СОДЕРЖАНИЕ

Введение 4

1 Технический проект 7

1.1 Анализ предметной области 7

1.2 Постановка задачи 8

1.3 Проектирование базы данных 10

1.4 Проектирование функциональности программы 13

1.5 Проектирование структуры программы 14

1.6 Выбор средств реализации 18

1.7 Требования к программному обеспечению и техническим средствам 21

2 Рабочий проект 24

2.1 Физическая модель данных 24

2.2 Функциональное взаимодействие модулей программы 27

2.3 Входные и выходные данные 36

2.4 Установка и настройка программы 38

2.5 Работа с программой 39

Заключение 47

Список использованных источников 49

Приложение А – Use Case диаграмма 50

Приложение Б – Код основных классов 51

ВВЕДЕНИЕ

Для выполнения своих задач сотрудникам отделения информационных технологий МГГТК ФГБОУ ВО «АГУ» необходимо оперировать большими объемами информации, формировать различные отчеты. Поэтому возникла необходимость автоматизировать некоторые процессы с использованием современных подходов и средств реализации информационных систем.

Любая информационная система включает в себя одну или несколько баз данных (БД), что значительно упрощает процессы хранения и обработки информации.

Одной из основных задач отделения является накопление и обработка сведений об обучающихся, их учебной деятельности. Именно данный процесс будет автоматизироваться в рамках данного дипломного проекта.

Темой дипломного проекта является разработка и создание электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ».

Объектом исследования является деятельность отделения информационных технологий по ведению информации об обучающихся.

Предметом исследования являются карточки обучающихся отделения информационных технологий, а также средства реализации информационной системы для создания электронных карточек обучающихся.

Целью данного дипломного проекта является разработка расширяемой информационной системы, которая позволит просматривать и редактировать информацию об обучающихся на отделении.

Написание данного дипломного проекта является целесообразным, потому что информационная система:

* позволит просматривать и добавлять необходимую информацию об обучающихся отделения;
* облегчит работу секретаря отделения за счёт возможности автоматического формирования различных отчётов;
* позволит контролировать успеваемость обучающихся по результатам промежуточной и итоговой аттестации;
* снизит объем бумажной документации.

Разработка нового приложения позволит учесть индивидуальные предпочтения пользователя в интерфейсе и предоставляемых функциях, не перегружая информационную систему лишним, реализованным «впрок», инструментарием. Это снизит системные требования к развертыванию данной информационной системы.

Для достижения поставленной цели будут рассмотрены и решены следующие задачи:

* произведен анализ предметной области и выделены необходимые данные;
* разработаны концептуальная и логическая модели БД;
* разработаны структура и функциональность приложения для управления созданной БД;
* произведён выбор средств реализации;
* описаны минимальные требования к программно-техническим средствам;
* произведена физическая реализация БД;
* описаны входные и выходные данные;
* разработан программный код приложения;
* разработана пользовательская система помощи;
* разработана инсталляция.

Практическая значимость проекта заключается в возможности использования приложения в образовательных учреждения с целью автоматизации процесса ведения электронных карточек обучающихся конкретного отделения.

Основными пользователями данной программы будут сотрудники отделения, ответственные за ведение отчётности по учебному процессу.

Данные для приложения будут взяты на основе анализа документации отделения.

Теоретическими основами и методами решения поставленных задач будут методологии проектирования функциональности приложения, структуры БД, особенности реализации программного кода с помощью конкретного языка программирования, принципы и алгоритмы применения инструментальных средств на различных этапах разработки программного продукта.

В техническом проекте будет произведен анализ предметной области и представлены этапы проектирования архитектуры, интерфейса и функциональности реализуемой информационной системы, включая разработку логической модели БД. Также в нем будет произведен выбор инструментов для физической реализации построенных моделей данных и информационной системы.

Наиболее удобной для реализации поставленной задачи является нереляционная документоориентированная БД, которая использует JSON-подобные документы и схему БД. В соответствии с этим будут подобраны система управления БД (СУБД) и инструментальные средства реализации интерфейса приложения, которое будет работать с данной БД.

Для проектирования интерфейса и функциональности приложения, а также архитектуры информационной системы будут использоваться UML-диаграммы.

В рабочем проекте представлены достигнутые практические результаты, в частности, описаны физическая модель данных и разработанные классы, реализующие функциональность приложения. Также в нем описаны процессы установки информационной системы и особенности ее эксплуатации.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

Отделение является основным учебно-научным структурным подразделением средних учебных заведений, которое проводит учебную, методическую и научно-исследовательскую деятельность по одной или нескольким родственным специальностям.

На отделение возлагаются следующие задачи и обязанности:

* реализация федеральных государственных образовательных стандартов (ФГОС СПО), целевая ориентация учебного процесса на формирование общих и профессиональных компетенций, установленных ФГОС СПО;
* обеспечение качественного образования, повышение ответственности за результаты образовательной деятельности;
* внедрение в образовательный процесс современных подходов к обучению и воспитанию обучающихся;
* усиление работы по сохранению контингента и адаптации обучающихся.

Реализация данных задач предполагает планирование и управление процессом проведения лекций, лабораторных, практических, семинарских и других видов учебных занятий, руководства учебной и производственной практикой, исследовательскими работами, курсовыми и дипломными проектами (работами), проведения экзаменов и зачётов на промежуточной аттестации, а также итоговой аттестации.

Достижения обучающихся в контрольных точках образовательного процесса (промежуточная и итоговая аттестация) необходимо фиксировать, следить за их изменениями, накапливать и обрабатывать. Также нужно отслеживать движение контингента. Не менее важным является фиксация, хранение и обработка достижений обучающихся в научной и внеурочной деятельности различной направленности.

Сопровождение данной деятельности требует оперирование большими объемами информации. Поэтому существует практическая потребность в автоматизации данных функций для облегчения процесса накопления, хранения и обработки информации, а также получения необходимых отчетов.

Разрабатываемое приложение будет автоматизировать и упрощать следующие процессы:

* формирование списков учебных групп отделения;
* заполнение и оформление документов обучающихся (карточка студента);
* планирование образовательного процесса (заполнение учебных планов групп);
* оформление распорядительной документации по движению контингента;
* организация промежуточной и итоговой аттестации: экзаменов и дифференцированных зачетов по учебным дисциплинам, междисциплинарным курсам, профессиональным модулям, учебной и производственным практикам, государственных экзаменов и защиты выпускной квалификационной работы (формирование общей ведомости успеваемости по каждой группе).

Разрабатываемое приложение будет оперировать данными об реализуемых учебных планах по специальностям, списками учебных групп, результатами успеваемости обучающихся.

1.2 Постановка задачи

Необходимо разработать клиент-серверную информационную систему для учета и хранения электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ», а также создать документационное сопровождение приложения, включающее в себе технический и рабочие проекты.

Необходимые данные должны храниться в удалённой БД.

Данная программа должна будет содержать все нужные сведения о преподавателях и обучающихся отделения, реализуемых учебных планах, иметь удобный и понятный пользователю графический интерфейс, обеспечивать необходимый уровень защиты данных от несанкционированного использования.

Программный продукт должен включать в себя две части:

* информационно-справочная часть, которая будет отображать необходимую информацию;
* операционная часть, в которой преподаватели и сотрудники отделения смогут изменять необходимую информацию.

Приложение должно обеспечивать выполнение следующих функций:

* авторизация;
* отображение личного кабинета пользователя;
* отображение меню;
* просмотр и редактирование информации о сотрудниках, преподавателях, группах, обучающихся группы, учебных планах, успеваемости обучающихся по дисциплинам;
* просмотр и редактирование новостей отделения;
* автоматизация составления отчётов за счет программного формирования аттестационных ведомостей;
* выставление оценки в ведомость в электронном виде;
* взаимодействие с данными, хранящимися в удалённой БД.

Данные будут предоставлены в виде графической и текстовой информации, загруженной из целевой БД.

Входными данными в приложении будут учебный план и информация о студентах и их успеваемости, преподавателях и сотрудниках, которые должны храниться в БД.

Выходными данными в приложении будут информация об успеваемости студентов, отчёты (сводная ведомость по группе), вывод информации в личном кабинете, новости отделения.

Приложение будет состоять из окон, на которых будет располагаться вся необходимая пользователю информация.

Способ решения поставленных задач зависит от выбора средств разработки.

Тестирование программы будет проводиться путем выбора пунктов меню и проверки корректности получаемого отклика программы.

1.3 Проектирование базы данных

Первым шагом на пути реализации проекта будет создание БД.

В широком смысле БД – это хранилище элементов данных, называемых «записями», имеющее определенную физическую и логическую структуру, а также программный интерфейс, позволяющий пользователю взаимодействовать с сохраняемой в ней информацией. Чтобы универсальным способом извлекать из нее группы записей, обрабатывать их, изменять и удалять, требуются специальные программы, которые называются СУБД [2].

Наиболее удобной для реализации поставленной задачи является нереляционная документоориентированная БД, которая использует JSON-подобные документы и схему БД.

В теории документоориентированная БД специально предназначена для хранения иерархических структур данных (документов) и обычно реализуется с помощью подхода NoSQL. В основе документоориентированных СУБД лежат документные хранилища (англ. document store), имеющие структуру дерева (иногда леса). Структура дерева начинается с корневого узла и может содержать несколько внутренних и листовых узлов. Листовые узлы содержат данные, которые при добавлении документа заносятся в индексы, что позволяет даже при достаточно сложной структуре находить место (путь) искомых данных. API для поиска позволяет находить по запросу документы и части документов. В отличие от хранилищ типа ключ-значение, выборка по запросу к документному хранилищу может содержать части большого количества документов без полной загрузки этих документов в оперативную память.

Документы могут быть организованы (сгруппированы) в коллекции. Их можно считать отдалённым аналогом таблиц реляционных СУБД, но коллекции могут содержать другие коллекции. Хотя документы коллекции могут быть произвольными, для более эффективного индексирования лучше объединять в коллекцию документы с похожей структурой.

В данном дипломном проекте предметной областью являются карточки обучающихся отделения информационных технологий. БД будет предназначена для хранения данных об обучающихся и их успеваемости.

Первый этап процесса проектирования БД заключается в создании концептуальной модели данных.

Компонентами концептуальной модели являются сущности и взаимосвязи. Она служит средством общения между различными пользователями, и поэтому разрабатывается без учета особенностей физического представления данных. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели базируется на основе анализа решаемых в предметной области задач по обработке данных [2].

На основе анализа предметной области и пункта 1.2 были выделены основные сущности:

* пользователи (идентификатор, логин, пароль, электронная почта, имя, фамилия, отчество, дата рождения, адрес, изображения профиля, статус активации аккаунта);
* сотрудники (идентификатор, идентификатор пользователя, идентификатор роли, занимаемая должность (должности), список предоставленных разрешений (уровень доступа в системе), статус сотрудника);
* обучающиеся (идентификатор, идентификатор пользователя, идентификатор учебной группы, идентификатор роли, список предоставленных разрешений (уровень доступа в системе), дата приема, идентификатор приказа о зачислении, информация о движении, примечания, личные достижения);
* приказы (идентификатор, номер приказа, наименование приказа);
* группы (идентификатор, наименование группы, идентификатор сотрудника (куратор), идентификатор учебного плана);
* новости (идентификатор, идентификатор пользователя, наименование новости, дата публикации, категория, текст, изображение);
* учебные планы (идентификатор, наименование плана, дата создания, список предметов);
* предметы (идентификатор, наименование, количество академических часов, семестр начала, семестр окончания, тип предмета);
* тип предметов (идентификатор, наименование);
* роли (идентификатор, наименование);
* ведомости (идентификатор, список предметов, оценка, идентификатор студента).

Затем была разработана концептуальная модель БД (рисунок 1) и логическая модель, которая отражает связи между сущностями (рисунок 2).



Рисунок 1 – Концептуальная модель БД



Рисунок 2 – Логическая модель БД

1.4 Проектирование функциональности приложения

Для описания основных функций проектируемой информационной системы воспользуемся методологией UML и построим диаграмму Use Case (приложение А). Она определяет:

* основных пользователей системы, демонстрируя их иерархию;
* основные функции системы и их взаимосвязи;
* группирование функций системы по доступности определенным пользователям;
* зависимость функционирования проектируемой системы от внешних приложений.

На ней видно разграничение уровней доступа к реализованным функциям и основные сценарии использования представленной информационной системы, предусмотренные при её разработке.

Обучающемуся, который является одним из пользователей системы доступны функции просмотра личного кабинета, списка новостей, отчётов по текущем учебным планам, а также и редактирования личного кабинета.

Секретарю, помимо общих пользовательских функций, доступна функция редактирования информации об учебных планах, группах, обучающихся, ведомостей. Кроме того, секретарь может выгрузить ведомости в формате электронной книги Excel.

Администратору системы дополнительно доступна функция редактирования сотрудников.

Преподавателю дополнительно доступна функция редактирования ведомостей.

Вход в личный кабинет подразумевает обязательную авторизацию.

При выполнении всех функций система взаимодействует с удаленной БД, расположенной на сервере. Взаимодействие происходит через стандартизированный интерфейс, предоставляемый СУБД.

1.5 Проектирование структуры приложения

При создании информационной системы одним из самых важных аспектов является её архитектура. Она представляет собой концептуальное видение структуры будущих функциональных процессов и технологий на системном уровне и во взаимосвязи.

Так как разработать нужно клиент-серверную информационную систему, то необходимо определиться с её архитектурой. Основными клиент-серверными архитектурами, реализуемыми в настоящее время являются:

* выделенный сервер;
* активный сервер;
* сервер приложений [5].

В архитектуре «выделенный сервер» средства управления БД и БД размещены на машине-сервере, а функциональная обработка данных выполняется на стороне клиента. Сервер только предоставляет данные по запросу клиента. В данной архитектуре предъявляются повышенные требования к аппаратному и программному обеспечению клиента, так как вся бизнес-логика реализуется на стороне клиента, а сервер выполняет только функции доступа к данным и их защиты.

В архитектуре «активный сервер» функции бизнес-логики разделяются между клиентской и серверной частями. Общие или критически значимые функции оформляются в виде хранимых процедур, включаемых в состав БД. Кроме этого, вводится механизм отслеживания событий БД – триггеров, также включаемых в состав БД. При возникновении соответствующего события (обычно изменения данных) СУБД вызывает для выполнения хранимую процедуру, связанную с триггером, что позволяет эффективно контролировать изменение БД. Хранимые процедуры и триггеры могут быть использованы любыми клиентскими приложениями, работающими с БД. Это снижает дублирование программных кодов и исключает необходимость компиляции каждого запроса.

В данной архитектуре снижаются требования к аппаратному и программному обеспечению клиента, но возрастают, применительно к серверу.

В архитектуре «сервер приложений» снижение уровня требований к ресурсам клиента достигается за счет введения промежуточного звена – сервера приложений, на который переноситься значительная часть программных компонентов управления данными и большая часть бизнес-логики. При этом серверы БД обеспечивают исключительно функции СУБД по ведению и обслуживанию БД [4].

Так как клиентская часть информационной системы будет устанавливаться на компьютерах, не обладающих повышенными характеристиками производительности процессора, оперативной и постоянной памяти, то выбор был остановлен на архитектуре «активный сервер».

Сервер БД будет осуществлять целый комплекс действий по управлению данными. Основными его обязанностями будут:

* поддержка ссылочной целостности данных согласно определенным в БД правилам;
* хранение и резервное копирование данных;
* выполнение пользовательских запросов на выбор и модификацию данных и метаданных, получаемых от клиентских приложений, функционирующих на персональных компьютерах локальной сети;
* обеспечение авторизованного доступа к данным на основе проверки прав и привилегий пользователей;
* протоколирование операций и ведение журнала транзакций.

В качестве рабочих мест пользователя будут использоваться персональные компьютеры секретаря, заведующего отделением и преподавателей, расположенные в локальной сети отделения, что позволяет не отказываться от привычной рабочей среды.

С целью удовлетворения предъявляемых к клиентскому приложению требований, приведенных в разделе 1.2, и представленных на Use Case диаграмме функций, разрабатываемое приложение будет состоять из следующих окон:

* авторизации – будет запускаться перед отображением главного окна для идентификации пользователя;
* главное – будет содержать навигационные кнопки для перехода на страницы просмотра аккаунта, настроек, новостей, информации об учебных планах, группах, приказах и обучающихся;
* ведомости – окно формирования сводной ведомости по выбранному учебному плану и учебной группе;
* добавления и редактирования – окна для добавления и редактирования выводимой пользователю информации (учебные планы, группы, обучающиеся, приказы, новости);
* подробного просмотра – окно для просмотра подробной информации о выбранном пользователем элементе (учебный план, группы, обучающиеся, приказы, новости).

Необходимые данные будут хранится в БД.

Проектируемая структура информационной системы представлена на рисунке 3.

Изображение выглядит как текст, снимок экрана, визитка

Автоматически созданное описание

Рисунок 3 – Схема проектируемой структуры информационной системы

Проектируемая структура клиентского приложения представлена на рисунке 4.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Схема проектируемой структуры клиентского приложения

Тестирование программы планируется проводить путём выбора каждого пункта меню и проверки корректности получаемого отклика программы.

1.6 Выбор средств реализации

Интерфейс и функциональность клиентского приложения могут быть написаны на следующих языках программирования: С++, С#, Python, Java. Все они позволяют работать с БД, файлами, текстом и графикой, с использованием объектно-ориентированного подхода.

Выбор был остановлен на языке программирования C#, с использованием .NET 5, и интегрированной среде разработки Microsoft Visual Studio 2019.

Visual Studio обеспечивает поддержку новейших языковых функций на протяжении всего процесса разработки. Данная среда разработки поддерживает C#, Visual Basic, C++, TypeScript, F# и даже сторонние языки, например, JavaScript [3].

Язык С# актуален потому, что:

* позволяет более рационально создавать популярные на сегодня типы приложений;
* интегрировал в себе преимущества языков Java и С++, при этом в объединённом языке исключены некоторые спорные директивы, макросы, отменены глобальные переменные;
* является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* является мощным объектным языком с возможностями наследования и универсализации;
* мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из БД и других хранилищ данных [4].

.NET 5 – следующий шаг в .NET Core. Проект призван улучшить .NET в нескольких ключевых аспектах:

* расширить возможности .NET за счёт лучших наработок из .NET Core, .NET Framework, Xamarin и Mono;
* создать единые исполняющую среду и фреймворк, которые можно использовать везде, с одинаковым поведением в runtime и опытом разработки;
* собрать продукт из единой кодовой базы, над которой разработчики (из Microsoft и сообщества) могут вместе работать и расширять её, что позволит улучшить все возможные сценарии.

Этот новый проект и направление полностью изменят ситуацию с .NET. Благодаря .NET 5 код и файлы проектов будут выглядеть единообразно, вне зависимости от типа создаваемого приложения. Из каждого приложения будет доступ к той же исполняющей среде, тем же API и возможностям языка, включая новые улучшения производительности, которые внедряются в corefx практически ежедневно.

Сохранилось всё, что было улучшено в .NET Core:

* open source и ориентированность на сообщество GitHub;
* кроссплатформенная реализация;
* поддержка использования специфических платформозависимых возможностей, таких как Windows Forms и WPF под Windows, а также нативных привязок (bindings) к каждой нативной платформе из Xamarin;
* высокая производительность;
* side-by-side инсталляция;
* маленький размер файлов проектов (SDK-стиль);
* интерфейс командной строки (CLI) с широкими возможностями;
* интеграция с Visual Studio, Visual Studio for Mac и Visual Studio Code.

Нововведения:

* возможность вызова кода Java из .NET 5 будет доступна на всех платформах;
* вызов кода Objective-C и Swift из .NET 5 будет поддерживаться в нескольких операционных системах;
* CoreFX будет расширен, чтобы поддерживать статическую компиляцию .NET (ahead-of-time – AOT), для уменьшения потребления ресурсов (footprints) и поддержки большего количества операционных систем [7].

Для проектирования физической модели БД будет использоваться СУБД MongoDB и утилита MongoDBCompass. Данная СУБД является высокоэффективной и надежной, в ней есть возможности для использования ее в любых критичных бизнес-приложениях.

Основные преимущества MongoDB:

* документоориентированное хранение данных – они хранятся в виде документов в формате JSON;
* индекс для любого атрибута;
* репликация и высокая доступность;
* автоматический шардинг;
* полнофункциональные запросы;
* быстрые обновления;
* профессиональная поддержка, предоставляемая MongoDB [2].

Для создания инсталляции была выбрана Inno Setup – система создания инсталляторов для Windows-программ с открытым исходным кодом. Она представляет собой среду для настройки проекта, создания сценария и конечного релиза дистрибутива.

Ключевыми особенностями Inno Setup являются:

* поддержка всех современных версий Windows: 10, 8, 7, 2008 R2, Vista. Поддерживает установку 64-битных программ на 64-битных выпусках. Также поддерживает 64-разрядные процессоры;
* поддержка создания одиночных EXE для облегчения установки и распространения программ через сеть Интернет;
* стандартный интерфейс мастера установки в стиле Windows 2000/XP;
* возможность выбора типа установки, например: полный, минимальный, выборочный;
* имеет встроенную поддержку DEFLATE, bzip2 и 7-Zip LZMA/LZMA2 сжатий;
* установщик умеет сравнивать версии файлов, заменять встроенные файлы, устанавливать разделяемые файлы, регистрировать DLL/OCX библиотеки и устанавливать шрифты;
* позволяет создавать ярлыки в меню «Пуск» и на «Рабочем столе»;
* позволяет создавать записи в реестре и .ini-файлы;
* поддерживает создание многоязычных инсталляторов программ;
* поддерживает Unicode и языки с направлением письма справа налево;
* поддерживает установку пароля и шифрование инсталляторов программ [10].

1.7 Требования к программному обеспечению и техническим средствам

Программное обеспечение должно удовлетворять всем реализуемым функциям системы, а также иметь определенный набор средств для организации всех требуемых процессов обработки данных, которые позволяют своевременно выполнять все функции во всех режимах функционирования системы.

Системное программное обеспечение должно реализовывать комплекс задач управления. Для корректной работы разрабатываемой программы необходимо, чтобы клиентский компьютер, с целью удовлетворения минимальным требованиям, был оснащен:

* операционной системой Windows 10 / 11;
* лицензионным программным обеспечением Microsoft Office 2019 +;
* .NET 5 +;

Сервер должен быть оснащён:

* операционной системой Windows Server 2019 / Ubuntu OS / CentOS Server / Oracle Linux Server;
* MongoDB 5.0.8 +.

При выборе технических средств, применяемых для функционирования разрабатываемой программы, должны учитываться следующие требования:

* выбор технических средств должен обеспечивать рациональное соотношение между затратами на создание системы и достигаемым эффектом;
* технические параметры системы управления не должны налагать ограничения на регламент технологического процесса функционирования системы.

Для реализации вышеуказанных требований со стороны клиентского оборудования необходим следующий состав технических средств:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 1 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 2 ГБ;
* дисплей с разрешением не менее 1024x768 точек;
* 130 МБ свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

Серверное оборудование должно соответствовать следующим аппаратным требованиям:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 4 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 8 ГБ;
* 150 ГБ свободного места на диске.

Для разработки, отладки и расширения реализуемой информационной системы необходим следующий набор технических средств:

* операционная система Windows 10 / 11;
* лицензионное программное обеспечение Microsoft Office 2019 +;
* .NET 5+, включая компоненты для разработки универсальных приложения (Universal Windows Platform);
* компоненты разработчика COM Office Interop;
* Visual Studio 2019 +;
* MongoDB 5.0.8 +;
* MongoDBCompass.

2 РАБОЧИЙ ПРОЕКТ

* 1. Физическая модель данных

Физическая модель данных описывает данные средствами конкретной СУБД. Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом решения, принятые на уровне логического моделирования, определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно также, в пределах этих границ можно принимать различные решения. Например, отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы или коллекции, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным.

Выбор документо-ориентированной СУБД основан на том, что в отличие от реляционных аналогов здесь не накладываются ограничения на набор полей у документа.

На основе логической модели, представленной в пункте 1.3, была создана физическая модель БД. Она описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей (таблицы 1-10).

Таблица 1 – Физическая модель документа «Сотрудники»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| UserId | Идентификатор пользователя | ObjectId |
| RoleId | Идентификатор роли | ObjectId |
| Post | Занимаемая должность | Object[] |
| Permissions | Список разрешений | Object[] |
| Status | Статус сотрудника | string |

Таблица 2 – Физическая модель документа «Пользователи»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Login | Логин | String |
| Password | Пароль | String |
| Email | Электронная почта | String |
| FirstName | Имя | String |
| Surname | Фамилия | String |
| MiddleName | Отчество | String |
| BirthDate | Дата рождения | Date |
| Address | Адрес | String |
| Image | Изображение профиля | Byte[] |
| IsActive | Статус активации аккаунта | Bit |

Таблица 3 – Физическая модель документа «Обучающиеся»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| UserId | Идентификатор пользователя | ObjectId |
| RoleId | Идентификатор роли | ObjectId |
| GroupId | Идентификатор учебной группы | ObjectId |
| Permissions | Список разрешений | Object[] |
| DateOfEntry | Дата поступления | Date |
| OrderId | Идентификатор приказа о зачислении | ObjectId |
| Movement | Информация о движении | Object[] |
| Achievements | Личные достижения | Object[] |

Таблица 4 – Физическая модель документа «Роли»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Name | Наименование роли | String |

Таблица 5 – Физическая модель документа «Приказы»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Number | Номер приказа | String |
| Name | Наименование приказа | String |

Таблица 6 – Физическая модель документа «Группы»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Name | Наименование группы | String |
| EmployeeId | Идентификатор сотрудника (куратора) | ObjectId |
| EducationalId | Идентификатор учебного плана | ObjectId |

Таблица 7 – Физическая модель документа «Новости»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| EmployeeId | Идентификатор автора | ObjectId |
| Title | Наименование новости | String |
| DateOfPublish | Дата публикации | Date |
| Category | Категория | String |
| Content | Текст новости | String[] |
| Image | Изображение новости | Byte[] |

Таблица 8 – Физическая модель документа «Учебные планы»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Name | Наименование учебного плана | String |
| DateOfCreate | Дата создания | Date |
| Subjects | Список предметов | Object[] |

Таблица 9 – Физическая модель документа «Предметы»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Name | Наименование предмета | String |
| Hours | Количество академических часов | Integer |
| SemestrStart | Семестр начала | Integer |
| SemestrEnd | Семестр окончания | Integer |
| Type | Тип предмета | Object[] |

Таблица 10 – Физическая модель документа «Ведомости»

|  |  |  |
| --- | --- | --- |
| Имя поля | Подпись поля | Тип данных |
| Id | Идентификатор | ObjectId |
| Subjects | Список предметов | Object[] |
| Mark | Оценка | Integer |
| StudentId | Идентификатор студента | ObjectId |

2.2 Функциональное взаимодействие модулей программы

На основе разработанного технического проекта было написано приложение, обеспечивающее ведение электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ». Для создания программы был использован язык программирования C# и интегрированная среда разработки Visual Studio 2019. В программе используются следующие языковые модули:

* System – содержит фундаментальные и базовые классы;
* System.Media – содержит классы для воспроизведения звуковых файлов и работы с системными звуками;
* System.Drawing – предоставляет доступ к основным графическим функциям GDI+ (Graphics Device Interface);
* Microsoft.Win32 – предоставляет два типа классов: те, которые обрабатывают события, вызванные операционной системой, и те, которые управляют системным реестром;
* MongoDB.Bson – содержит фундаментальны классы для работы с MongoDB;
* MongoDB.Driver.GridFS – содержит классы для обеспечения загрузки объемных файлов;
* System.Collections.Generic – содержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие повышенную производительность и безопасность типов по сравнению с неуниверсальными строго типизированными коллекциями;
* System.IO – содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок;
* System.Linq – предоставляет классы и интерфейсы, поддерживающие запросы с использованием LINQ;
* System.Text – cодержит классы, которые представляют кодировки ASCII и Юникода; абстрактные базовые классы для преобразования блоков знаков в блоки байтов и обратно; вспомогательный класс, который обрабатывает и форматирует объекты String, не создавая промежуточные экземпляры String;
* System.Threading.Tasks – предоставляет типы, которые упрощают работу по написанию параллельного и асинхронного кода;
* System.Windows – предоставляет несколько важных классов базовых элементов Windows Presentation Foundation (WPF), различные классы, которые поддерживают систему свойств и логику событий WPF, а также другие типы, более широко применяемые в ядре и инфраструктуре WPF;
* System.Windows.Input – предоставляет типы для поддержки системы ввода Windows Presentation Foundation (WPF). Сюда входят классы абстрагирования устройств для устройств мыши, клавиатуры и пера, часто используемые классы диспетчера ввода, поддержка для команд и пользовательских команд, а также различные служебные классы [6].

Также были разработаны следующие элементы для взаимодействия пользователя с программой:

* AuthWindow.xaml.сs (окно авторизации) – осуществляет функции авторизации в приложении и переход на главное окно;
* MainWindow.xaml.cs (главное окно программы) – осуществляет переход между страницами приложения, а также обеспечивает вывод руководства по использованию программы;
* AddOrUpdateStudentWindow.xaml.cs (окно добавления и редактирования студента) – обеспечивает интерфейс для изменения информации о конкретном студенте;
* StudentPage.xaml.cs (страница со списком студентов) – обеспечивает интерфейс для вывода списка студентов;
* MainPage.xaml.cs (страница новостей) – обеспечивает отображение новостей отделения;
* CreateOrEditEdu.xaml.cs (окно создания и редактирования учебного плана) – обеспечивает создание и редактирование всей информации об учебном плане;
* EducationalPlansPage.xaml.cs (страница со списком учебных планов) –обеспечивает вывод информации обо всех учебных планах;
* OrdersPage.xaml.cs (страница со списком приказов) – обеспечивает вывод информации о приказах;
* AddOrUpdateOrder.xaml.cs (окно создания и редактирования приказа) – обеспечивает создание и редактирование приказов;
* AddOrEditNews.xaml.cs (окно создания и редактирования новостей) – обеспечивает создание и редактирование новостей;
* EmployeePage.xaml.cs (страница со списком сотрудников) – обеспечивает вывод всей информации о сотрудниках отделения;
* AddOrEditEmployee.xaml.cs (окно создания и редактирования сотрудника) – обеспечивает создание и редактирование информации о сотруднике;
* PersonalPage.xaml.cs (страница личного кабинета) – обеспечивает просмотр и редактирование информации о пользователе, авторизованном в системе;
* SettingsPage.xaml.cs (страница настроек) – обеспечивает настройку параметров приложения.

Для реализации функциональности описанного пользовательского интерфейса были разработаны следующие модули, взаимодействие которых представлено на рисунке 5:

* Core (ядро системы) – модуль содержит классы, которые реализуют ключевую логику взаимодействия приложения и операционной системы (ОС) компьютера;
* Command (команды) – модуль содержит классы, реализующие логику различных «команд», которые пользователь отдаёт приложению на выполнение;
* Services (сервисы) – модуль содержит классы, реализующие логику работы с удалённой БД и логическую реализацию внутренних сервисов приложения (пользовательские сессии);
* Assets (ресурсы) – модуль содержит классы и файлы, которые являются общими ресурсами приложения. Классы модуля описывают логику доступа ко внутренним файлам;
* View (интерфейс) – модуль содержит классы и объекты пользовательского интерфейса, необходимые для его отрисовки и взаимодействия с пользователем;
* ViewModel – модуль, являющейся связующим звеном между интерфейсом и логикой программы.

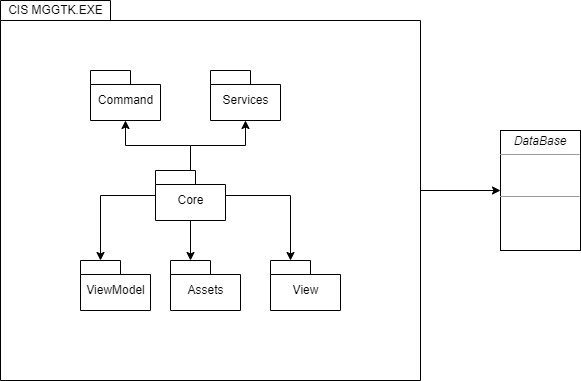


Рисунок 5 – Схема функционального взаимодействия модулей

Для реализации функциональности модуля Command были написаны следующие классы:

* RelayCommand.cs – абстрактный класс, реализующий паттерн команды;
* CommandManager.cs – класс, инкапсулирующий в себе логику создания комманд.

Методы класса RelayCommand.cs:

* public bool CanExecute(object parameter) – вызывается при выполнении команды с предусловием;
* public void Execute(object parameter) – вызывается для немедленного выполнения команды.

Взаимодействие данных классов представлено на рисунке 6.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Диаграмма классов модуля Command

Для реализации функциональности модуля Services были написаны следующие классы:

* FrameManager.cs – класс, реализующий логику переключений страниц в приложении;
* WindowManager.cs – класс, реализующий логику работы с окнами;
* IAuthenticationService.cs – интерфейс, описывающий работу сервиса авторизации;
* AuthenticationService.cs – класс, реализующий логику авторизации;
* SessionStorage.cs – класс, реализующий логику работы пользовательских сессий;
* IBaseCRUDService.cs – интерфейс, описывающий логику базовых операций чтения/записи в удалённой БД;
* BaseCRUDService.cs – класс, реализующий логику базовых операций чтения/записи в удалённой БД;
* UserAuthenticationService.cs – класс, реализующий логику аутентификации пользователя.

Код классов AuthenticationService.cs, UserAuthenticationService.cs, FrameManager.cs и BaseCRUDService.cs представлен в приложении Б.

Метод класса FrameManager.cs: public static void SetSource<T>(T target) where T : Page – вызывается при установке нового источника для отображения данных.

Метод класса WindowManager.cs: public static void CloseWindow(Type window) – вызывается при закрытии модальных окон.

Методы класса AuthenticationService.cs:

* public void InvalidateSession() – вызывается при закрытии текущей сессии пользователя;
* public async Task<bool> ValidateSession(string login, string password, bool isSaved) – вызывается при авторизации пользователя для проверки подлинности данных и сохранения текущей сессии в память.

Методы класса BaseCRUDService.cs:

* public IMongoDatabase GetDatabase() – вызывается для получения экземпляра БД;
* public Task Add(T document) – вызывается при добавлении нового документа в БД;
* public Task Delete(string id) – вызывается при удалении документа из БД;
* public Task<T> Get(string id) – вызывается для получения конкретного документа по уникальному идентификатору;
* public IQueryable<T> GetAll() – вызывается для получения всех документов из коллекции;
* public Task Update(T document) – вызывается для обновления документа в коллекции.

Взаимодействие данных классов представлено на рисунке 7.

Для реализации функциональности модуля Assets был написан класс AssetManager.cs, реализующий логику взаимодействия приложения с ресурсами.

Метод класса AssetManager.cs: public FileInfo GetFileByName(string name) позволяет получить ресурс по его названию.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Диаграмма классов модуля Services

Схема класса AssetManager.cs представлена на рисунке 8.

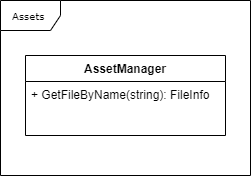


Рисунок 8 – Диаграмма классов модуля Assets

Для реализации функциональности модуля Сore был написан класс Utilities.cs, реализующий логику взаимодействия приложения с ОС.

Методы класса Utilities.cs:

* public void SetTaskPriority(Task task) – вызывается при установке приоритета новой задачи;
* public void CloseApp() – корректно завершает работу приложения, сохраняя все данные в БД;
* public void WriteToDrive(byte[] data, string path) – сохраняет бинарные данные на жесткий диск для временного хранения (при загрузке изображения на сервер).

Схема класса представлена на рисунке 9.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Диаграмма классов модуля Core

Для реализации функциональности модуля ViewModel был написан класс BaseVm.cs, реализующий логику взаимодействия приложения с интерфейсом.

Методы класса BaseVm.cs:

* protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null) – вызывается при установке нового значения у элемента интерфейса для его обновления;
* protected virtual bool SetPropertyChanged<T>(ref T source, T value, [CallerMemberName] string propertyName = null) – устанавливает переданное значение свойству и одновременно обновляет интерфейс.

Схема класса представлена на рисунке 10.

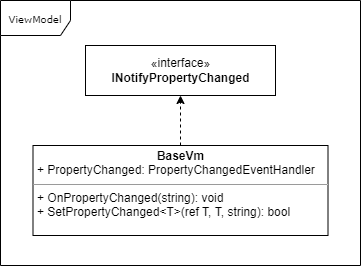


Рисунок 10 – Диаграмма классов модуля ViewModel

Для реализации функциональности модуля View был написан класс ViewManager.cs, реализующий логику получения текущего отображаемого компонента (окна или страницы).

Метод класса WindowManager.cs: public T GetCurrentView() – позволяет получить текущий отображаемый пользователю элемент (страница или окно).

Схема класса представлена на рисунке 11.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 11 – Диаграмма классов модуля View

2.3 Входные и выходные данные

Входными данными в модуле View являются текстовые и графические данные из удалённой БД.

Выходными данными для модуля View является информация, вводимая в различные компоненты на форме. Изменения автоматически сохраняются в соответствующих документах в БД.

Входными данными в модуле AuthWindow.xaml.cs являются данные из документа «пользователи». При авторизации пользователя входные и выходные данные хранятся в сущности User в следующих полях:

* string Login – содержит введенный логин пользователя при регистрации;
* string Password – содержит введенный пароль пользователя.

Выходными данными для модуля MainWindow.xaml.cs являются отчеты, которые можно сформировать из документов «пользователи», «студенты», «группы», «учебные планы».

Входными данными в модулях StudentPage.xaml.cs являются данные из документа «Студенты».

Выходными данными для модулей StudentPage.xaml.cs является информация, вводимая в различные компоненты форм добавления и редактирования отчетов.

Входными данными в модуле MainPage.xaml.cs являются данные из документа «Новости».

Выходными данными для модуля AddOrEditNews.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле OrdersPage.xaml.cs являются данные из документа «Приказы».

Выходными данными для модуля OrdersPage.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле AddOrUpdateStudentWindow.xaml.cs являются данные из документа «Студенты».

Выходными данными для модуля AddOrUpdateStudentWindow.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле AddOrEditEmployee.xaml.cs являются данные из документа «Сотрудники».

Выходными данными для модуля AddOrEditEmployee.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле EducationalPlansPage.xaml.cs являются данные из документа «Учебные планы».

Выходными данными для модуля EducationalPlansPage.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

2.4 Установка и настройка программы

Программа устанавливается на компьютер при запуске файла «Setup.exe». При установке программы происходит создание папки «CIS MGGTK» в папке программ системы и осуществляется копирование файлов, необходимых для запуска и корректной работы программы:

* папки:
* Assets;
* runtimes;
* ru;
* libraries;
* CIS MGGTK.exe;
* uninstall.exe.

Для корректной работы приложения компьютер должен иметь стабильное интернет-соединение.

Программу можно удалить, выбрав пункт меню «Деинсталлировать» в меню Пуск/Программы или запустив файл uninstall.exe в папке программы.

Для нормальной работы программы необходимо:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 1 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 2 ГБ;
* дисплей с разрешением не менее 1024x768 точек;
* 130 МБ свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

Для создания инсталляции была использована программа Inno Setup. В ходе инсталляции пользователю показываются следующие окна:

* приветствие – в нем предлагается закрыть все запущенные приложения;
* краткая информация об устанавливаемом приложении – назначение, минимальные требования к программному обеспечению и техническим средствам;
* выбор папки установки;
* создание дополнительных ярлыков – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыков;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

2.5 Работа с программой

Для запуска программы нужно нажать на иконку с названием «CIS MGGTK.exe», двойным кликом. В результате открывается окно авторизации приложения, в котором необходимо указать логин и пароль для входа в систему (рисунок 12).

При выборе пункта «Запомнить меня» система сохранит учётные данные пользователя для последующих попыток входа на компьютере. Для безопасности система будет запрашивать подтверждение пароля каждые два дня.

После авторизации открывается главное окно c активной страницей новостей (рисунок 13).



Рисунок 12 – Окно авторизации

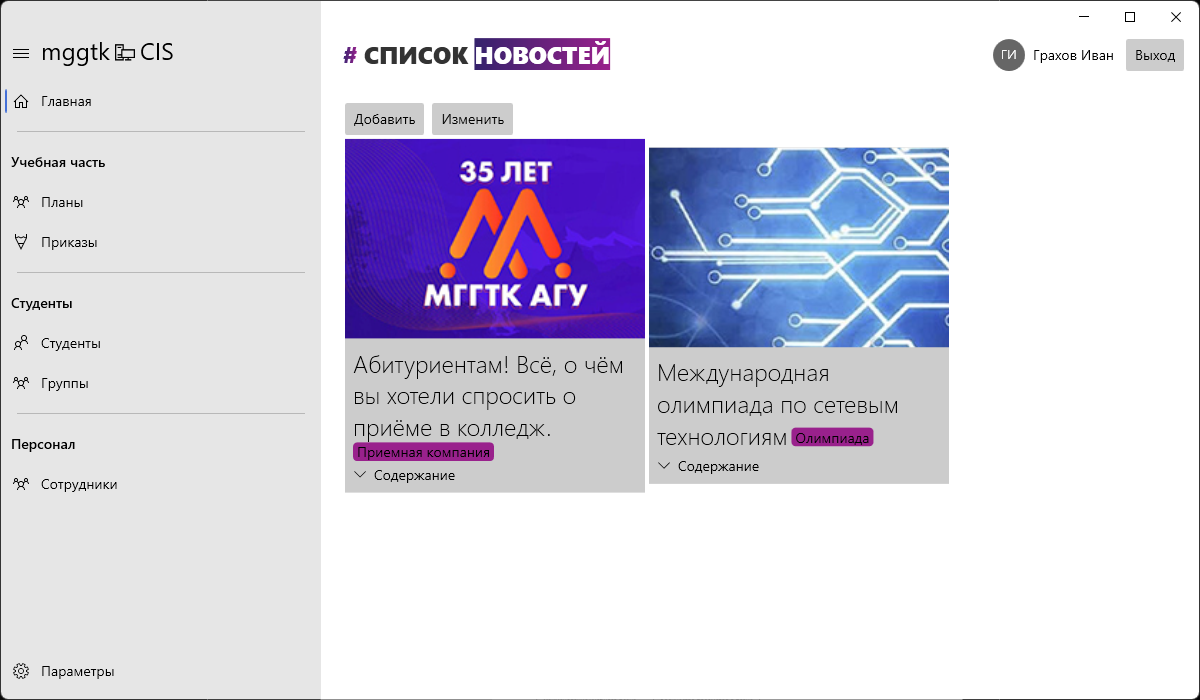


Рисунок 13 – Главное окно

При выборе пункта меню «Планы», открывается новое окно с информацией об учебных планах (рисунок 14).

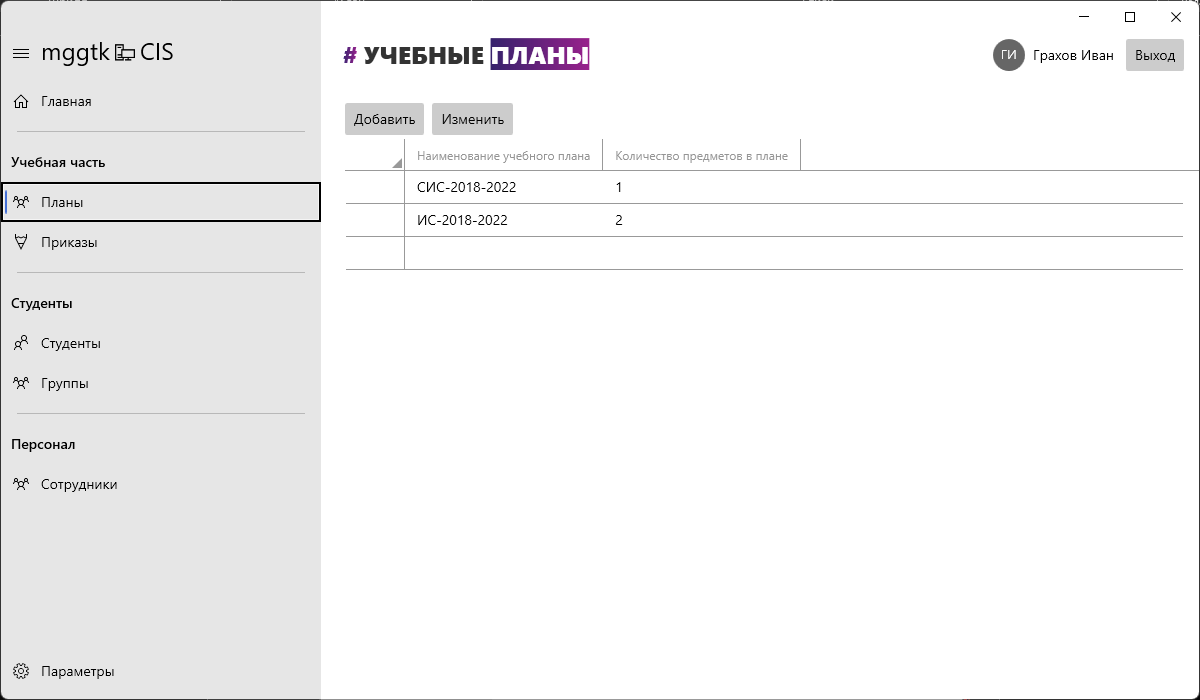


Рисунок 14 – Окно информации об учебных планах

На каждой форме есть возможность добавления и изменения нужной информации. Окно добавления/редактирования учебного плана предоставлено на рисунке 15.

Нажатие на кнопку «Сохранить» приведет к сохранению записей в целевой БД. При вводе осуществляется автоматическая проверка корректности значений.

При выборе пункта меню «Приказы», открывается новое окно с информацией о приказах (рисунок 16).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 – Добавление/редактирование учебного плана

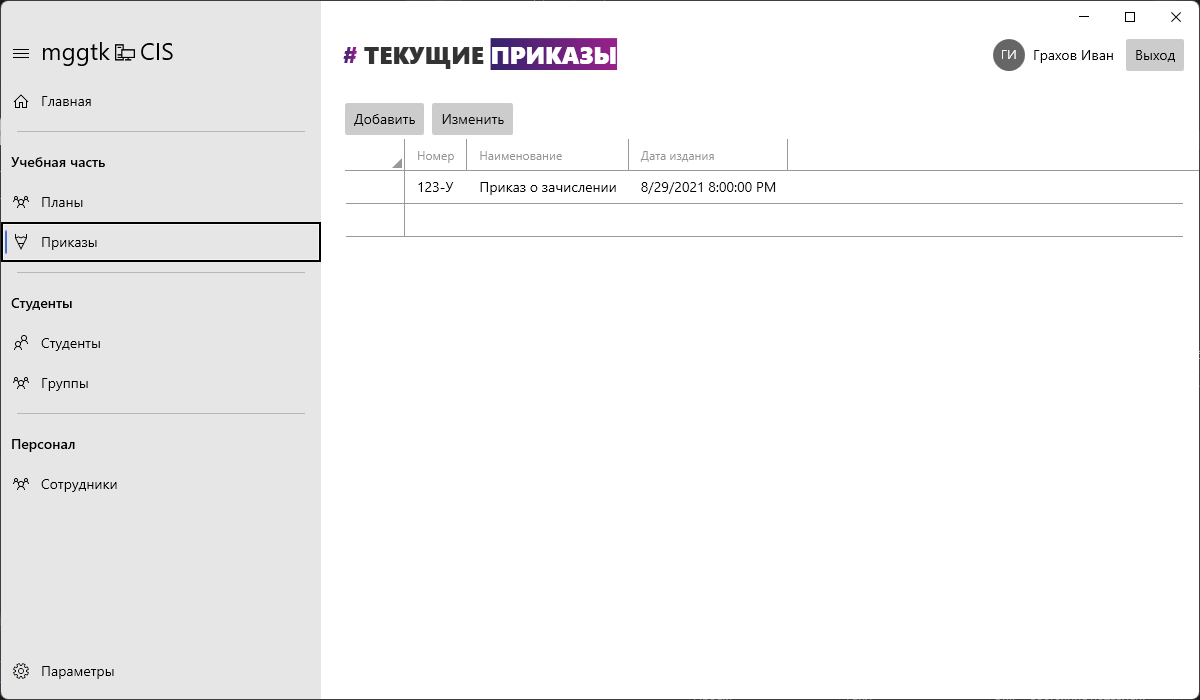


Рисунок 16 – Информация о текущих приказах

После нажатия на кнопку «Добавить» откроется окно добавления/редактирования информации о приказе (рисунок 17). При вводе осуществляется автоматическая проверка корректности значений. Все поля являются обязательными для заполнения.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – Добавление/редактирование приказов

Если в главном окне приложения выбрать пункт меню «Студенты», то откроется окно со сводной информацией обо всех обучающихся на отделении (рисунок 18).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18 – Окно со списком студентов

При выборе пункта «Показать карточку» из контекстного меню открывается окно с электронной карточкой студента (рисунок 19).

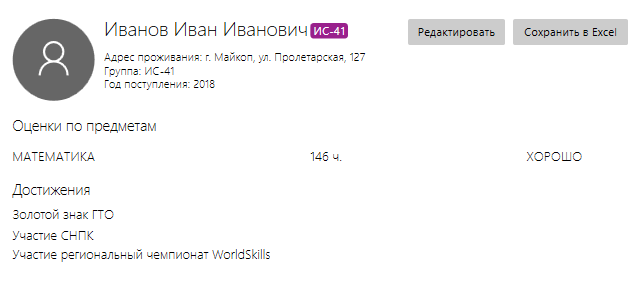


Рисунок 19 – Электронная карточка студента

Если в главном окне приложения выбрать пункт меню «Группы», то откроется окно со сводной информацией обо учебных группах (рисунок 20).

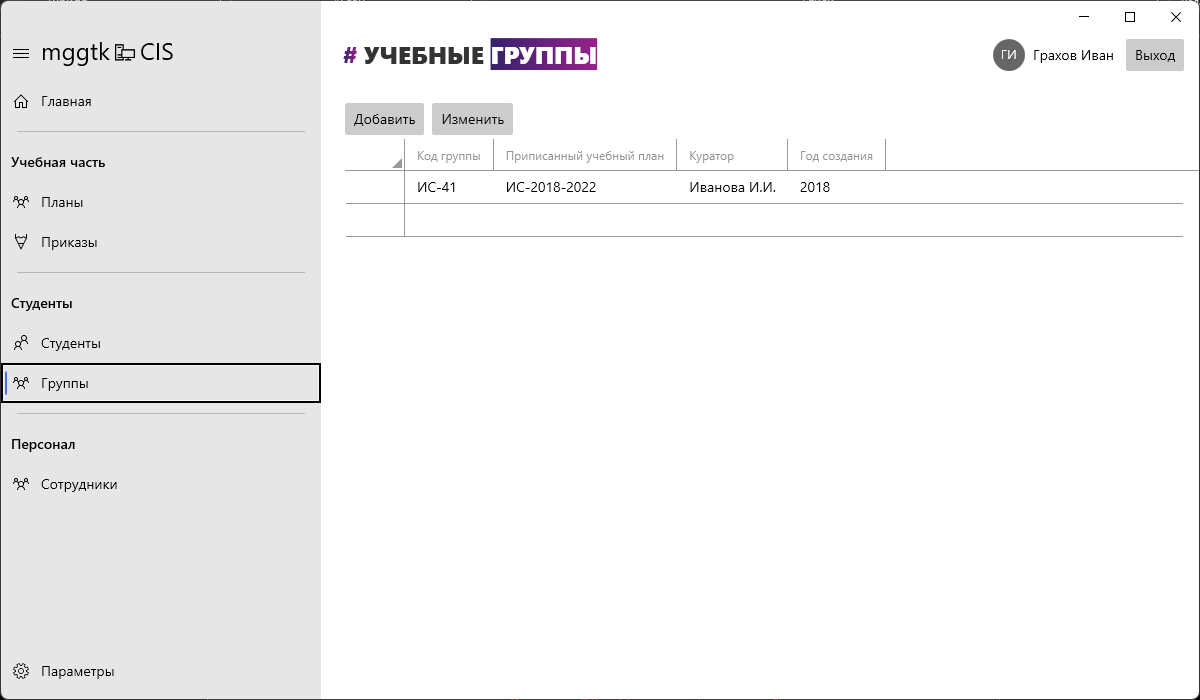


Рисунок 20 – Список учебных групп

Если в главном окне приложения выбрать пункт меню «Сотрудники», то откроется окно со сводной информацией о сотрудниках, работающих на отделении информационных технологий (рисунок 21).

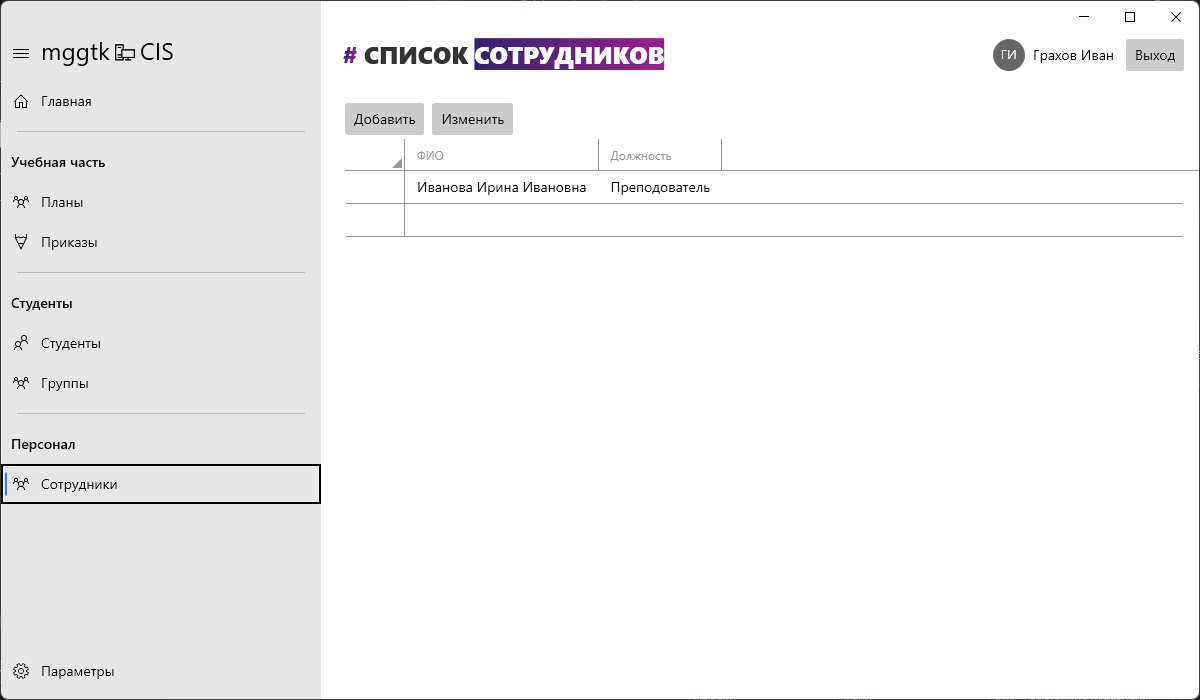


Рисунок 21 – Список сотрудников

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана клиент-серверная информационная система для учета и хранения электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ». Также было разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты.

В ходе выполнения дипломного проекта была достигнута поставленная цель и решены следующие задачи:

* выполнен анализ предметной области и спроектированы концептуальная и логическая модели БД для хранения необходимой информации;
* спроектирована функциональность и структура приложения, работающего с созданной БД. При проектировании использовался инструментарий UML-диаграмм;
* произведен выбор средств реализации БД, приложения и инсталляции;
* определены минимальные требования к техническим и программным средствам;
* создана физическая модель БД с помощью СУБД SQL Server;
* с помощью средств интегрированной среды разработки Visual Studio 2019 и языка программирования С# был разработан интерфейс приложения, позволяющий получить доступ ко всем необходимым функциям;
* с помощью средств интегрированной среды разработки Visual Studio 2019 и языка программирования С# были разработаны модули, обеспечивающие выполнение всех необходимых функций;
* создана инсталляция с помощью средств Inno Setup.

Полученные в результате анализа предметной области данные были распределены по документам, которые логически связаны между собой. Основываясь на логической связи документов, строились запросы к БД.

Для доступа к БД были использованы компоненты среды программирования C#, реализующие технологию параллельных запросов (MongoDB Driver).

Удобный интерфейс программы, с одной стороны, позволяет легко ориентироваться в приложении, не требуя от пользователя каких-либо специальных навыков работы, с другой стороны – предоставляет пользователю оперативную информацию.

Разработанное приложение обеспечивает следующие возможности:

* авторизацию пользователей;
* доступ к окнам и функциям системы через основное меню;
* внесение информации;
* редактирование информации;
* поиск информации;
* фильтрация информации по запросам пользователя;
* сортировку информации;
* работу с информационной БД.

Основные достоинства разработанного приложения:

* низкая требовательность к ресурсам системы;
* простота установки и настройки;
* удобство эксплуатации;
* возможность оперативного получения необходимых отчетов;
* возможность сохранения отчетов в виде файлов на компьютере в формате .xlsx.

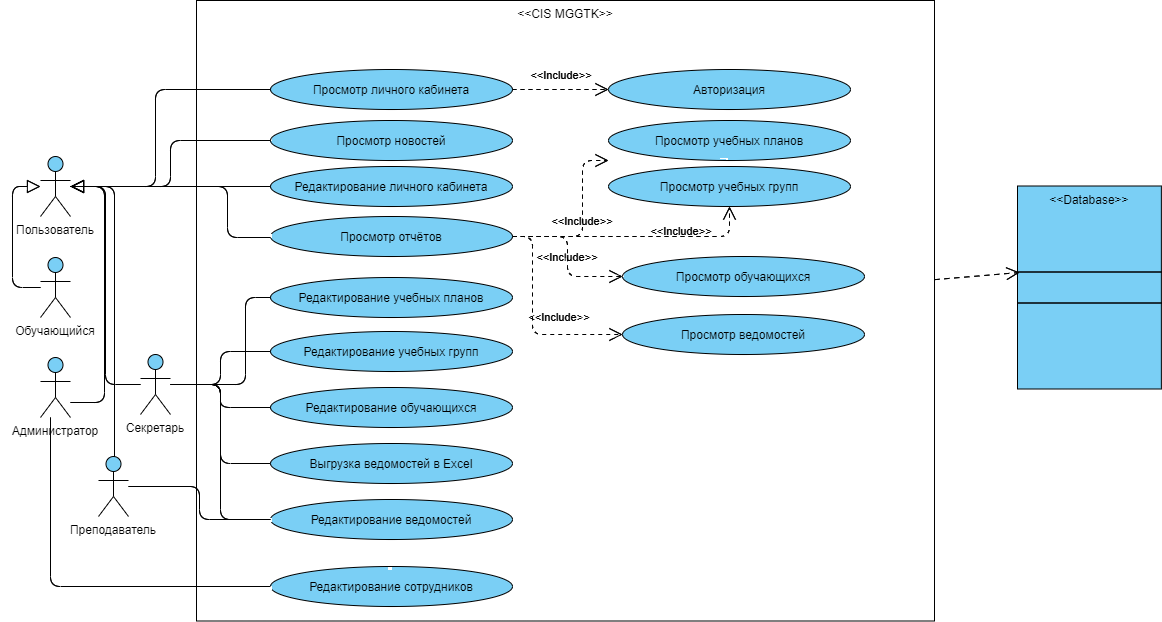
К недостаткам данной программы можно отнести то, что информация об отделении хранится в незашифрованном виде.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Албахари Б., Албахари Дж. С# 7.0. Справочник. Полное описание языка. М.: Вильямс, 2018. 1024 с.
2. Бэнкер К. MongoDB в действии. М.: ДМК Пресс, 2018. 838 c.
3. Виссер Д. Разработка обслуживаемых программ на языке С#. М.: ДМК Пресс, 2019. 194 с.
4. Кугаевских А. Проектирование информационных систем. Системная и бизнес-аналитика. Новосибирск: НГТУ, 2018. 256 с.
5. Лотка Р. C# и CSLA .NET Framework. Разработка бизнес-объектов. М.: Диалектика / Вильямс, 2018. 842 c.
6. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 5.0 и платформа .NET 4.5 для профессионалов. М.: Вильямс, 2018. 693 c.
7. Скит Д. С# Для профессионалов. Тонкости программирования. М.: Вильямс, 2019. 608 с.
8. Borland B. Pentaho Analytics for Mongodb. М.: Книга по Требованию, 2018. 146 c.
9. Документация Microsoft. URL: https://docs.microsoft.com/ru-ru/documentation (дата обращения: 28.04.2022).
10. Официальный сайт Inno Setup. URL: https://jrsoftware.org/isinfo.php (дата обращения: 28.04.2022).
11. Простое руководство по UML-диаграммам и моделированию баз данных. URL: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата обращения: 28.04.2022).

ПРИЛОЖЕНИЕ А

Use Case диаграмма проектируемой информационной системы



ПРИЛОЖЕНИЕ Б

Код основных классов

Класс FrameManager.cs

using System;

using System.Windows.Controls;

namespace CIS\_MGGTK.Desktop.Service.ApplicationService

{

public class FrameManager

{

public static Frame MainFrame { get; set; }

public static void SetSource<T>(T target) where T : Page

{

var contentType = MainFrame.Content?.GetType();

if (contentType != typeof(T))

{

MainFrame.Navigate(target);

MainFrame.NavigationService.RemoveBackEntry();

GC.Collect();

}

}

}

}

Класс BaseCRUDService.cs

using MongoDB.Bson;

using MongoDB.Driver;

using System.Linq;

using System.Threading.Tasks;

namespace CIS\_MGGTK.Desktop.Service.DatabaseService

{

public class BaseCRUDService<T> : IBaseCRUDService<T> where T : class,

IDocument

{

private readonly MongoClient \_client;

private readonly IMongoDatabase \_database;

private readonly IMongoCollection<T> \_collection;

public BaseCRUDService(MongoClient client, string db, string collection)

{

\_client = client;

\_database = \_client.GetDatabase(db);

\_collection = \_database.GetCollection<T>(collection);

}

public IMongoDatabase GetDatabase() =>

\_database;

public async Task Add(T document)

{

await \_collection.InsertOneAsync(document);

}

public async Task Delete(string id)

{

await \_collection.DeleteOneAsync(new BsonDocument("\_id",

new ObjectId(id)));

}

public async Task<T> Get(string id)

{

return await \_collection.Find(new BsonDocument("\_id",

new ObjectId(id))).FirstOrDefaultAsync();

}

public IQueryable<T> GetAll()

{

return \_collection.AsQueryable();

}

public async Task Update(T document)

{

await \_collection.ReplaceOneAsync(new BsonDocument("\_id",

new ObjectId(document.Id.ToString())), document);

}

}

}

Класс UserAuthenticationService.cs

using CIS\_MGGTK.Desktop.Data.Model;

using MongoDB.Driver;

using System.Threading.Tasks;

namespace CIS\_MGGTK.Desktop.Service.DatabaseService

{

public class UserAuthenticationService

{

private readonly MongoClient \_client;

private readonly IMongoDatabase \_database;

private readonly IMongoCollection<User> \_collection;

public UserAuthenticationService(MongoClient client, string db)

{

\_client = client;

\_database = \_client.GetDatabase(db);

\_collection = \_database.GetCollection<User>("users");

}

public async Task<User> Authorize(string login, string password)

{

return await \_collection.Find(x => (x.Login == login || x.Email == login) &&

x.Password == password).FirstOrDefaultAsync();

}

}

}

Класс AuthenticationService.cs

using CIS\_MGGTK.Desktop.Service.DatabaseService;

using CIS\_MGGTK.Desktop.View;

using System.Threading.Tasks;

using System.Windows;

namespace CIS\_MGGTK.Desktop.Services.AuthenticationServices

{

public class AuthenticationService : IAuthenticationService

{

private UserAuthenticationService \_userAuthenticationService;

public AuthenticationService()

{

\_userAuthenticationService = new(ClientManager.GetClient(),

ClientManager.GetDatabase());

}

public void InvalidateSession()

{

var auth = new AuthWindow();

auth.Show();

foreach (Window item in App.Current.Windows)

{

if (!(item is AuthWindow))

{

item.Close();

}

}

}

public async Task<bool> ValidateSession(string login, string password,

bool isSaved)

{

var result = await \_userAuthenticationService.Authorize(login, password);

if (result != null)

{

SessionStorage.User = result;

return true;

}

return false;

}

}

}