенных в объекте локальных атрибутах [19].

В ходе анализа, а также предлагаемых IDE Visual Studio языков программирования, был сделан выбор в пользу объектно-ориентированного языка C#, так как данный язык непосредственно связан с .NET и предоставляется полный спектр возможностей для работы с СУБД MS Access.

7 Реализация разработки

7.1 Построение моделей данных

На этапе проектирования системы, проанализировав предметную область становиться возможным построение даталогической модели. Даталогическая модель базы данных является моделью логического уровня и представляет собою отображение логических связей между элементами данных [20]. Суть этапа даталогического проектирования заключается в отображении инфологической модели предметной области базы данных на структурные и процедурные возможности. Они предоставляют: тип структуры данных, например, реляционная модель, выбранный тип СУБД, которая эту модель данных поддерживает, технологию и средства прикладного программировании.

Для каждого локального представления указывают:

* определения типов сущностей;
* определения типов связей;
* определения атрибутов и связывание их с типами сущностей и типами связей;
* определения доменов атрибутов;
* определения атрибутов, являющихся первичными, альтернативными или внешними ключами;
* определения ограничений целостности.

Соответствующие спецификации должны быть документированы. Анализ документации позволяет сформировать перечень атрибутов.

На основе представления о предметной области, для выявления информационных объектов необходимо осуществить анализ ее входной информации.

Входную информацию принято делить на условно-постоянную и оперативно-учетная информация.

К условно-постоянной информации относятся:

* план учебного расписания;
* списки отделений и факультетов;
* списки преподавателей;
* перечень дисциплин, аудиторий;
* время пары.

К оперативно-учетной информации относятся:

* списки курсов и групп;
* дата;
* количество студентов в группах.

Перечень первичных сущностей и их атрибутов представим в виде таблицы 1.

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование сущности | Атрибуты |
| 1. | План расписания | Специальность  Группа  Тип недели  День недели  ФИО преподавателя  Дисциплина  Время  Аудитория |
| 2. | Списки преподавателей | № преподавателя  ФИО преподавателя |
| 3. | Перечень дисциплин | № дисциплины  Наименование дисциплины |
| 4. | Список специальностей | № специальности  Наименование специальности |
| 5. | Списки групп | № группы  Наименование группы |
| 6. | Списки аудиторий | № аудитории  Наименование аудитории |

Таблица1 - Перечень первичных сущностей с атрибутами

Необходимо проанализировать каждый атрибут на наличие связи с другими атрибутами. Атрибут приобретает смысл тогда, когда он связан с другими, обладающими смысловым единством.

По ходу выполнения разработки предстоит детализировать некоторые атрибуты, для упрощения и предупреждения неприятных последствий, а также создать некоторые вспомогательные информационные объекты для более интуитивного и детализированного хранения данных.

Среди выявленных атрибутов, описывающих сущность, можно выделить один или несколько атрибутов, которые определяют экземпляр сущности. Такой атрибут будет называться ключом. Между ключом и другими атрибутами существует функциональная зависимость. Результат занесем в таблицу 2.

Таблица 2 – Информационные объекты и их атрибуты.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Информационные объекты | Ключ | Атрибуты |
| 1 | Аудиторный фонд | \* | ID кабинета  Наименование кабинета |
| 2 | Список дисциплин | \* | ID дисциплины  Наименование дисциплины |
| 3 | Кабинеты дисциплин | \*  \* | ID дисциплины  ID кабинета  Рейтинг кабинета  Курс |
| 4 | Список преподавателей | \* | ID преподавателя  ФИО преподавателя |
| 5 | Рабочее время преподавателя | \*  \* | ID преподавателя  Номер занятия  День недели (1..6) |
| 6 | Дисциплины преподавателя | \*  \* | ID преподавателя  ID дисциплины |
| 7 | Список специальностей | \* | ID специальности  Наименование специальности |
| 8 | Список групп | \*  \* | ID специальности  ID группы  Наименование группы |
| 9 | Специальность – курс – преподаватель дисциплины | \*  \*  \*  \* | ID специальности  ID дисциплины  ID преподавателя  Курс |
|  |  |  |  |
|  |  |  |  |
| Продолжение таблицы 2 | | | |
| 10 | Учебная (рабочая) нагрузка | \*  \*  \*  \* | ID специальности  Курс  ID дисциплины  Семестр  Количество часов |
| 11 | Календарь | \*  \* | День недели  Тип недели |
| 12 | Семестр | \* | ID семестра  Количество недель |
| 13 | Расписание занятий |  | Тип недели  День недели  Номер занятия  Специальность  Группа  Аудитория  ФИО преподавателя  Наименование дисциплины |

Проанализировав предметную область, выделив все атрибуты, обозначив ключи, получаем 13 информационных объектов:

* аудиторный фонд;
* список дисциплин;
* кабинеты дисциплин;
* список преподавателей;
* рабочее время преподавателя;
* дисциплины преподавателя;
* список специальностей;
* список групп;
* специальность – курс – преподаватель дисциплины;
* учебная (рабочая) нагрузка;
* календарь;
* семестр;
* расписание занятий.

Следующим этапом разработки является определение связей между сущностями.

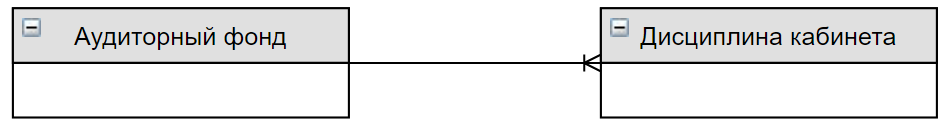


Рисунок 8 – связь один-ко-многим объектов «Аудиторный фонд» и «Дисциплина кабинета»

Для связи аудиторного фонда со списком дисциплин, потребуется дополнительный информационный объект, название которого «Дисциплина кабинета», так как возможна ситуация, когда кабинет может быть привязан к определённой дисциплине, также в данном объекте был внесён атрибут «Рейтинг кабинета» для определения приоритета аудитории для определённой дисциплины при составлении расписания. В данном случае тип связи между информационным объектом «Аудиторный фонд» и «Дисциплины кабинетов» будет один-ко-многим. Связь между этими объектами будет обеспечиваться с помощью атрибута ID аудитории.

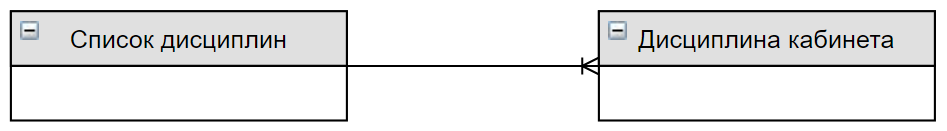


Рисунок 9 – связь один-ко-многим объектов «Список дисциплин» и «Дисциплина кабинета»

Тип связи между информационными объектами «Дисциплина» и «Дисциплина кабинета», будет осуществляться по такому же свойству, как и в случае со связью «Аудиторный фонд» и «Дисциплина кабинета», то есть один-ко-многим. Связь между этими объектами будет обеспечиваться с помощью атрибута ID дисциплины.

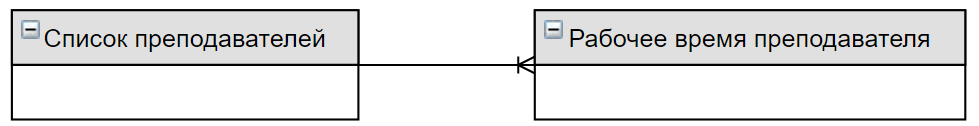


Рисунок 10 – связь один-ко-многим объектов «Список преподавателей» и «Рабочее время преподавателя»

Для связи информационных объектов «Список преподавателей» и «Рабочее время преподавателя» будет использован тип один-ко-многим, так как у преподавателя могут быть разные рабочие дни, а также разные промежутки времени в эти дни, выходные, и т.д. Связь между этими объектами будет обеспечиваться с помощью атрибута ID преподавателя.

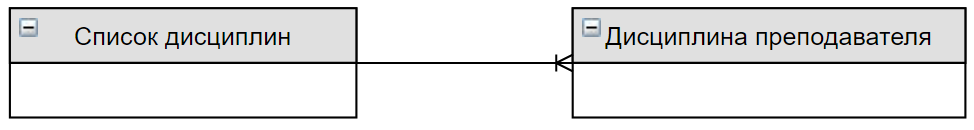


Рисунок 11 – связь один-ко-многим объектов «Список дисциплин» и «Дисциплина преподавателя»

Для связи списка дисциплин со списком преподавателей, потребуется дополнительный информационный объект «Дисциплины преподавателя», так как возможна ситуация что один преподаватель может преподавать несколько дисциплин, или же у одной дисциплины может быть несколько преподавателей. В данном случае тип связи между информационным объектом «Дисциплина» и «Дисциплина преподавателя» будет один-ко-многим. Связь между этими объектами будет обеспечиваться с помощью атрибута ID дисциплины.

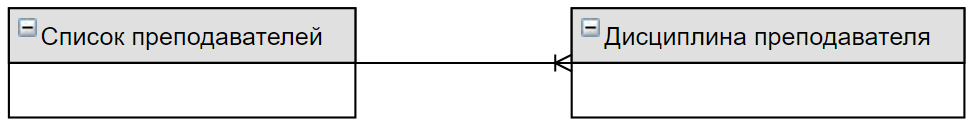


Рисунок 12 – связь один-ко-многим объектов «Список преподавателей» и «Дисциплина преподавателя»

Тип связи между информационными объектами «Список преподавателей» и «Дисциплина преподавателя», будет осуществляться по такому же свойству, как и в случае со связью «Список дисциплин» и «Дисциплина преподавателя», то есть один-ко-многим. Связь между этими объектами будет обеспечиваться с помощью атрибута ID преподавателя.

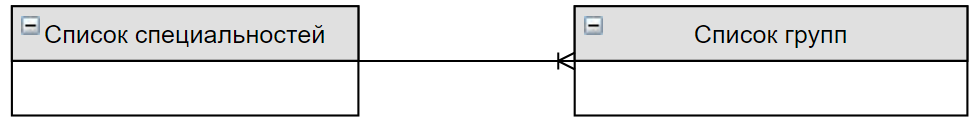


Рисунок 13 – связь один-ко-многим объектов «Список специальностей» и «Список групп»

Тип связи один-ко-многим, так как каждая специальность может иметь несколько групп. Связь между этими объектами будет обеспечиваться с помощью атрибута ID специальности.

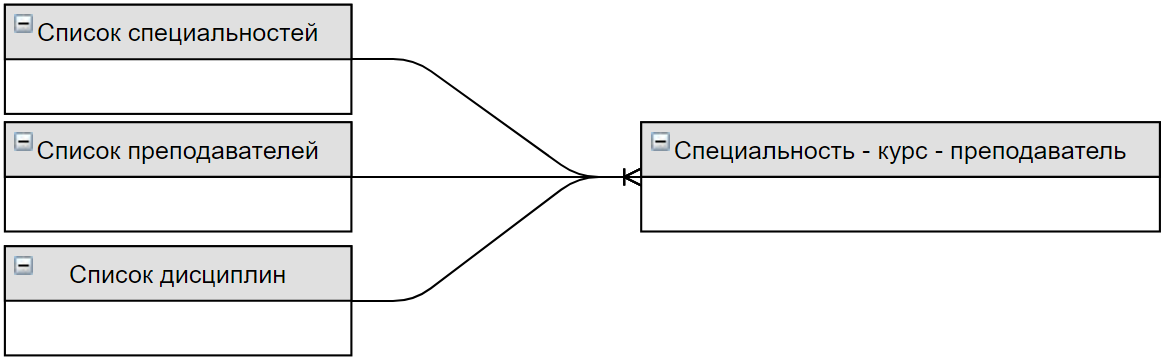


Рисунок 14 – связь один-ко-многим между несколькими объектами и объектом «Специальность – курс – преподаватель»

Стоит отметить и ситуацию, когда преподаватель может вести занятия только на определённом курсе определённой специальности, для этого был создан информационный объект «Специальность – курс – преподаватель дисциплины», где можно хранить данные с необходимым нам условием. Связь между объектами «Список специальностей», «Список преподавателей» и «Список дисциплин» обеспечивается с помощью соответствующих атрибутов ID специальности, ID преподавателя и ID дисциплины.

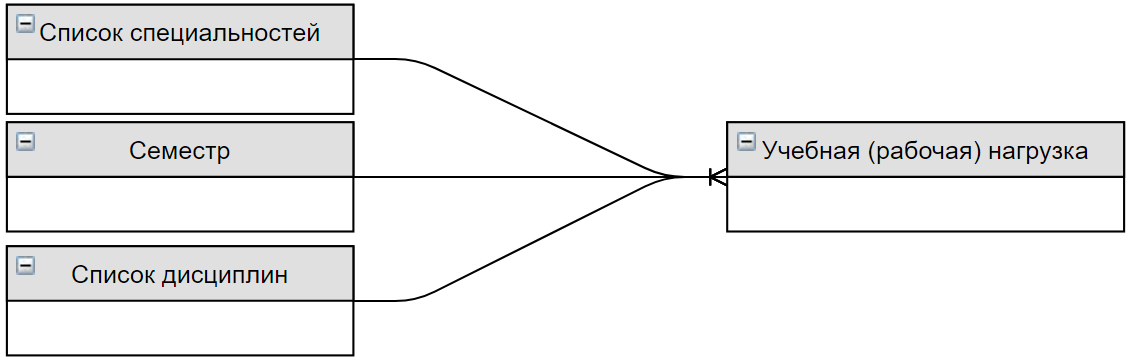


Рисунок 15 – связь один-ко-многим между несколькими объектами и объектом «Учебная (рабочая) нагрузка»

Для хранения данных о рабочей (учебной) нагрузки был создан объект «Учебная (рабочая) нагрузка». В рабочую нагрузку должны входить сведения о наименование специальности, курсе, дисциплине, семестре, и соответственно о количестве часов для данной дисциплины, для хранения этих данных были созданы соответствующие атрибуты и определена связь между подходящими сущностями, а именно ID дисциплины объекта «Список дисциплин» и ID специальности объекта «Список специальностей. Тип связи между объектами была определена как один-ко многим от объекта «Список дисциплин» и «Список специальностей» к объекту «Учебная (рабочая) нагрузка».

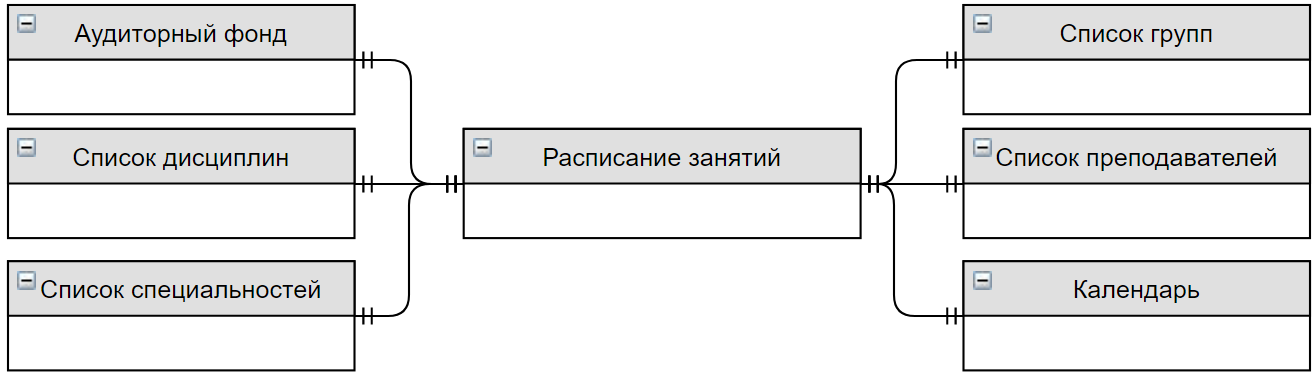


Рисунок 16 – связь один-к-одному между несколькими объектами и объектом «Расписание занятий»

Тип связи один-к-одному, так как к сущности «Расписание» относятся индивидуальные атрибуты «Специальность» «Группа» «Аудитория» «ФИО преподавателя». Связь с сущностями осуществляется с помощью индивидуальных атрибутов.

Обозначив основные связи можно построить информационно-логическую модель [20].

Информационно логическая модель – модель данных отображающих предметную область в виде совокупности информационных объектов и структурных связей между ними. Представление данной модели изображено на рисунке 17.

|  |  |
| --- | --- |
| Рисунок 17 – Информационно логическая модель данных. |  |

Следующим этапом проектирование является физическая модель данных [20].

При построении физической модели данных описываются свойства и типы данных, которые используются в процессе разработки системы. Представление физической модели данных изображено на рисунке 18.

|  |  |
| --- | --- |
| Рисунок 18 – Физическая модель данных. |  |

На основе построенных логической и физической модели данных можно приступить к построению реляционной схемы БД. Её реализация будет происходить непосредственно в выбранной ранее СУБД MS Access последней версии (2016).

Для лучшего преобразования моделей данных необходимо выполнить следующие условия:

* каждая сущность превращается в таблицу (отношение);
* каждый атрибут становиться столбцом;
* компоненты ключа превращаются в первичный ключ (primary key).

В результате реляционная база данных будет состоять из 13 таблиц:

* cabinet;
* discipline;
* cabinet\_discipline;
* teacher;
* work\_time;
* discipline\_teacher;
* specialty;
* group;
* specialty\_course;
* fragment\_summary;
* calendar;
* semestre;
* second\_solution.

Также в процессе разработки системы для промежуточных расчётов, а также для оптимального хранения данных потребуются, ещё несколько таблиц данных, а именно:

* calculate\_hours;
* first\_solution.

Таким образом в процессе преобразования и анализа необходимых данных реляционная база данных будет состоять из 15 таблиц.

Для создания схемы данных, средствами MS Access необходимо внести все таблицы и впоследствии связать их между собой, основываясь на построенной физической модели данных.

В процессе создания связи между таблицами, следует обеспечить целостность данных, поставив флажок в соответствующем предложении MS Access, так же после установки флажка, если имеется возможность следует обеспечить «Каскадное обновление связанных полей» и «Каскадное удаление связанных записей».

Создание схемы данных представляет поддержание целостности взаимосвязанных данных при дальнейшей корректировке таблиц.

Основываясь на выше сказанные задачи при проектировании физической модели, можно получить схему данных изображенную на рисунке 19.

|  |  |
| --- | --- |
| Рисунок 19 – Реляционная модель данных. | C:\Users\Sabot\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1222.jpg |

7.2 Разработка интерфейса пользователя

Разработка интерфейса пользователя для системы составления расписания является очень важной задачей. Так как в операционной системе Windows большинство пользователей предпочитают графический интерфейс пользователя, было принято решение разработать классическое Windows Form приложение.

В настоящее время пользовательский интерфейс является очень важным для любого программного продукта, будь то веб-приложение или же классическое настольное. Помимо самого интерфейса так же большое внимание стоит уделять моментам самого взаимодействия пользователя с программным продуктом, так как для достижения необходимого результата порой приходится пройти длинный путь, и этот путь может запутать пользователя. В таком случае результат может не оправдать ожиданий или вовсе заставить пользователя отказаться от данного программного продукта. Для решения данной проблемы был применён UX [21] на основе работы специалиста учебно-методического отдела, а также личного опыта использования аналогичных программных продуктов.

UX - это User Experience (дословно: «опыт пользователя»). То есть это то, какой опыт/впечатление получает пользователь от работы с интерфейсом. Удается ли ему достичь цели и на сколько просто или сложно это сделать.

Так же при разработке пользовательского интерфейса были использованы решения, предоставленные компанией Telerik [22]. Решения в виде компонентов пользовательского интерфейса, которые облегчают работу как программиста, так и дизайнера интерфейсов. Данные компоненты имеют возможность смены темы, которые предоставлены данным решением, или созданием новых, посредствам специального приложения. В данном случае была выбрана тема «materia».

7.3 Испытания

Оценка корректности работы программного обеспечения выполнялась посредствам тестирования отдельных компонентов их функций и процедур, а также корректности выводимых системой отчётов.

Для тестирования была создана копия базы данных. На основании вымышленных данных проводилось непосредственное заполнение таблиц БД через пользовательский интерфейс, что также являлось тестированием на обеспечение целостности данных в связях и работоспособности интерфейса пользователя. После ввода необходимых данных, были сформированы отчёты с оптимальными данными, на основании которых можно сделать вывод что тестирование прошло успешно.

При разработке компонентов применялся подход к разработке, называемый «разработка через тестирование». При таком подходе разработка осуществляется в цикле: написание теста конкретной функции; написание кода приложения, позволяющего успешно выполнить тест; рефакторинг написанного кода. Обзор метода приводится в [23].

8 Разработка проектной документации

8.1 Руководство пользователя

Данное программное обеспечение предназначено для составления расписания занятий учебно-методической части образовательной организации.

Предполагается, что пользователь обладает навыками работы в операционной системе Windows, а также работе с классическими Windows Desktop приложениями.

Программное обеспечение позволяет осуществлять ввод данных в соответствующие формы, осуществлять их редактирование, модификацию. На основе входных дынных осуществляется вывод соответствующих отчётов в форматах \*.pdf и \*.exls.

Для применения программного обеспечения требуется персональный компьютер с центральным процессором архитектуры amd64 или i386 с тактовой частотой не менее 1 ГГц, не менее 512 МиБ оперативной памяти и не менее 150 МиБ свободного пространства в файловой системе.

Программное обеспечение работает только в операционных системах Windows. Требуется использовать дистрибутив программного обеспечения совместимый с архитектурой процессора.

Дистрибутив программного обеспечения состоит из архивного файла, внутри которого располагаются исполняемые файлы и необходимые файлы данных. В имени файла архива содержится дата выпуска данной версии программного обеспечения.

Для подготовки к работе следует распаковать содержимое архива в любое удобное место файловой системы. Для распаковки потребуется до 50 МиБ свободного места на файловой системе. После распаковки архива в файловой системе должны появиться следующие директории: bin, mainPanelControl, obj, release.

Для проверки работоспособности следует перейти в директорию release и убедиться, что исполняемые файлы, входящие в состав программного обеспечения, запускаются.

Следует запустить исполняемый файл schedule.exe. В результате успешного запуска программы должна появиться форма начальная форма программного обеспечения.

Приведём описание действий пользователя.

После успешного запуска исполняемого файла schedule.exe, открывается главное окно программного продукта, рисунок 20.

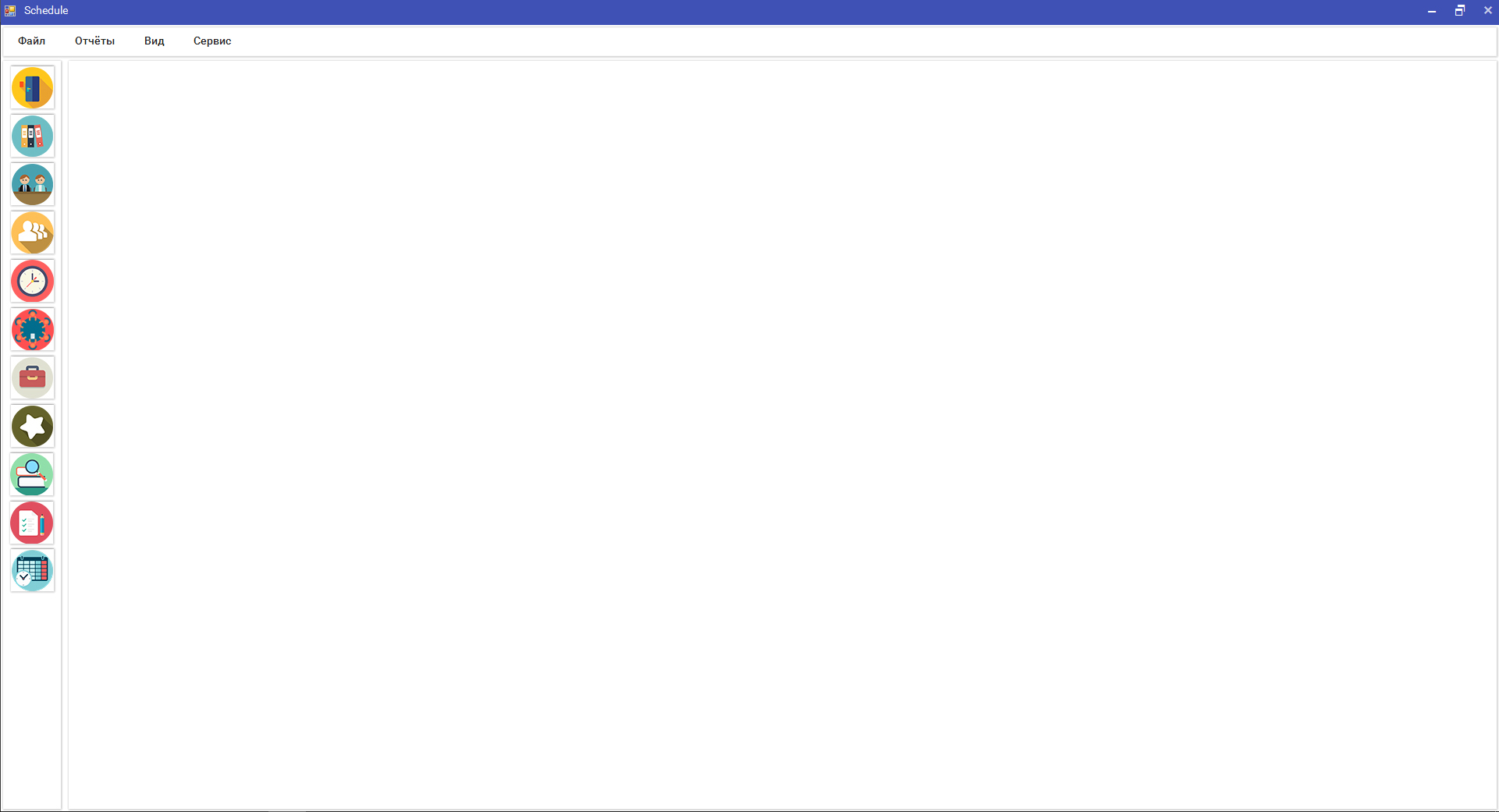


Рисунок 20 – Главное окно программного продукта.

Главное окно программного продукта представляет собою форму в минималистическом стиле, на которой расположено:

* главное меню;
* панель элементов;
* рабочая область.

Главное меню имеет следующую иерархию:

* файл;
  + выход;
* отчёты;
  + расписание занятий;
  + сводная выписка;
  + дисциплины преподавателя;
  + список преподавателей и дисциплин;
  + время работы преподавателя;
  + список групп;
* вид;
  + настройки;
* сервис;
  + о программе.

Панель элементов представляет собою набор, расположенных вертикально, кнопок, каждая из которых отвечает за вызов в рабочую область соответствующей ей панели.

Заполнение необходимых для отчётности данных, происходит в очередном порядке, то есть, пока не будет заполнена предыдущая таблица, следующая будет недоступна для заполнения. Из этого следует, что для достижения необходимого результат пользователь пройдёт полный путь, от заполнения, с непосредственным обучением, до получения отчётности. Каждая кнопка на панели может быть подписана в Настройках программы, или же для сохранения минималистичного дизайна, предоставлена так называемая функция подсказки при наведении. Пример вызова нескольких панелей изображён на рисунке 21. Стоит ответить, что данная функция возможна только при условии, что предыдущие таблицы, соответствующие каждой кнопке, заполнены. Количество открытых одновременно панелей ограниченно лишь их общим количеством, а также шириной разрешения монитора. Также действует правило, по которому последняя открытая панель отображается первой в списке.

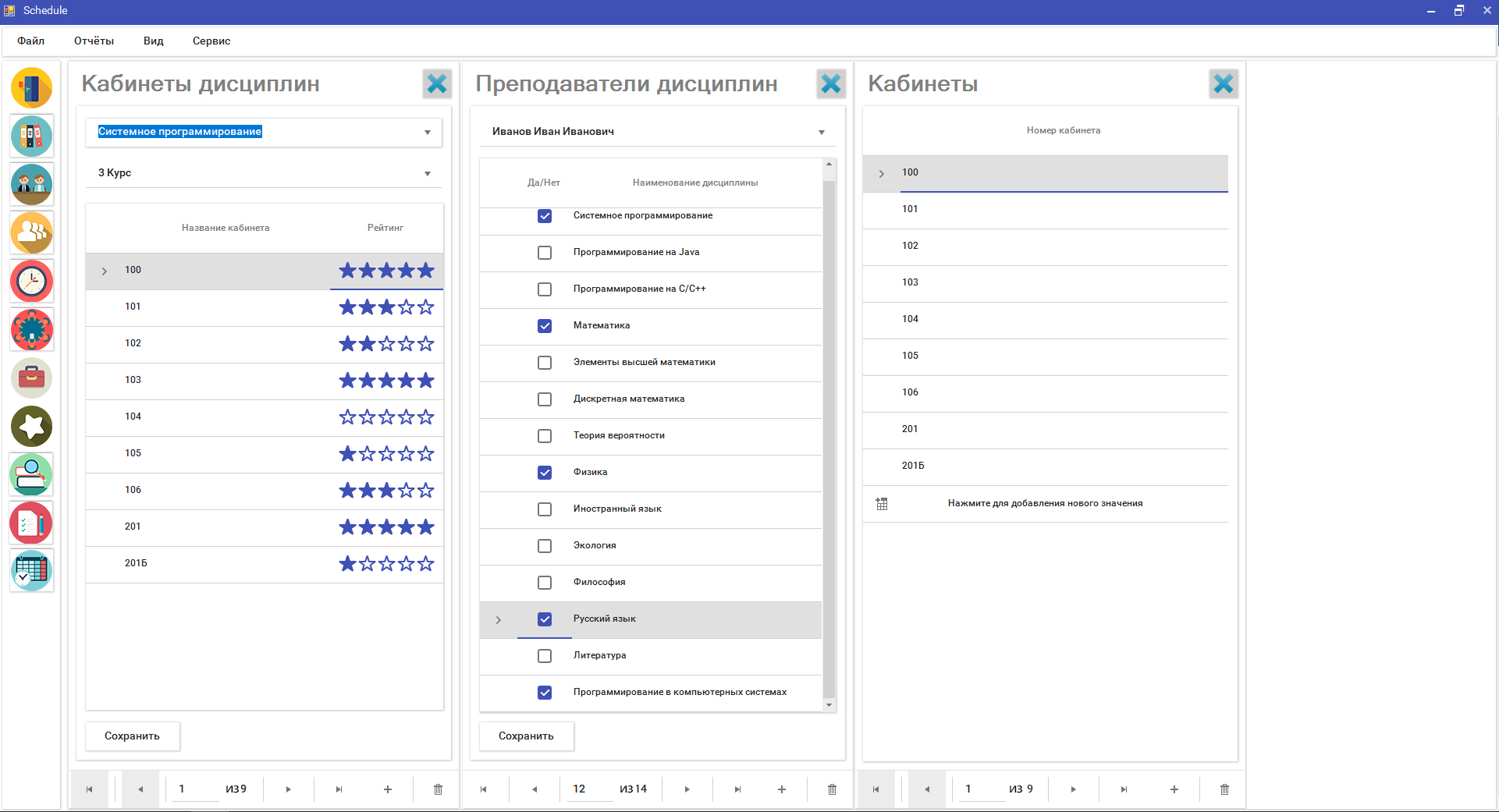


Рисунок 21 – Вызов несколько панелей в рабочую область.

При ошибках, связанных с программным обеспечением (ОС, драйверы устройств), восстановление работоспособности возлагается на ОС.

При неверных действиях пользователя (перенос БД, неверно указанный путь к БД) программа выдаёт сообщение об ошибке и возвращает управление форме приложения. При возникновении данной аварийной ситуации восстановление работоспособности возлагается на системного администратора.

9 Оценка эффективности программного продукта

Под эффективностью понимается отношение эффекта, достигаемого от внедрения программного обеспечения, к затратам. Под затратами подразумеваются совокупные затраты на приобретение, установку, настройку и поддержку программного обеспечения [24].

Главный эффект от использования программного продукта состоит в уменьшении временных затрат и увеличении производительности деятельности учебно-методического отдела образовательной организации по созданию расписания занятий.

Деятельность по созданию расписания занятий можно разделить на несколько действия. Далее можно оценить увеличение или уменьшение временных затрат и производительности учебно-методического отдела в сравнении с другими программными продуктами, рассмотренными в разделе «Обзор существующих продуктов».

Если на совершении действия пользователь экономит *ΔT = B1 – A1* единиц времени, то повышение производительности можно оценивать по следующей формуле:

|  |  |
| --- | --- |
|  | (1) |

где, P1 – увеличение производительности действия, в процентах;

B1 – время на действие без использования программного обеспечения;

A1 – время на действие при использовании программного обеспечения.

Тогда общее увеличение производительности составит:

|  |  |
| --- | --- |
|  | (2) |

Оценка увеличения производительности конкретных действия пользователя при использовании программного продукта приведена в таблице 3. Увеличение произвольности оценивается в сравнении с использование методов получение расписания занятий на объекте Заказчика.

|  |  |  |
| --- | --- | --- |
| Действия | B1, мин | A1 , мин |
| Получение оптимального расписания занятий | 330 | 90 |
| Получение необходимой отчётности | 30 | 2 |
| Подготовительные процессы | 60 | 180 |

Таблица 3 – Производительность действия пользователя

Таким образом, если сравнивать производительность учебно-методического отдела при использовании программного продукта и при использовании методов используемыми в настоящее время в ЧУ ПОО КТУИС, производительность значительно выше

При использовании аналогов, рассмотренных в разделе «Обзор существующих продуктов» производительность пользователя не будет кардинально отличаться.

Для оценки эффективности программного продукта в сравнении с аналогами может использоваться индекс эксплуатационно-технического уровня продукта. Этот показатель является обобщенной характеристикой эксплуатационных свойств продукта, его возможностей, степени новизны.

Индекс эксплуатационно-технического уровня определяется как:

|  |  |
| --- | --- |
|  | (3) |

где *n* – число рассматриваемых показателей;

*w*1 – коэффициент весомости показателя в долях единиц;

*x*1 – относительный показатель качества, устанавливаемый экспертным путём по выбранной шкале оценки (например, от 1 до 5).

Выполним сравнение эксплуатационно-технических уровней разработанного программного продукта и наиболее близкого аналога - «Ректор-Колледж» по следующему ряду показателей:

* удобство пользовательского интерфейса;
* скорость доступа к данным;
* скорость обучения пользователя;
* необходимая отчётность;
* нетребовательность к ресурсам.

Сравнение приводится в таблице 4.

Таблица 4 – Показатели качество в сравнении с аналогом.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатель |  | Проект | | Аналог | |
|  |  |  |  |
| Удобство пользовательского интерфейса | 0,2 | 5 | 1 | 2 | 0,4 |
| Скорость доступа к данным | 0,3 | 4 | 1,2 | 4 | 1,2 |
| Скорость обучения пользователя | 0,4 | 4 | 1,6 | 3 | 1,2 |
| Необходимая отчётность | 0,3 | 3 | 0,9 | 4 | 1,2 |
| Нетребовательность к ресурсам | 0,15 | 4 | 0,6 | 5 | 0,75 |
|  | | 5,3 | | 4,75 | |

Отношение двух индексов называют коэффициентом технического уровня проектируемого программного продукта по отношению к аналогу:

|  |  |
| --- | --- |
|  | (4) |

где *J1* – индекс эксплуатационно-технического уровня продукта;

*J2* – индекс эксплуатационно-технического уровня аналога.

Считается, что разработка системы обоснована, если A > 1.

Индекс эксплуатационно-технического уровня для программного продукта

5,3, индекс для аналога 4,75. Коэффициент технического уровня 1,11, что подразумевает оправданность разработки программного продукта.

10 Мероприятия по охране труда

10.1 Общие требования безопасности при работе за компьютером

Все персональные ЭВМ, используемые в работе, должны соответствовать СанПиН 2.2.2/2.4.1340-03 [25].

К работе на компьютере допускаются лица не моложе 18 лет, обученные и прошедшие инструктаж по охране труда.

Лица, работающие с персональными электронно-вычислительными машинами (ПЭВМ) более 50 % рабочего времени (профессионально связанные с эксплуатацией ПЭВМ), должны проходить обязательные предварительные (при поступлении на работу) и периодические медицинские осмотры в установленном порядке.

Все лица, работающие на компьютерах, должны знать меры защиты и приемы оказания первой медицинской помощи при поражении электрическим током.

При работе с ПК имеет место наличие вредных факторов. Работа на компьютерах связана с нагрузками на зрение, опорно-двигательный аппарат, а также нагрузками эмоционального и психического характера. Нагрузки на зрение обычно проявляются в раздражении и усталости глаз. Из всех видов расстройств опорно-двигательного аппарата наиболее часто встречаются боли в спине, шее и плечевом поясе.

К категории вредных факторов психологического характера можно отнести несколько видов расстройств, которые объединены вместе, поскольку они вызывают изменения в эмоциональной сфере. Усталость является наиболее часто встречающимся расстройством.

Для снижения воздействия неблагоприятных факторов и получения оптимального результата при работе на компьютере необходимо уделить внимание:

* правильному выбору оборудования;
* должной подготовке и обучению операторов;
* регулярному техническому обслуживанию;
* правильному освещению рабочего места.

Помещения, в которых эксплуатируются ПЭВМ, должны иметь естественное и искусственное освещение. Оконные проемы должны быть оборудованы регулируемыми жалюзи или занавесями. Необходимо также следить, чтобы свет из окна или от осветительных приборов не отражался от экрана монитора.

Помещения, в которых эксплуатируются ПЭВМ, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации.

Подключение ПЭВМ в электрическую сеть производится только через специально установленные штепсельные розетки и вилки с заземляющим контактом. Подключение компьютера проводом без вилки запрещается.

В помещениях, оборудованных ПЭВМ, должна проводиться ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

Для обеспечения нормируемых значений освещенности необходимо проводить чистку стекол оконных рам и светильников не реже двух раз в год и своевременно заменять перегоревшие лампы.

Рабочие столы следует размещать таким образом, чтобы экраны компьютеров были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева. Освещение не должно создавать бликов на поверхности экрана.

Экран монитора должен находиться от глаз пользователя на расстоянии 60–70 см, но не ближе 50 см с учётом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы.

Требования к рабочему столу должны быть следующими:

Требования к рабочему столу должны быть следующими:

* рабочий стол должен регулироваться по высоте в пределах 68–80 см. При отсутствии возможности регулировки высоты, она должна составлять 72,5 см;
* ширина рабочей поверхности стола под персональным компьютером – 80 - 140 см, чтобы с обеих сторон было свободное пространство 30–50 см;
* рабочий стол должен иметь пространство для ног высотой не менее 60 см, шириной не мене 50 см, глубиной на уровне колен, не менее 45 см.

Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не мене 30 см, глубину не менее 40 см.

Клавиатуру следует располагать на поверхности стола на расстоянии 10 -30 см от края стола.

Для снижения утомления при работе на ПЭВМ необходимо соблюдать режим труда и отдыха. Рациональный режим труда и отдыха предусматривает соблюдение регламентированных перерывов.

10.2 Безопасность перед началом работы

Перед началом работы оператор должен совершить следующие действия:

1. отрегулировать освещённость на рабочем месте, убедиться в отсутствии отражений (бликов) на экране и клавиатуре, а также встречного светового потока. Для устранения бликов, вызванных ярким солнечным светом, необходимо закрыть окно жалюзи;
2. осмотреть рабочее место, проверить правильность подключения оборудования к электросети, убедится в отсутствии повреждений электрических кабелей;
3. протереть салфеткой поверхность экрана;
4. убедиться в наличии специального коврика для устройства «мышь»;
5. проверить правильность установки стола, кресла, подставок для ног, положения оборудования, угла наклона экрана; при необходимости отрегулировать положение кресла, а также расположить оборудование так, чтобы исключить неудобные позы, длительное напряжение тела.

10.3 Безопасность во время работы за компьютером

Требования к безопасности при работе на компьютере включают следующие позиции:

1. при необходимости, отрегулировать яркость и контрастность монитора. Не следует делать изображение слишком ярким;
2. чтобы монитор не загрязнялся пылью, надо регулярно стирать ее с экрана тканью с антистатиком;
3. не следует открывать крышку монитора и трогать находящиеся там детали;
4. минимально допустимое расстояние между лицом пользователя и монитором персонального компьютера 50 см.

Работник обязан:

1. выполнять только ту работу, которая ему была поручена, и по которой он прошёл соответствующую подготовку и обучение;
2. содержать в порядке и чистоте рабочее место в течение всего рабочего дня;
3. держать открытыми все вентиляционные отверстия устройств;
4. корректно закрыть все активные задачи при необходимости прекращения работы на относительно короткое время;
5. выполнять санитарно-гигиенические требования;
6. использовать регламентированные перерывы в работе для отдыха и выполнения рекомендованных упражнений для глаз, шеи, рук, ног;
7. соблюдать расстояние от глаз до экрана в пределах 60–70 см.

10.4 Безопасность при возникновении аварийной ситуации

При возникновении аварийной ситуации оператор должен совершить следующие действия:

1. если на частях оборудования компьютера обнаружен электрический ток (искрение, ощущение тока и т. д.), немедленно отключить компьютер от электрической сети и доложить о неисправности непосредственному руководителю. Без его указания к работе не приступать;
2. при прекращении подачи электроэнергии компьютер необходимо отключить;
3. при возникновении возгорания, немедленно отключить компьютер. Сообщить о пожаре руководителям работ, принять меры по ликвидации очага возгорания первичными средствами пожаротушения и вызвать пожарную охрану;
4. при несчастном случае в первую очередь освободить пострадавшего от травмирующего фактора. При освобождении пострадавшего от действия электрического тока необходимо следить за тем, чтобы самому не оказаться в контакте с токоведущей частью. Необходимо о происшедшем несчастном случае сообщить руководителю работ и оказать первую медицинскую помощь пострадавшему.

10.5 Безопасность по окончании работы

По окончании работы производятся следующие действия:

1. закрыть все активные задачи (программы);
2. отключить питание системного блока;
3. отключить питание всех периферийных устройств;
4. отключить общее питание ПК;
5. очистить внешние поверхности ПК мягкой хлопчатобумажной тканью;
6. навести порядок на рабочем столе и убрать все ненужные документы.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была исследована возможность автоматизации процесса составления расписания занятий, а также рассмотрен вопрос внедрение системы в ЧУ ПОО «Краснодарский техникум управления, информатизации и сервиса».

Проанализирована предметная область исследования – актуальность электронного расписания занятий в учреждении среднего профессионального образования. Результатом исследования предметной области является непротиворечивая модель предметной области с выявленными сущностями и связями между ними.

Данная модель использована при выявлении требований к программному обеспечению, повлияла на выбор инструментов и языков для реализации программного обеспечения, явилась основанием для структур данных, использованных в реализации.

В результате анализа требований, сформирован набор задач, которые реализуют функциональное назначение системы, предложен способ решения задачи – автоматизация рабочего процесса специалиста учебно-методического управления образовательной организации. Разработана система электронного расписания занятий для учебно-методического управления образовательной организации.

Данное решение позволяет значительно увеличить производительность учебно-методического управления, за счёт сокращения временных затрат на составления расписания занятий, а также необходимой отчётности.

По мере работы над проектом, обозначился ряд новых направлений для развития разработанной системы. К новым возможностям, отсутствующим в текущей реализации, относится, например, реализация нескольких алгоритмов построения расписания занятий, списки замен, оставшееся количество часов учебной нагрузки, «запоминание» внесённых ранее изменений при составлении нового расписания занятия, реализация дополнительной отчётности. Такие возможности должны быть проанализированы и реализованы в будущих версиях программного обеспечения.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. КТУИС | Краснодарский техникум управления информатизации и сервиса [Электронный ресурс]. – [Б.м.: б.и.], 2017. Режим доступа: <http://ktyis.ru/istoriya-texnikuma/> (дата обращения: 3.06.2017).
2. Российское образование. Федеральный портал. [Электронный ресурс]. – [Б.м.: б.и.], 2017. Режим доступа: <http://www.edu.ru/abitur/act.86/index.php> (дата обращения: 8.07.2017).
3. Приказ Минобрнауки России от 14 июня 2013 г. № 464 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам среднего профессионального образования» (и изменениями и дополнениями) [Текст].
4. Программный центр. Компьютерные программы и материалы. [Электронный ресурс]. – [Б.м.: б.и.], 2017. Режим доступа: <http://pbprog.ru/products/programs.php?ELEMENT_ID=376> (дата обращения: 20.09.2017).
5. Расписание занятий/уроков: «Ректор» - программа лидер [Электронный ресурс]. – [Б.м.: б.и.], 2013. Режим доступа: <http://rector.spb.ru/raspisanie-ptu-4c.php> (дата обращения: 20.09.2017).
6. Отраслевые и специализированные решения 1С: Предприятие [Электронный ресурс]. – [Б.м.: б.и.], 2017. Режим доступа: <http://solutions.1c.ru/asp_spo> (дата обращения: 21.09.2017).
7. IBM-PC-совместимый компьютер. Свободная энциклопедия. [Электронный ресурс]. – [Б.м.: б.и.], 2014. Режим доступа: <https://ru.wikipedia.org/wiki/IBM-PC-совместимый_компьютер> (дата обращения: 19.08.2017).
8. Microsoft Access Database Engine 2016 Redistributable. Additional Information [Электронный ресурс]. – [Б.м.: б.и.], 2017. Режим доступа: <https://www.microsoft.com/en-us/download/details.aspx?id=54920> (дата обращения: 10.06.2017).
9. Леоненков А.В. Самоучитель UML: Самоучитель [Текст] / Леоненков А.В., - 2-е изд., перераб. и доп. – СПб:БХВ-Петербург, 2015. – 418с.
10. Visual Studio IDE. Latest documentation on Visual Studio 2017 [Электронный ресурс]. – [Б.м.: б.и.], - 2017. Режим доступа: <https://msdn.microsoft.com/en-us/eu-eu/library/dn762121.aspx> (дата обращения: 4.06.2017).
11. Понамарев В.А. Программирование на C++/С# в Visual Studio .NET: Пособие [Тескт] / Понамарев В.А. – СПб:БХВ-Петербург, 2015 г. – 340с.
12. Агальцов В. П. Базы данных. В 2-х кн. Книга 2. Распределенные и удаленные базы данных: учебник [Текст] / В.П. Агальцов. – М.: ИД «Форум» : ИНФРА-М, 2017. – 271с.
13. Лазицкас Е.А. Базы данных и сестемы управления базами данных: Учебное пособие [Текст] /Лазицкас Е.А., Загумёнников И.Н., Гилевский П.Г. – Мн.:РИПО, 2016 г. – 268с.
14. Microsoft Access. Свободная энциклопедия. [Электронный ресурс]. – [Б.м.: б.и.], - 2015. Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Access> (дата обращения: 23.06.2017).
15. Справочное руководство. Поставщик OLE DB для Jet [Электронный ресурс]. [Б.м.: б.и.], - 2015. Режим доступа: <https://technet.microsoft.com/ru-ru/library/ms175866(v=sql.105).aspx> (дата обращения: 25.06.2017).
16. ADO.NET. Documentation. [Электронный ресурс]. [Б.м.: б.и.], 2017. Режим доступа: <https://msdn.microsoft.com/ru-ru/library/e80y5yhx(v=vs.110).aspx> (дата обращения 27.07.2017).
17. Гуриков С.Р. Введение в программирование на языке Visual C# : учебное пособие [Текст] / С.Р. Гуриков. – М.: ФОРУМ : ИНФРА-М, 2013. – 448с.
18. Ben Albahari. C# 6.0 in a Nutshell: The Definitive Reference [Текст]/ Ben Albahari. O'Reilly Media – 2015. – 1136c.
19. Фленов М.Е. Библия C# [Текст] / Фленов М.Е. БХВ-Петербург. – 2016. – 544с.
20. Дадян. Э.Г. Методы, модели, средства хранения и обработки данных: учебник [Текст] / Э.Г. Дадян, Ю.А. Зеленков. – М.: Вузовский учебник: ИНФРА-М, 2017 – 168с.
21. Джейми Леви: UX-стратегия. Чего хотят пользователи и как и это дать. [Текст] / Джейми Леви. Питер, 2017. – 304с.
22. Telerik. Free Encyclopedia. [Электронный ресурс], [Б.м.: б.и.], 2014, режим доступа: <https://en.wikipedia.org/wiki/Telerik> (дата обращения: 2.09.2017).
23. Martin, R. The Cycles of TDD [Электронный ресурс]. — [Б. м.: б. и.], 2014. — Режим доступа: http://blog.cleancoder.com/uncle-bob/2014/12/17/TheCyclesOf TDD.html (дата обращения: 09.12.2017).
24. Гойко Аджич. Impact Mapping. Как повысить эффективность программных продуктов и проектов по их разработке. [Текст] / Гойко Аджич. Отдельное издание, 2017 – 86с.
25. КонсультантПлюс – надёжная правовая поддержка. Постановление Главного государственного санитарного врача РФ от 03.06.2003 N 118 (ред. от 21.06.2016) [Электронный ресурс], — [Б. м.: б. и.], 2016, режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_42836/> (дата обращения: 4.01.2018).

ПРИЛОЖЕНИЯ А

Исходный код основных модулей

MainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using Telerik.WinControls;

using schedule.mainPanelControls;

using Telerik.WinControls.UI.Localization;

using Telerik.WinControls.UI;

using System.Data.OleDb;

using System.Data.SqlClient;

namespace schedule

{

// TODO: добавить таблицу с настройками + предворительные настройки в самом начале!

public partial class Schedule : Telerik.WinControls.UI.RadForm

{

PanelBusy busy;

PanelJoin panelJoin;

PanelTwoDownView twoDownView;

PanelFragmentSummary fragmentSummary;

RadButton buttonSave;

GridViewComboBoxColumn supplierColumn;

public Schedule()

{

InitializeComponent();

RadGridLocalizationProvider.CurrentProvider = new MyEnglishRadGridLocalizationProvider();

//Загрузка данных таблиц

this.cabinetTableAdapter.Fill(this.scheduleDataSet.cabinet);

this.disciplineTableAdapter.Fill(this.scheduleDataSet.discipline);

this.employeeTableAdapter.Fill(this.scheduleDataSet.employee);

this.specialtyTableAdapter.Fill(this.scheduleDataSet.specialty);

this.time\_work\_teachersTableAdapter.Fill(this.scheduleDataSet.time\_work\_teachers);

this.courseTableAdapter.Fill(this.scheduleDataSet.course);

this.teacherTableAdapter.Fill(this.scheduleDataSet.teacher);

this.discipline\_cabinetTableAdapter.Fill(this.scheduleDataSet.discipline\_cabinet);

}

public int countPanel = 0;

//Открытие панели с таблицей КАБИНЕТЫ

private void button1\_Click(object sender, EventArgs e) {

PanelTable panelCabinet = new PanelTable(this, cabinetBindingSource, button1);

panelCabinet.Dock = DockStyle.Left;

countPanel++;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(panelCabinet);

panelCabinet.header.Text = "Кабинеты";

button1.Enabled = false;

panelCabinet.radGridView1.DataSource = cabinetBindingSource;

panelCabinet.radGridView1.Columns[0].IsVisible = false;

panelCabinet.radGridView1.Columns[1].HeaderText = "Номер кабинета";

panelCabinet.radGridView1.Columns[1].Width = 420;

}

private void button2\_Click(object sender, EventArgs e) {

PanelTable panelDiscipline = new PanelTable(this, disciplineBindingSource, button2);

panelDiscipline.Dock = DockStyle.Left;

countPanel++;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(panelDiscipline);

panelDiscipline.header.Text = "Дисциплины";

button2.Enabled = false;

panelDiscipline.radGridView1.DataSource = disciplineBindingSource;

panelDiscipline.radGridView1.Columns[0].IsVisible = false;

panelDiscipline.radGridView1.Columns[1].HeaderText = "Наименование дисциплины";

panelDiscipline.radGridView1.Columns[1].Width = 420;

}

private void buttonTeachers\_Click(object sender, EventArgs e) {

PanelTable panelEmployee = new PanelTable(this, employeeBindingSource, buttonTeachers);

panelEmployee.Dock = DockStyle.Left;

countPanel++;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(panelEmployee);

panelEmployee.header.Text = "Преподователи";

buttonTeachers.Enabled = false;

panelEmployee.radGridView1.DataSource = employeeBindingSource;

panelEmployee.radGridView1.Columns[0].IsVisible = false;

panelEmployee.radGridView1.Columns[1].HeaderText = "ФИО преподавателя";

panelEmployee.radGridView1.Columns[1].Width = 420;

}

private void buttonSpecialty\_Click(object sender, EventArgs e) {

PanelTable panelSpecialty = new PanelTable(this, specialtyBindingSource, buttonSpecialty);

panelSpecialty.Dock = DockStyle.Left;

countPanel++;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(panelSpecialty);

panelSpecialty.header.Text = "Специальность";

buttonSpecialty.Enabled = false;

panelSpecialty.radGridView1.DataSource = specialtyBindingSource;

panelSpecialty.radGridView1.Columns[0].IsVisible = false;

panelSpecialty.radGridView1.Columns[1].HeaderText = "Наименование специальности";

panelSpecialty.radGridView1.Columns[1].Width = 420;

}

private void buttonEmployee\_Click(object sender, EventArgs e) {

busy = new PanelBusy(this, employeeBindingSource, buttonEmployee);

busy.Dock = DockStyle.Left;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(busy);

buttonEmployee.Enabled = false;

busy.radGridView1.DataSource = employeeBindingSource;

busy.radGridView1.Columns[0].IsVisible = false;

busy.radGridView1.Columns[1].HeaderText = "ФИО Преподавателя";

busy.radGridView1.Columns[1].Width = 410;

}

private void groupButton\_Click(object sender, EventArgs e) {

panelJoin = new PanelJoin(this, specialtyBindingSource, groupButton);

panelJoin.Dock = DockStyle.Left;

groupButton.Enabled = false;

//tableLayoutPanel1.Controls.Add(panelCabinet, 1, 1);

radPanel1.Controls.Add(panelJoin);

panelJoin.header.Text = "Группы специальностей";

panelJoin.radDropDownList1.DataSource = specialtyBindingSource;

panelJoin.radDropDownList1.DisplayMember = "name\_specialty";

panelJoin.radDropDownList1.ValueMember = "name\_specialty";

BindingSource bindingSource = new BindingSource();

bindingSource.DataSource = specialtyBindingSource;

bindingSource.DataMember = "specialtycourse";

panelJoin.radGridView2.DataSource = bindingSource;

panelJoin.radGridView2.Columns[0].IsVisible = false;

panelJoin.radGridView2.Columns[1].HeaderText = "Наименование группы";

panelJoin.radGridView2.Columns[1].Width = 405;

panelJoin.radBindingNavigator1.BindingSource = bindingSource;

}

private void teacherDisceplineButton\_Click(object sender, EventArgs e) {

//Сохранение в таблицу tacher

//Кнопку на СОХРАНИТЬ

//Считывание с таблицы

//

teacherDisceplineButton.Enabled = false;

panelJoin = new PanelJoin(this, specialtyBindingSource, teacherDisceplineButton);

panelJoin.Dock = DockStyle.Left;

radPanel1.Controls.Add(panelJoin);

panelJoin.header.Text = "Преподаватели дисциплин";

panelJoin.radDropDownList1.DataSource = employeeBindingSource;

panelJoin.radDropDownList1.DisplayMember = "FIO";

panelJoin.radDropDownList1.ValueMember = "FIO";

panelJoin.radDropDownList1.SelectedValueChanged += RadDropDownList1\_SelectedValueChanged;

BindingSource bindingSource = new BindingSource();

bindingSource.DataSource = disciplineBindingSource;

panelJoin.radGridView2.DataSource = bindingSource;

GridViewCheckBoxColumn checkBoxColumn = new GridViewCheckBoxColumn();

checkBoxColumn.DataType = typeof(int);

checkBoxColumn.Name = "check";

checkBoxColumn.FieldName = "Да/Нет";

checkBoxColumn.HeaderText = "Да/Нет";

panelJoin.radGridView2.MasterTemplate.Columns.Add(checkBoxColumn);

panelJoin.radGridView2.Columns.Move(2, 0);

panelJoin.radGridView2.Columns[0].Width = 70;

panelJoin.radGridView2.Columns[1].IsVisible = false;

panelJoin.radGridView2.Columns[2].HeaderText = "Наименование дисциплины";

panelJoin.radGridView2.Columns[2].Width = 320;

panelJoin.radGridView2.Columns[2].ReadOnly = true;

panelJoin.radGridView2.AllowAddNewRow = false;

panelJoin.radBindingNavigator1.BindingSource = bindingSource;

panelJoin.radGridView2.Height = 710;

//кнопка сохранить

RadButton buttonSave = new RadButton();

panelJoin.radPanel2.Controls.Add(buttonSave);

buttonSave.Location = new Point(12, 790);

buttonSave.ThemeName = "Material";

buttonSave.Text = "Сохранить";

buttonSave.Click += ButtonSave\_Click;

//инициализация CheckBox PS> magic! without this code don't work

for (int i = 0; i < panelJoin.radGridView2.RowCount; i++) {

panelJoin.radGridView2.Rows[i].Cells[0].Value = 0;

}

//проставляем галочки с таблицы teacher

CheckBoxTableDiscipline();

//конец простановки

}

private void RadDropDownList1\_SelectedValueChanged(object sender, EventArgs e) {

for (int i = 0; i < panelJoin.radGridView2.RowCount; i++) {

panelJoin.radGridView2.Rows[i].Cells[0].Value = 0;

}

CheckBoxTableDiscipline();

}

//TODO: получение ID\_teacher быстрее

private void ButtonSave\_Click(object sender, EventArgs e) {

OleDbConnection sqlConnection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=|DataDirectory|\\schedule.accdb");

sqlConnection.Open();

OleDbCommand cmd = new OleDbCommand();

cmd.Connection = sqlConnection;

cmd.CommandType = CommandType.Text;

//находим ID Преподавателя в таблице employee

string nameTeacher = panelJoin.radDropDownList1.SelectedValue.ToString(); //имя преподавателя выбранное в DropDownList

string idTeacher=""; //переменная для хранение id Преподавателя

string discepline="";

//пробегаемся по всей таблице ища имя, после успешного нахождения, сохраняем ID найденной строки в переменную idTeacher

for (int i = 0; i < scheduleDataSet.Tables["employee"].Rows.Count; i++) {

if (scheduleDataSet.Tables["employee"].Rows[i][1].ToString() == nameTeacher) {

idTeacher = scheduleDataSet.Tables["employee"].Rows[i][0].ToString();

}

}

for (int i = 0; i < panelJoin.radGridView2.RowCount; i++) {

int j = 0 ;

if (panelJoin.radGridView2.Rows[i].Cells[0].Value.ToString() == "1") {

discepline = panelJoin.radGridView2.Rows[i].Cells[1].Value.ToString();

string queryString =

"INSERT INTO teacher (ID\_teacher, ID\_discipline) " +

"VALUES (" + idTeacher + "," + discepline + ");";

//MessageBox.Show(queryString); //Посмотреть запрос....

cmd.CommandText = queryString;

cmd.ExecuteNonQuery();

}

}

sqlConnection.Close();

}

private void cabinetDisciplineButton\_Click(object sender, EventArgs e) {

cabinetDisciplineButton.Enabled = false;

twoDownView = new PanelTwoDownView(this, disciplineBindingSource, cabinetDisciplineButton);

twoDownView.Dock = DockStyle.Left;

twoDownView.Width -= 150;

twoDownView.radPanel1.Width -=150;

twoDownView.closePanel.Location = new Point(twoDownView.closePanel.Location.X - 150, twoDownView.closePanel.Location.Y);

twoDownView.radPanel2.Width -= 150;

twoDownView.radDropDownList1.Width -= 150;

twoDownView.radDropDownList2.Width -= 150;

twoDownView.radGridView2.Width -= 148;

twoDownView.radGridView2.Height -= 110;

//кнопка сохранить

RadButton buttonSave = new RadButton();

twoDownView.radPanel2.Controls.Add(buttonSave);

buttonSave.Location = new Point(12, 790);

buttonSave.ThemeName = "Material";

buttonSave.Text = "Сохранить";

buttonSave.Click += ButtonSave\_Click1;

radPanel1.Controls.Add(twoDownView);

twoDownView.header.Text = "Кабинеты дисциплин";

twoDownView.radDropDownList1.DataSource = disciplineBindingSource;

twoDownView.radDropDownList1.DisplayMember = "name\_discipline";

twoDownView.radDropDownList1.ValueMember = "name\_discipline";

twoDownView.radGridView2.DataSource = cabinetBindingSource;

GridViewRatingColumn column = new GridViewRatingColumn("Rating Column");

twoDownView.radGridView2.Columns.Add(column);

column.Name = "rating";

column.FieldName = "Рейтинг";

column.HeaderText = "Рейтинг";

column.SelectionMode = RatingSelectionMode.FullItem;

column.Minimum = 0;

column.Maximum = 5;

twoDownView.radGridView2.Columns.Move(1, 0);

twoDownView.radGridView2.Columns[0].Width = 265;

twoDownView.radGridView2.Columns[0].HeaderText = "Название кабинета";

twoDownView.radGridView2.Columns[1].IsVisible = false;

twoDownView.radGridView2.Columns[2].HeaderText = "Рейтинг";

twoDownView.radGridView2.Columns[2].Width = 145;

twoDownView.radGridView2.Columns[2].ReadOnly = true;

twoDownView.radGridView2.AllowAddNewRow = false;

twoDownView.radBindingNavigator1.BindingSource = cabinetBindingSource;

CheckTableDisciplineCabinet();

//получение значений

twoDownView.radDropDownList1.SelectedValueChanged += RadDropDownList1\_SelectedValueChanged1;

twoDownView.radDropDownList2.SelectedIndexChanged += RadDropDownList2\_SelectedValueChanged;

}

private void RadDropDownList2\_SelectedValueChanged(object sender, EventArgs e) {

CheckTableDisciplineCabinet();

}

private void RadDropDownList1\_SelectedValueChanged1(object sender, EventArgs e) {

CheckTableDisciplineCabinet();

}

private void ButtonSave\_Click1(object sender, EventArgs e) {

OleDbConnection sqlConnection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=|DataDirectory|\\schedule.accdb");

sqlConnection.Open();

OleDbCommand cmd = new OleDbCommand();

cmd.Connection = sqlConnection;

cmd.CommandType = CommandType.Text;

string nameDiscepline = twoDownView.radDropDownList1.SelectedValue.ToString();

string idDiscepline = "";

string course = "";

string numCourse = "";

for (int i = 0; i < scheduleDataSet.Tables["discipline"].Rows.Count; i++) {

if (scheduleDataSet.Tables["discipline"].Rows[i][1].ToString() == nameDiscepline) {

idDiscepline = scheduleDataSet.Tables["discipline"].Rows[i][0].ToString();

}

}

try {

numCourse = twoDownView.radDropDownList2.SelectedItem.Text;

}

catch (Exception) {

numCourse = "1 Курс";

}

switch (numCourse) {

case "1 Курс":

course = "1";

break;

case "2 Курс":

course = "2";

break;

case "3 Курс":

course = "3";

break;

case "4 Курс":

course = "4";

break;

}

for (int i = 0; i < twoDownView.radGridView2.RowCount; i++) {

string idCabinet = idCabinet = twoDownView.radGridView2.Rows[i].Cells[1].Value.ToString();

string rating = twoDownView.radGridView2.Rows[i].Cells[2].Value.ToString();

string queryString =

"INSERT INTO discipline\_cabinet (ID\_discipline, ID\_cabinet, course, rating) " +

"VALUES (" + idDiscepline + "," + idCabinet + "," + course + "," + rating + ");";

cmd.CommandText = queryString;

cmd.ExecuteNonQuery();

}

sqlConnection.Close();

}

private void specialtyCourseButton\_Click(object sender, EventArgs e) {

twoDownView = new PanelTwoDownView(this, specialtyBindingSource, specialtyCourseButton);

twoDownView.Dock = DockStyle.Left;

radPanel1.Controls.Add(twoDownView);

twoDownView.header.Text = "Преподаватели дисциплин на курсе";

specialtyCourseButton.Enabled = false;

twoDownView.radDropDownList1.DataSource = specialtyBindingSource;

twoDownView.radDropDownList1.DisplayMember = "name\_specialty";

twoDownView.radDropDownList1.ValueMember = "name\_specialty";

supplierColumn = new GridViewComboBoxColumn {

Name = "supplierColumn",

HeaderText = "ФИО Преподавателя",

DataSource = this.employeeBindingSource,

ValueMember = "FIO",

DisplayMember = "FIO",

FieldName = "ФИО Преподавателя",

Width = 300

};

GridViewComboBoxColumn disceplineColumn = new GridViewComboBoxColumn {

Name = "disceplineColumn",

HeaderText = "Дисциплина",

DataSource = this.disciplineBindingSource,

ValueMember = "name\_discipline",

DisplayMember = "name\_discipline",

FieldName = "Наименование дисциплины",

Width = 250

};

twoDownView.radGridView2.Columns.Add(disceplineColumn);

twoDownView.radGridView2.Columns.Add(supplierColumn);

twoDownView.radGridView2.Height -= 110;

buttonSave = new RadButton();

twoDownView.radPanel2.Controls.Add(buttonSave);

buttonSave.Location = new Point(497, 790);

buttonSave.ThemeName = "Material";

buttonSave.Text = "Добавить";

buttonSave.Click += ButtonSave\_Click2;

//twoDownView.radGridView2.Columns[0].PropertyChanged += MainForm\_PropertyChanged;

//twoDownView.radGridView1.AutoGenerateColumns = false;

//twoDownView.radGridView1.DataSource = this.employeeTableAdapter.GetDataBy(twoDownView.radGridView2.CurrentCell.Value.ToString());

//supplierColumn.DataSource = Table;

//supplierColumn.DisplayMember = Table.Tables[0].ToString();

//twoDownView.radGridView2.Columns[0].IsVisible = false;

//twoDownView.radGridView2.Columns[1].Width = 250;

//twoDownView.radGridView2.Columns[1].HeaderText = "Наименование дисциплины";

//twoDownView.radBindingNavigator1.BindingSource = bindingSource;

}

private void ButtonSave\_Click2(object sender, EventArgs e) {

twoDownView.Width += 500;

twoDownView.radPanel1.Width += 480;

twoDownView.radPanel2.Width += 480;

addTeachers addTeachers = new addTeachers();

twoDownView.radPanel2.Controls.Add(addTeachers);

addTeachers.Location = new Point(625, 12);

twoDownView.closePanel.Location = new Point(twoDownView.closePanel.Location.X + 480, 12);

GridViewCheckBoxColumn checkBoxColumn = new GridViewCheckBoxColumn();

checkBoxColumn.DataType = typeof(int);

checkBoxColumn.Name = "check";

checkBoxColumn.FieldName = "Да/Нет";

checkBoxColumn.HeaderText = "Да/Нет";

GridViewTextBoxColumn gridViewTextBoxColumn = new GridViewTextBoxColumn();

gridViewTextBoxColumn.Name = "dis";

gridViewTextBoxColumn.HeaderText = "Наименование дисциплины";

addTeachers.radGridView2.MasterTemplate.Columns.Add(checkBoxColumn);

addTeachers.radGridView2.MasterTemplate.Columns.Add(gridViewTextBoxColumn);

//addTeachers.radGridView2.Columns.Move(1, 0);

addTeachers.radGridView2.Columns[0].Width = 70;

//addTeachers.radGridView2.Columns[1].IsVisible = false;

addTeachers.radGridView2.Columns[1].HeaderText = "Наименование дисциплины";

addTeachers.radGridView2.Columns[1].Width = 320;

//addTeachers.radGridView2.Columns[1].ReadOnly = true;

addTeachers.radDropDownList1.DataSource = employeeBindingSource;

addTeachers.radDropDownList1.DisplayMember = "FIO";

addTeachers.radDropDownList1.ValueMember = "FIO";

buttonSave.Text = "Закрыть";

}

private void MainForm\_PropertyChanged(object sender, PropertyChangedEventArgs e) {

//ChangeComboBox();

try {

supplierColumn.DataSource = this.employeeTableAdapter.GetDataBy(twoDownView.radGridView2.CurrentCell.Value.ToString());

}

catch (Exception) {

}

}

private void fragmentSummaryButtn\_Click(object sender, EventArgs e) {

fragmentSummary = new PanelFragmentSummary(this, specialtyBindingSource, fragmentSummaryButton);

fragmentSummary.Dock = DockStyle.Left;

radPanel1.Controls.Add(fragmentSummary);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MyFunc\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

private void CheckBoxTableDiscipline() {

//проставляем галочки с таблицы teacher

string idTeacher = "";

string dis = "";

string nameTeacher = panelJoin.radDropDownList1.SelectedValue.ToString();

for (int i = 0; i < scheduleDataSet.Tables["employee"].Rows.Count; i++) {

if (scheduleDataSet.Tables["employee"].Rows[i][1].ToString() == nameTeacher) {

idTeacher = scheduleDataSet.Tables["employee"].Rows[i][0].ToString();

}

}

for (int i = 0; i < scheduleDataSet.Tables["teacher"].Rows.Count; i++) {

if (scheduleDataSet.Tables["teacher"].Rows[i][0].ToString() == idTeacher) {

dis = scheduleDataSet.Tables["teacher"].Rows[i][1].ToString();

for (int j = 0; j < panelJoin.radGridView2.RowCount; j++) {

if (panelJoin.radGridView2.Rows[j].Cells[1].Value.ToString() == dis) {

panelJoin.radGridView2.Rows[j].Cells[0].Value = 1;

}

}

}

}

}

//проставляем звёздочки с таблицы discipline\_cabinet

private void CheckTableDisciplineCabinet() {

string idDiscipline = "",

idCabinet = "",

course = "",

numCourse = "",

rating = "";

for (int i = 0; i < twoDownView.radGridView2.RowCount; i++) {

twoDownView.radGridView2.Rows[i].Cells[2].Value = 0;

}

for (int i = 0; i < scheduleDataSet.Tables["discipline"].Rows.Count; i++) {

if (scheduleDataSet.Tables["discipline"].Rows[i][1].ToString() == twoDownView.radDropDownList1.SelectedValue.ToString()) {

idDiscipline = scheduleDataSet.Tables["discipline"].Rows[i][0].ToString();

}

}

try {

numCourse = twoDownView.radDropDownList2.SelectedItem.Text;

}

catch (Exception) {

numCourse = "1 Курс";

}

switch (numCourse) {

case "1 Курс":

course = "1";

break;

case "2 Курс":

course = "2";

break;

case "3 Курс":

course = "3";

break;

case "4 Курс":

course = "4";

break;

}

for (int i = 0; i < scheduleDataSet.Tables["discipline\_cabinet"].Rows.Count; i++) {

if (scheduleDataSet.Tables["discipline\_cabinet"].Rows[i][0].ToString() == idDiscipline && scheduleDataSet.Tables["discipline\_cabinet"].Rows[i][2].ToString() == course) {

string cab = twoDownView.radGridView2.Rows[i].Cells[1].Value.ToString();

if (scheduleDataSet.Tables["discipline\_cabinet"].Rows[i][1].ToString() == cab) {

twoDownView.radGridView2.Rows[i].Cells[2].Value = scheduleDataSet.Tables["discipline\_cabinet"].Rows[i][3].ToString();

}

}

}

}

//При выборе одного выпадающего списка изменяется другой

private string[] ChangeComboBox() {

OleDbConnection sqlConnection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=|DataDirectory|\\schedule.accdb");

sqlConnection.Open();

OleDbCommand cmd = new OleDbCommand {

Connection = sqlConnection,

CommandType = CommandType.Text

};

string queryString =

"SELECT employee.FIO " +

"FROM teacher, discipline, employee " +

"WHERE teacher.ID\_discipline = discipline.ID\_discipline AND " +

"teacher.ID\_teacher = employee.ID\_teacher AND(discipline.name\_discipline = '"+ twoDownView.radGridView2.CurrentCell.Value.ToString() +"')";

cmd.CommandText = queryString;

OleDbDataAdapter sqlDataAdapter = new OleDbDataAdapter(cmd);

DataSet dataTable = new DataSet();

sqlDataAdapter.Fill(dataTable);

//TODO: странное получение списка из таблицы

string[] Table = new string[dataTable.Tables[0].Rows.Count];

for (int i = 0; i < dataTable.Tables[0].Rows.Count; i++) {

Table[i] = dataTable.Tables[0].Rows[i][0].ToString();

}

sqlConnection.Close();

return Table;

}

private void radMenuItem18\_Click(object sender, EventArgs e) {

this.Close();

}

}

}