**АКАДЕМИЯ ГОСУДАРСТВЕННОГО УПРАВЛЕНИЯ**

**ПРИ ПРЕЗИДЕНТЕ АЗЕРБАЙДЖАНСКОЙ РЕСПУБЛИКИ**

**Шифр и наименование специальности:** 060509 Компьютерные науки

**Шифр и наименование специализации:** Цифровое правительство

**Кафедра:** Информационные технологии в государственном управлении

**Группа:** RH221-222

**Раздел:** Русский

**М А Г И С Т Е Р С К А Я Р А Б О Т А**

НА ТЕМУ:

**ПРИМЕНЕНИЕ АВТОМАТИЗИРОВАННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В СИСТЕМЕ МЕНЕДЖМЕНТА УНИВЕРСИТЕТА С ПРИМЕНЕНИЕМ C#**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Магистрант:** |  | Ахундова Минира Азад | |  | (подпись) | *фамилия, имя, отчество* | | **Руководитель:** |  | к.п.н., доц. Гаджиева Р.Д. | |  | (подпись) | *научная степень, фамилия, имя, отчество* | | **Заведующий кафедры:** |  | к.э.н., доц. Абасов Е.А. | |  | (подпись) | *научная степень, фамилия, имя, отчество* | |

**БАКУ – 2024**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](#_Toc41567237) 3

[ГЛАВА 1. ОБЩИЕ ПОНЯТИЯ ПО ЯЗЫКУ ПРОГРАММИРОВАНИЯ C# И ПЛАТФОРМЕ.NET…….......…….](#_Toc41567238).......................................................................8

1.1.Создание и становление C#.................................................................................8

1.2.[Связь C# со средой .NET Framework](https://professorweb.ru/my/csharp/charp_theory/level1/1_3.php)………...…………………...…...........…20

1.3.Роль платформы .NET и типы его приложений……………....………......….22

[ГЛАВА 2. ОСНОВЫ ASP.NET, ASP.NET MVC И ВИДЫ СУБД..................26](#_Toc41567247)

2.1.Основы и особенности ASP.NET……..……………………………………….26

2.2.Фреймворк ASP.NET Core MVC и его преимущества …………….........…...31

2.3.СУБД и их виды…............................................................……………….…......39

**ГЛАВА 3.ПРАКТИКА СИСТЕМ МЕНЕДЖМЕНТА УНИВЕРСИТЕТА НА ОСНОВЕ АИС В РАЗЛИЧНЫХ СТРАНАХ, А ТАКЖЕ ПРАКТИЧЕСКАЯ РАЗРАБОТКА ПРОЕКТА ....................................................................................51**

3.1.Системы менеджмента университета на АИС в различных странах….....…51

3.2.Построение файловой структуры проекта и установка связи между нашим проектом и системой управления базами данных .…...........................................54

3.3.Демонстрация основных функциональностей системы.…..……….….….…65

[ЗАКЛЮЧЕНИЕ](#_Toc41567260) 72

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ](#_Toc41567261) 74

**ВВЕДЕНИЕ**

В современном обществе информация играет ключевую роль в продвижении человечества. Это объясняется тем, что информация тесно связана с любой сферой человеческой деятельности и требует обработки. Количество информации огромно, и обработка и хранение ее становятся все более сложными с использованием традиционных методов. В связи с этим возникает потребность в автоматизации обработки информации в различных областях человеческой деятельности.

Автоматизированные информационные технологии (АИТ) представляют собой область, где компьютеры и программное обеспечение используются для автоматизации процессов, связанных с сбором, хранением, обработкой, анализом и передачей информации. Эти технологии играют важную роль в современных организациях и предприятиях, способствуя увеличению эффективности и принятию более обоснованных решений. Они помогают снизить операционные расходы, увеличить производительность и обеспечивают быстрый доступ к необходимой информации. АИТ можно рассматривать как систему автоматизации информационных ресурсов.

Автоматизированная информационная система (АИС) представляет собой совокупность программных и аппаратных средств, разработанных для сбора, хранения, обработки и передачи информации внутри организации или для решения конкретных задач. АИС объединяет технологии, методы и процессы, направленные на улучшение управления и операционной деятельности организации. В рамках автоматизированной информационной системы все факторы и ресурсы отражаются в общей информационной среде в виде согласованных данных. Один из наглядных примеров таких систем - система автоматизации управления университетом.

Как и во многих других областях, включая сложные системы безопасности и устройства в наших домах, в АИС также используется код, написанный на различных языках программирования. Один из таких языков программирования - C#.

C# представляет собой современный язык программирования общего назначения, являющийся частью семейства языков, произошедших от языка C, таких как C++ и C, и схожих с Java и JavaScript. Он является одним из ключевых языков для разработки приложений под платформой. .NET Framework, а также .NET Core. C# давно предоставляет множество полезных функций, таких как инкапсуляция, наследование, полиморфизм, перегрузка операторов и статическая типизация.

Использование автоматизированных информационных технологий (АИТ) в системе управления имеет ключевое значение для повышения эффективности и производительности в организации. Примеры применения АИТ в системе управления включают:

1. Управление проектами

2. Управление персоналом

3. Управление задачами и процессами

Использование автоматизированных информационных технологий (АИТ) в управлении университетом с применением C# может существенно улучшить эффективность и точность администрирования университетской деятельности. Автоматизация позволит упростить рутинные задачи, обеспечить более удобный доступ к данным и оптимизировать взаимодействие между учебными заведениями, факультетами, студентами и другими участниками образовательного процесса. Ниже представлены несколько способов, как это можно реализовать:

1. Учет студентов и управление их данными:

- Разработать C#-приложение для регистрации студентов, учета их личных данных, академической информации, а также истории курсов и успехов.

- Обеспечить быстрый поиск и фильтрацию студентов по различным параметрам.

- Создать систему автоматических уведомлений о важных событиях и сроках, таких как ближайшие экзамены.

2. Управление учебными курсами:

- Разработать систему управления учебными курсами с возможностью добавления, редактирования и удаления курсов администраторами.

- Предоставить автоматическую регистрацию студентов на курсы и отслеживание их академического прогресса.

3. Оптимизация управления финансами:

- Создать систему учета финансов университета, включая бюджетирование, учет расходов и доходов.

- Разработать инструменты для автоматической генерации отчетов и анализа финансового состояния университета.

4. Мониторинг и анализ данных:

- Разработать панели управления на C#, которые позволят администраторам и руководству университета мониторить и анализировать ключевые показатели, такие как академический успех студентов и загрузка преподавателей.

- Внедрить систему аналитики данных для выявления тенденций и принятия обоснованных решений.

5. Управление ресурсами университета:

- Разработать систему управления ресурсами, включая учёт учебных аудиторий, расписание занятий и управление активами.

6. Электронное обучение и дистанционное образование:

- Разработать платформу для проведения онлайн-курсов и вебинаров.

- Обеспечить интеграцию с системой учета студентов и их успехов.

Для успешной реализации таких проектов на C# может потребоваться использование современных технологий разработки, таких как ASP.NET, Entity Framework, WPF, а также баз данных, например, SQL Server. Важно также обеспечить безопасность данных и защиту конфиденциальной информации студентов и университета.

**Актуальность диссертационной работы.** Применение АИТ в системе менеджмента университета с применением C# имеет несколько актуальных аспектов:

1.Автоматизация рутинных процессов: Использование C# позволяет разработать программные решения для автоматизации рутинных задач управления университетом. Например, это может быть автоматизация процессов регистрации студентов, формирование расписания занятий, управление библиотечными ресурсами и т.д.

2.Интеграция с существующими системами: Часто у университетов уже есть различные информационные системы, такие как системы учета успеваемости, электронные библиотеки, системы электронного обучения и другие. Применение C# позволяет легко интегрировать новые и существующие системы, обеспечивая целостность данных и снижая необходимость в ручной обработке информации.

3.Безопасность и защита данных: C# предоставляет мощные инструменты для обеспечения безопасности приложений и данных. Это критически важно для университетов, учитывая чувствительность персональной информации студентов и сотрудников.

4.Развитие аналитических возможностей: Применение C# позволяет реализовать аналитические инструменты для обработки данных, что помогает университетам анализировать статистику посещаемости, успеваемости студентов, эффективности преподавания и другие параметры, что в свою очередь помогает улучшать качество образования и управления университетом.

**Цель диссертационной работы.**

Целью диссертационной работы является объяснение и наглядная демонстрация актуальности данной темы в системе университета, при помощи современных инструментов автоматизации, таких как языки программирования, библиотеки, платформы, а также рейтинговых СУБД. В рамках темы диссертационной работы будут реализованы современные подходы построения бизнес логики проекта, такие как вход в систему при помощи аутентификации и идентификации, формирование и фильтрация студентов записанных на определённые предметы, часов преподавателей, свободных и занятых аудиторий, настройка различной информации (удаление, создание, модификация) и т.д..

**Структура и объём работы.** Диссертационная работа состоит из титульного листа, введения, трёх глав, 9 разделов, заключения, списка литературы, а также из рисунков, что в общем составило 75 страниц.

В главе 1 приведены общие понятия по языку программирования C# и платформе .NET. О создании и становлении C#, его связью со средой .NET Framework, роли данной платформы и типах приложений, которые можно реализовать при помощи неё, а также их актуальности и универсальности.

Во главе 2 мы ознакомимся с основами и особенностями ASP.NET, ASP.NET MVC. Также здесь будет разъяснены актуальности и преимущества различных топовых СУБД, которые помогут реализовать информационную сторону нашей системы.

В главе 3 мы приведём различия и особенности систем менеджмента университета на основе автоматизированных информационных систем в различных странах, а также приступим к практической части нашей работы, а именно реализуем веб интерфейс системы, созданием Entity, DTO, Repositorу, сервисов и контроллеры и вслучае необходимости настроим валидации. Здесь также будут продемонстрированы последние этапы разработки проекта, как установка связи между СУБД и системой при помощи программы DBeaver.

ГЛАВА 1. ОБЩИЕ ПОНЯТИЯ ПО ЯЗЫКУ ПРОГРАММИРОВАНИЯ

C# И ПЛАТФОРМЕ .NET

1.1. Создание и становление C#

Язык C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программи­рования, который был специально создан для использования в .NET Framework. Например, с применением C# и .NET Framework можно создавать динамичес­кие веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возмож­ностями для работы в оперативном и автономном режимах [1, стр. 18-57].

Наличиe в названии Framework слова "NET" и думать, что данная среда предназначена только для создания приложений, ориентированных на Интернет. Слово "NET" здесь является лишь показателем того, что, по мнению Microsoft, распределенные приложения, в которых обработка распределяется между клиентом и сервером, являются шагом вперед. Однако важно понимать, что C# представляет собой язык, предназначенный не только для написания приложений, способных работать в Интернете и в сети. Он предоставляет средства для кодирования практически любого типа программного обеспечения или компонентов для платформы Windows. Язык C# и среда .NET привели к революционным изменениям в способе написания разработчиками программ и сделали программирование приложений для Windows гораздо более простым, чем когда-либо.

C# — это относительно новый язык программирования, который характеризуется двумя следующими преимуществами:

C# спроектирован и разработан специально для применения с Microsoft .NET Framework (развитой платформой разработки, развертывания и выполнения распределенных приложений).

C# — язык, основанный на современной объектно-ориентированной методологии проектирования, при разработке которого специалисты из Microsoft опирались на опыт создания подобных языков, построенных в соответствии с предложенными около 20 лет назад объектно-ориентированными принципами.

Нужно подчеркнуть то важное обстоятельство, что C# — это полноценный язык программирования. Хотя он и предназначен для генерации кода, выполняемого в среде .NET, сам по себе он не является частью .NET. Существует ряд средств, которые поддерживаются .NET, но не поддерживаются C#, и, возможно, вас удивит, что есть также средства, поддерживаемые C# и не поддерживаемые .NET (например, некоторые случаи перегрузки операций). Однако поскольку язык C# предназначен для применения на платформе .NET, вам, как разработчику, важно иметь представление о .NET Framework, если вы хотите эффективно разрабатывать приложения на C# [20, стр. 92].

Для понимания важности .NET не помешает вспомнить о природе многих технологий Windows, которые появились в последние примерно 18 лет. Хотя на первый взгляд все они могут выглядеть довольно разными, на самом деле все операционные системы Windows, начиная с Windows 3.1 (которая вышла в 1992 г.) и заканчивая Windows 7 и Windows Server 2008 R2, в основе своей имеют один и тот же хорошо знакомый API-интерфейс Windows. По мере появления новых версий Windows в этот API-интерфейс добавлялось много новых функций, но это был скорее процесс совершенствования и расширения API-интерфейса, а не его замена.

Давайте рассмотрим язык C# в его историческом контексте, упомянув и те движущие силы, которые способствовали его становлению.

**Язык C и API-интерфейс Windows**

Создание С знаменует собой начало современной эпохи программи­рования . Язык С был разработан Деннисом Ритчи (Dennis Ritchie) в 70-е годы для программирования на мини-ЭВМ DEC PDP-11 под управлением операционной системы Unix. Несмотря на то что в ряде предшествовавших языков, в особенности Pascal, был достигнут значительный прогресс, именно С установил тот образец, которому до сих пор следуют в программировании.

Язык С появился в результате революции в структурном программи­ровании в 1960-е годы. До появления структурного программирования писать большие программы было трудно, поскольку логика программы постепенно вырождалась в так называемый "макаронный" код — запутанный клубок безусловных переходов, вызовов и возвратов, которые трудно отследить. В структурированных языках программирования этот недостаток устранялся путем ввода строго определенных управляющих операторов, подпрограмм с локальными переменными и других усовершенствований. Благодаря применению методов структурного программирования сами программы стали более организованными, надежными и управляемыми [5, стр. 413].

И хотя в то время существовали и другие структурированные языки программирования, именно в С впервые удалось добиться удачного сочетания эффективности, изящества и выразительности. Благодаря своему краткому, но простому синтаксису в сочетании с принципом, ставившим во главу угла программиста, а не сам язык, С быстро завоевал многих сторонников. Сейчас уже нелегко представить себе, что С оказался своего рода "струей свежего воздуха", которого так не хватало программистам. В итоге С стал самым распространенным языком структурного программирования в 80-е годы.

Но даже у такого достойного языка, как С, имелись свои ограничения. К числу самых труднопреодолимых его ограничений относится неспособность справиться с большими программами. Как только проект достигает определенного масштаба, язык С тут же ставит предел, затрудняющий понимание и сопровождение программ при их последующем разрастании. Конкретный предел зависит от самой программы, программиста и применяемых инструментальных средств, тем не менее, всегда существует "порог", за которым программа на С становится неуправляемой.

Традиционно разработка программного обеспечения для операционных систем семейства Windows подразумевала использование языка программи­рования С в сочетании с API-интерфейсом Windows (Application Programming Interface — интерфейс прикладного программирования). И хотя то, что за счет применения этого проверенного временем подхода было успешно создано очень много приложений, мало кто станет спорить по поводу того, что процесс создания приложений с помощью одного только API-интерфейса является очень сложным занятием. Первая очевидная проблема состоит в том, что С представляет собой очень лаконичный язык. Разработчики программ на языке С вынуждены мириться с необходимостью "вручную" управлять памятью, безобразной арифметикой указателей и ужасными синтаксическими конструкциями [7, стр 86]. Из-за сочетания тысяч глобальных функций и типов данных, определенных в API-интерфейсе Windows, с языком, который и без того выглядит устрашающе, совсем не удивительно, что сегодня в обиходе присутствует столь много дефектных приложений.

**Появление C++ и платформы MFC**

К концу 70-х годов масштабы многих проектов приблизились к пределам, с которыми уже не могли справиться методики структурного программирования вообще и язык С в частности. Для решения этой проблемы было открыто новое направление в программировании — так называемое объектно-ориентированное программирование (ООП) [3, стр.71]. Применяя метод ООП, программист мог работать с более "крупными" программами. Но главная трудность заключалась в том, что С, самый распространенный в то время язык, не поддерживал ООП. Стремление к созданию объектно­ориентированного варианта С в конечном итоге привело к появлению С++.

Язык С++ был разработан в 1979 году Бьярне Страуструпом (Bjarne Stroustrup), работавшим в компании Bell Laboratories, базировавшейся в Мюррей-Хилл, штата Нью-Джерси. Первоначально новый язык назывался "С с классами", но в 1983 году он был переименован в С++. Язык С полностью входит в состав С++, а следовательно, С служит основанием, на котором зиждется С++. Большая часть дополнений, введенных Страуструпом, обеспечивала плавный переход к ООП. И вместо того чтобы изучать совершенно новый язык, программирующему на С требовалось лишь освоить ряд новых свойств, чтобы воспользоваться преимуществами методики ООП.

В течение 1980-х годов С++ все еще оставался в тени, интенсивно развиваясь, но к началу 1990-х годов, когда он уже был готов для широкого применения, его популярность в области программирования заметно возросла. К концу 1990-х годов он стал наиболее широко распространенным языком программирования и в настоящее время по-прежнему обладает неоспоримыми преимуществами языка разработки высокопроизводительных программ системного уровня.

Важно понимать, что разработка С++ не была попыткой создать совершенно новый язык программирования. Напротив, это была попытка усовершенствовать уже существовавший довольно удачный язык. Такой подход к разработке языков программирования, основанный на уже существующем языке и совершенствующий его далее, превратился в упрочившуюся тенденцию, которая продолжается до сих пор.

Язык С++ во многих отношениях может считаться объектно-ориенти­рованной надстройкой поверх языка С. Из-за этого, хотя в случае его применения программисты уже могут начинать пользоваться преимуществами известных "главных столпов ООП" (таких как инкапсуляция, наследование и полиморфизм), они все равно вынуждены иметь дело с утомительными деталями языка С (вроде необходимости осуществлять управление памятью "вручную", безобразной арифметики указателей и ужасных синтаксических конструкций).

Невзирая на сложность, сегодня существует множество платформ для программирования на С++. Например, MFC (Microsoft Foundation Classes — библиотека базовых классов Microsoft) предоставляет в распоряжение разработчику набор классов С++, которые упрощают процесс создания приложений Windows. Основное предназначение MFC заключается в представлении "разумного подмножества" исходного API-интерфейса Windows в виде набора классов, "магических" макросов и многочисленных средств для автоматической генерации программного кода (обычно называемых мастерами). Несмотря на очевидную пользу данной платформы приложений (и многих других основанных на С++ наборов средств), процесс программирования на С++ остается трудным и чреватым допущением ошибок занятием из-за его исторической связи с языком С.

**Visual Basic 6.**

Многие программисты перешли из "мира платформ" на базе С (С++) в мир менее сложных и более дружественных языков наподобие Visual Basic 6.0 (VB6). Язык VB6 стал популярным благодаря предоставляемой им возможности создавать сложные пользовательские интерфейсы, библиотеки программного кода (вроде СОМ-серверов) и логику доступа к базам данных с приложением минимального количества усилий. Во многом как и в MFC, в VB6 сложности API-интерфейса Windows скрываются из вида за счет предоставления ряда интегрированных мастеров, внутренних типов данных, классов и специфических функций VB.

Главный недостаток языка VB6 (который с появлением платформы .NET был устранен) состоит в том. что он является не полностью объектно-ориентированным, а скорее — просто "объектным". Например, VB6 не позволяет программисту устанавливать между классами отношения "подчиненности" (т.е. прибегать к классическому наследованию) и не обладает никакой внутренней поддержкой для создания параметризованных классов. Более того, VB6 не предоставляет возможности для построения многопоточных приложений, если только программист не готов опускаться до уровня вызовов API-интерфейса Windows (что в лучшем случае является сложным, а в худшем — опасным подходом).

Язык Visual Basic, используемый внутри платформы .NET (и часто называемый языком VB.NET), имеет мало чего общего с языком VB6. Например, в современном языке VB поддерживается перегрузка операций, классическое наследование, конструкторы типов и обобщения.

**Java.**

Следующим важным шагом в развитии языков программирования стала разработка Java. Работа над языком Java, который первоначально назывался Oak (Дуб), началась в 1991 году в компании Sun Microsystems. Главной "движущей силой" в разработке Java был Джеймс Гослинг (James Gosling), но немалая роль в работе над этим языком принадлежит также Патрику Ноутону (Patrick Naughton), Крису Уорту (Chris Warth), Эду Фрэнку (Ed Frank) и Майку Шеридану (Mike Sheridan).

Java представляет собой структурированный, объектно-ориентированный язык с синтаксисом и конструктивными особенностями, унаследованными от С++. Нововведения в Java возникли не столько в результате прогресса в искусстве программирования, хотя некоторые успехи в данной области все же были, сколько вследствие перемен в вычислительной среде. До появления на широкой арене Интернета большинство программ писалось, компилировалось и предназначалось для конкретного процессора и операционной системы. Как известно, программисты всегда стремились повторно использовать свой код, но, несмотря на это, легкой переносимости программ из одной среды в другую уделялось меньше внимания, чем более насущным задачам. Тем не менее с появлением Интернета, когда в глобальную сеть связывались разнотипные процессоры и операционные системы, застаревшая проблема переносимости программ вновь возникла с неожиданной остротой. Для решения проблемы переносимости потребовался новый язык, и им стал Java.

Самым важным свойством (и причиной быстрого признания) Java является способность создавать межплатформенный, переносимый код, тем не менее, интересно отметить, что первоначальным толчком для разработки Java послужил не Интернет, а потребность в независящем от платформы языке, на котором можно было бы разрабатывать программы для встраиваемых контроллеров. В 1993 году стало очевидно, что вопросы межплатформенной переносимости, возникавшие при создании кода для встраиваемых контроллеров, стали актуальными и при попытке написать код для Интернета. В итоге оказалось, что теми же самыми методами, которыми решалась проблема переносимости программ в мелких масштабах, можно решать аналогичную задачу в намного более крупных масштабах Интернета.

Переносимость программ на Java достигалась благодаря преобразованию исходного кода в промежуточный, называемый байт-кодом. Этот байт-код затем выполнялся виртуальной машиной Java (JVM) — основной частью исполняющей системы Java. Таким образом, программа на Java могла выполняться в любой среде, для которой была доступна JVM. А поскольку JVM реализуется относительно просто, то она сразу же стала доступной для большого числа сред.

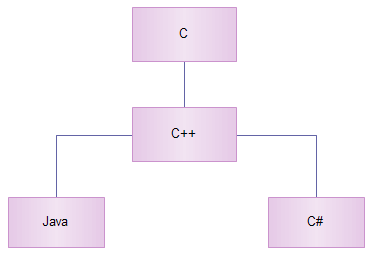
Применением байт-кода Java коренным образом отличается от С и С++, где исходный код практически всегда компилируется в исполняемый машинный код, который, в свою очередь, привязан к конкретному процессору и операционной системе. Так, если требуется выполнить программу на С или С++ в другой системе, ее придется перекомпилировать в машинный код специально для данной вычислительной среды. Следовательно, для создания программы на С или С++, которая могла бы выполняться в различных средах, потребовалось бы несколько разных исполняемых версий этой программы. Это оказалось бы не только непрактично, но и дорого. Изящным и рентабельным решением данной проблемы явилось применение в Java промежуточного кода. Именно это решение было в дальнейшем приспособлено для целей языка C#.

Хотя Java и представляет собой очень элегантный язык, одной из потенциальных проблем является то, что применение Java обычно означает необходимость использования Java в цикле разработки и для взаимодействия клиента с сервером. Надежды на появление возможности интегрировать Java с другими языками мало, поскольку это противоречит главной цели Java — быть единственным языком программирования для удовлетворения любой потребности. В действительности, однако, в мире существуют миллионы строк программного кода, которым бы идеально подошло смешивание с более новым программным кодом на Java. К сожалению, Java делает выполнение этой задачи проблематичной. Пока в Java предлагаются лишь ограниченные возможности для получения доступа к отличным от Java API-интерфейсам, поддержка для истинной межплатформенной интеграции остается незначительной.

**Создание C#**

Язык C# был разработан корпорацией Microsoft в конце 90-х годов как часть общей стратегии .NET. Впервые он был выпущен в виде альфа-версии в середине 2000 года. Главным разработчиком C# был Андерс Хейльсберг — один из ведущих в мире специалистов по языкам программирования, который может похвалиться рядом заметных достижений в данной области. Достаточно сказать, что в 80-е годы он был автором очень удачной и имевшей большое значение разработки — языка Turbo Pascal, изящная реализация которого послужила образцом для создания всех последующих компиляторов.

Язык C# непосредственно связан с С, С++ и Java [9, стр. 20-53]. И это не случайно. Ведь это три самых широко распространенных и признанных во всем мире языка программирования. Кроме того, на момент создания C# практически все профессиональные программисты уже владели С, С++ или Java. Благодаря тому что C# построен на столь прочном и понятном основании, перейти на этот язык из С, С++ или Java не представляло особого труда. А поскольку и Хейльсбергу не нужно (да и нежелательно) было изобретать велосипед то он мог сосредоточиться непосредственно на усовершен­ство­ваниях и нововведениях в C#.



**Рисунок 1.1. Генеалогическое дерево C#**

Предком C# во втором поколении является С, от которого он унаследовал синтаксис, многие ключевые слова и операторы. Кроме того, C# построен на усовершенствованной объектной модели, определенной в С++. Если вы знаете С или С++, то будете чувствовать себя уютно и с языком C#.

Родственные связи C# и Java более сложные. Java также происходит от С и С++ и обладает общим с ними синтаксисом и объектной моделью. Как и Java, C# предназначен для получения переносимого кода, но C# не происходит непосредственно от Java. Напротив, C# и Java — это близкие, но не кровные родственники, имеющие общих предков, но во многом отличающиеся друг от друга. Впрочем, если вы знаете Java, то многие понятия C# окажутся вам знакомыми. С другой стороны, если вам в будущем придется изучать Java, то многие понятия, усвоенные в C#, могут быть легко распространены и на Java.

В C# имеется немало новых средств, но самое важное из них связано со встроенной поддержкой программных компонентов. В действительности C# может считаться компонентно-ориентированным языком программирования, поскольку в него внедрена встроенная поддержка написания программных компонентов. Например, в состав C# входят средства прямой поддержки таких составных частей программных компонентов, как свойства, методы и события. Но самой важной компонентно-ориентированной особенностью этого языка, вероятно, является возможность работы в безопасной среде многоязыкового программирования.

**Развитие C#**

С момента выпуска исходной версии 1.0 развитие C# происходило быстро. Вскоре после версии 1.0 корпорация Microsoft выпустила версию 1.1, в которую было внесено немало корректив, но мало значительных возможностей. Однако ситуация совершенно изменилась после выпуска версии C# 2.0.

Появление версии 2.0 стало поворотным моментом в истории развития C#, поскольку в нее было введено много новых средств, в том числе обобщения, частичные типы и анонимные методы, которые основательно расширили пределы возможностей и область применения этого языка, а также повысили его эффективность. После выпуска версии 2.0 "упрочилось" положение C#. Ее появление продемонстрировало также приверженность корпорации Microsoft к поддержке этого языка в долгосрочной перспективе.

Следующей значительной вехой в истории развития C# стал выпуск версии 3.0. В связи с внедрением многих новых свойств в версии C# 2.0 можно было ожидать некоторого замедления в развитии C#, поскольку программистам требовалось время для их освоения, но этого не произошло. С появлением версии 3.0 корпорация Microsoft внедрила ряд новшеств, совершенно изменивших общее представление о программировании. К числу этих новшеств относятся, среди прочего, лямбда-выражения, язык интегрированных запросов (LINQ), методы расширения и неявно типизированные переменные. Конечно, все эти новые возможности очень важны, поскольку они оказали заметное влияние на развитие данного языка, но среди них особенно выделяются две: язык интегрированных запросов (LINQ) [2, стр.524 - 601] и лямбда-выражения. Язык LINQ и лямбда-выражения вносят совершенно новый акцент в программирование на C# и еще глубже подчеркивают его ведущую роль в непрекращающейся эволюции языков программирования.

Текущей является версия C# 4.0. Эта версия прочно опирается на три предыдущие основные версии C#, дополняя их целым рядом новых средств. Вероятно, самыми важными среди них являются именованные и необязательные аргументы. В частности, именованные аргументы позволяют связывать аргумент с параметром по имени. А необязательные аргументы дают возможность указывать для параметра используемый по умолчанию аргумент. Еще одним важным новым средством является тип dynamic, применяемый для объявления объектов, которые проверяются на соответствие типов во время выполнения, а не компиляции. Кроме того, ковариантность и контравариантность параметров типа поддерживается благодаря новому применению ключевых слов in и out. Тем, кто пользуется моделью СОМ вообще и прикладными интерфейсами Office Automation API в частности, существенно упрощен доступ к этим средствам. В целом, новые средства, внедренные в версии C# 4.0, способствуют дальнейшей рационализации программирования и повышают практичность самого языка C#.

Еще два важных средства, внедренных в версии 4.0 и непосредственно связанных с программированием на C#, предоставляются не самим языком, а средой .NET Framework 4.0. Речь идет о поддержке параллельного программирования с помощью библиотеки распараллеливания задач (TPL) и параллельном варианте языка интегрированных запросов (PLINQ). Оба эти средства позволяют существенно усовершенствовать и упростить процесс создания программ, в которых применяется принцип параллелизма. И то и другое средство упрощает создание многопоточного кода, который масштабируется автоматически для использования нескольких процессоров, доступных на компьютере. В настоящее время широкое распространение получили компьютеры с многоядерными процессорами, и поэтому возможность распараллеливать выполнение кода среди всех доступных процессоров приобретает все большее значение практически для всех, кто программирует на C#.

**1.2.** [**Связь C# со средой .NET Framework**](https://professorweb.ru/my/csharp/charp_theory/level1/1_3.php)

C# спроектирован и разработан специально для применения с .NET Framework.

**Назначение .NET Framework —** служить средой для поддержки разработки и выполнения сильно распределенных компонентных приложений. Она обеспечивает совместное использование разных языков программирования, а также безопасность, переносимость программ и общую модель программирования для платформы Windows.

Базовые функциональные возможности платформы .NET включают в себя:

**Возможность обеспечения взаимодействия с существующим программным кодом**

Эта возможность, несомненно, является очень хорошей вещью, поскольку позволяет комбинировать существующие двоичные единицы СОМ (т.е. обеспечивать их взаимодействие) с более новыми двоичными единицами .NET и наоборот. С выходом версии .NET 4.0 эта возможность стала выглядеть даже еще проще, благодаря добавлению ключевого слова dynamic.

**Поддержка для многочисленных языков программирования**

Приложения .NET можно создавать с помощью любого множества языков программирования (C#, Visual Basic. F#. S# и т.д.). При этом в .NET код, написанный на любом языке компилируется в код на промежуточном языке (Intermediate Language - IL).

**Полная интеграция языков**

В .NET поддерживается межъязыковое наследование, межъязыковая обработка исключений и межъязыковая отладка кода. При этом .NET использует общий исполняющий механизм, основным аспектом которого является хорошо определенный набор типов, который способен понимать каждый, поддерживающий .NET язык.

**Усовершенствованная поддержка для создания динамических веб-страниц**

В .NET предлагается интегрированная поддержка для создания веб-страниц с помощью ASP.NET. В случае применения ASP.NET код создаваемых страниц поддается компиляции и может быть написан на любом поддерживающем .NET языке высокого уровня.

**Эффективный доступ к данным**

Набор компонентов .NET, известный под общим названием [ADO.NET](https://professorweb.ru/my/glossariy/glossy/files/102.php#a2), позволяет получать эффективный доступ к реляционным базам данных и многим другим источникам данных. Также предлагаются компоненты, позволяющие получать доступ к файловой системе и каталогам.

**Установка с нулевым воздействием**

Сборки бывают двух типов: разделяемые и приватные. Разделяемые сборки представляют собой обычные библиотеки, доступные всему программному обеспечению, а приватные сборки предназначены для использования только с определенными программами. Приватные сборки являются полностью самодостаточными, поэтому процесс их установки выглядит просто. Никакие записи в системный реестр не добавляются; все нужные файлы просто размещаются в соответствующей папке файловой системы.

Версия .NET Framework 4 является одним из серьезных выпусков данного продукта с множеством замечательных новых возможностей:

* **Динамическая типизация**

Знать статическим образом, какими объекты могут получаться в конце, не всегда возможно. Теперь можно предоставить возможность решать этот вопрос среде DLR (Dynamic Language Runtime — исполняющая среда динамического языка) непосредственно во время выполнения.

Среда DLR построена на основе среды [CLR](https://professorweb.ru/my/csharp/charp_theory/level1/1_4.php) (Common Language Runtime — общеязыковая исполняющая среда) для предоставления возможности связывать вместе все взаимодействие с динамическими языками.

Доступ к новой среде DLR в C# получается с помощью нового ключевого слова dynamic. Это ключевое слово служит флагом для компилятора; при каждой встрече с ним компилятор будет понимать, что речь идет о динамическом, а не обычном статическом вызове.

* **Ковариантность и контравариантность**

Возможности ковариантности (covariance) и контравариантности (contravariance) предлагались и в предыдущих версиях .NET Framework, но в предыдущих версиях .NET контравариантность можно было использовать с объектами и массивами, но нельзя, например, с обобщенными интерфейсами. В .NET 4 это стало возможным.

* **Технология ASP.NET MVC**

Технология ASP.NET MVC, ставшая последним серьезным добавлением в ASP.NET, вызвала большую шумиху в сообществе разработчиков. Она предоставляет средства для создания приложений ASP.NET с использованием шаблона "модель-представление-контроллер" (Model-View-Controller), которых давно ожидали многие разработчики. ASP.NET MVC обеспечивает разработчиков возможностями тестирования, гибкости и обслуживания создаваемых ими приложений [15, стр.211]. Важно понимать, что ASP.NET MVC не предназначена служить заменой хорошо известной технологии ASP.NET, а является просто другим способом построения приложений.

**1.3. Роль платформы .NET и типы его приложений**

Язык C# можно использовать для создания консольных приложений — текстовых приложений, запускаемых в окне DOS. Однако гораздо чаще язык C# придется использовать для создания приложений, имеющих доступ к множеству технологий, связанных с .NET.

**Создание приложений ASP.NET**

Изначально появление ASP.NET 1.0 фундаментально изменило модель веб-программирования. ASP.NET 4 является старшим выпуском продукта и в котором реализованы основные революционные шаги, направленные на повышение продуктивности работы.

**Возможности ASP.NET**

Первое, и, возможно, самое важное — это то, что страницы структурированы. То есть каждая страница — это, по сути, класс, унаследованный от класса .NET System.Web.UI.Page.

Другая приятная особенность страниц ASP.NET заключается в том, что их можно создавать в Visual Studio 2022 — той же среде, в которой программируется бизнес-логика и компоненты доступа к данным, используемые этими же страницами ASP.NET. Проект Visual Studio 2022, или решение (solution), содержит все файлы, ассоциированные с приложением.

Последнее, но не менее важное свойство ASP.NET, которое следует упомянуть — это увеличенная производительность. В то время как классические ASP-страницы интерпретировались при каждом обращении к странице, страницы ASP.NET после компиляции подвергаются [кэшированию](https://professorweb.ru/my/glossariy/glossy/files/100.php#k2) на веб-сервере. Это значит, что все последующие запросы страниц ASP.NET выполняются быстрее, чем первый.

**Веб-формы**

В Visual Studio 2022 предлагается библиотека Web Forms (Веб-формы). Это средство позволяет графически создавать страницы ASP.NET. Здесь потребуется перетащить необходимые элементы управления из панели инструментов на поверхность формы, затем слегка подкорректировать код формы и написать обработчики событий для элементов управления.

**Элементы управления веб-сервера**

Элементы управления, применяемые в веб - формах представляют собой XML-дескрипторы, принадлежащие пространству имен ASP.NET, которые веб-браузер динамически трансформирует в HTML-дескрипторы и сценарии клиентской стороны, когда осуществляется запрос этой страницы. Веб-сервер может отображать одни и те же элементы управления серверной стороны различными способами, генерируя трансформацию, соответствующую конкретному веб-браузеру, который прислал запрос.

**Веб-службы XML**

На сегодняшний день HTML-страницы составляют большую часть трафика World Wide Web. Однако благодаря XML, компьютеры получают независимый от устройства формат, который может использоваться для их взаимодействия в Интернете. Веб-службы XML предназначены для веб-среды, ориентированной на службы, когда удаленные компьютеры предоставляют друг другу динамическую информацию, которая может быть проанализирована и переформатирована, прежде чем получить свое окончательное представление для пользователя.

**Создание Windows-форм**

Хотя C# и .NET предназначены, в частности, для веб-разработки, они по-прежнему поддерживают так называемые приложения "толстого" клиента — т.е. приложения, которые должны устанавливаться на машине конечного пользователя, где и выполняется большая часть обработки. Это обеспечивает библиотека Windows forms (Windows-формы).

**Службы Windows**

*Служба Windows* — это программа, предназначенная для запуска в фоновом режиме в среде Windows NT/2000/XP/2003/ Vista/7 (но не Windows 9х). Эти службы удобны, когда нужно организовать непрерывную работу программ, готовых реагировать на события, без необходимости явного их запуска по инициативе пользователя. Хорошим примером является служба World Wide Web Service на веб-серверах, которая прослушивает веб-запросы клиентов.

**Windows Communication Foundation (WCF)**

Изучая способы перемещения данных и служб из одной точки в другую, используемые в технологиях Microsoft, мы можем обнаружить, что в нашем распоряжении оказывается широкое разнообразие возможных решений. Например, мы можем использовать веб-службы ASP.NET, технологию .NET Remoting, Enterprise Services либо MSMQ.

Microsoft собрали все эти технологии вместе в .NET Framework, как единый способ перемещения данных -Windows Communication Foundation (WCF). Библиотека WCF обеспечивает возможностью первоначального построения службы с последующей доставкой этой службы различными путями (даже по разным протоколам) простым изменением конфигурационного файла.

[ГЛАВА 2.](#_Toc41567247) ОСНОВЫ ASP.NET, ASP.NET MVC И ВИДЫ СУБД

**2.1.Основы и** [**особенности ASP.NET**](https://professorweb.ru/my/ASP_NET/base/level1/1_1.php)

ASP.NET Core представляет технологию для создания веб-приложений на платформе .NET, развиваемую компанией Microsoft. В качестве языков программирования для разработки приложений на ASP.NET Core используются C# и F#.

История ASP.NET фактически началась с выходом первой версии .NET в начале 2002 года и с тех пор ASP.NET и .NET развивались параллельно: выход новой версии .NET знаменовал выход новой версии ASP.NET, поскольку ASP.NET работает поверх .NET.

Сегодня ASP.NET пользуется небывалой популярностью, но также появилось и еще как минимум одно новое направление, составляющее конкуренцию традиционному программированию с использованием ASP.NET, которое получило название ASP.NET MVC.

**ASP.NET интегрируется с .NET Framework**

Платформа .NET Framework делится на практически неповторимый ряд функциональных частей с десятками тысяч типов (в .NET так называются классы, структуры, интерфейсы и другие ключевые элементы программирования). Прежде чем пытаться программировать любое приложение .NET, необходимо сначала получить хотя бы общее представление об этих частях и о том, почему они организованы именно таким, а не каким-то другим образом.

То, как организована предлагаемая в .NET Framework обширная коллекция функциональности, программистам традиционных Windows-приложений, несомненно, покажется замечательным улучшением. Каждый из тысяч доступных в .NET Framework классов размещен в логическом иерархическом контейнере, который называется пространством имен (namespace).

В разных пространствах имен предоставляется разная функциональность, но все вместе они предлагают функциональные возможности для практически каждого аспекта области распределенной разработки, начиная с организации очередей сообщений и заканчивая обеспечением безопасности. Весь целиком этот обширный набор инструментов называется библиотекой классов (class library).

Интересно то, что способ, которым классы .NET Framework можно использовать в ASP.NET, ничем не отличается от того, которым они применяются в приложениях .NET любого другого типа (в том числе автономных Windows-приложениях, Windows-службах, утилитах командной строки и т.д.).

Хотя в .NET предлагаются ориентированные специально на Windows- и на веб-приложения классы для построения пользовательских интерфейсов, большинство возможностей .NET Framework (начиная с получения доступа к базам данных и закачивания поддержкой многопоточной обработки) допускается использовать в приложениях любого типа. Другими словами, в .NET разработчикам веб-приложений предлагаются те же самые инструменты, что и разработчикам многофункциональных клиентских приложений.

**Код ASP.NET компилируется, а не интерпретируется**

Подобно всем приложениям .NET, приложения ASP.NET всегда компилируются. На самом деле выполнение кода на C# или Visual Basic без предварительной компиляции просто невозможно [9, стр. 26-35].

Приложения ASP.NET в действительности проходят через два этапа компиляции.

На первом этапе написанный код на C# компилируется в код на промежуточном языке, который называется MSIL (Microsoft Intermediate Language — промежуточный язык Microsoft), или просто IL. Этот первый этап как раз и является одной из главных причин, по которым в .NET могут использоваться самые разные языки.

Дело в том, что все языки .NET (в том числе C#, Visual Basic и многие другие), по сути, компилируются в практически идентичный код IL. Этот первый этап компиляции может происходить как автоматически при первом запросе страницы, так и выполняться заранее. Скомпилированный файл с кодом на IL представляет собой сборку [13, стр. 209].

Второй этап компиляции происходит непосредственно перед фактическим выполнением страницы. На этом этапе код IL компилируется в код на низкоуровневом машинном языке. Называется этот этап оперативной (Just-In-Time — JIT) компиляцией и выглядит одинаково для всех приложений .NET (включая, например, Windows-приложения).

На следующем рисунке показан этот состоящий из двух этапов процесс компиляции:



Механизм выполнения .NET

**Рисунок 2.1.1. Процесс компиляции в .NET**

Процесс компиляции в .NET разделен на два этапа для предоставления разработчикам как можно большего удобства и мобильности. Перед тем, как создавать код на низкоуровневом машинном языке, компилятор должен знать, в какой операционной системе и на каком базовом оборудовании будет выполняться приложение (например, будет это 32- или 64-разрядная ОС Windows). Благодаря выполнению таких двух этапов компиляции, можно создавать скомпилированную сборку с кодом .NET и по-прежнему распространять ее среди более, чем одной, платформы.

Разумеется, этап оперативной компиляции, пожалуй, не был бы столь полезным, если бы его нужно было выполнять при каждом запросе той или иной страницы веб-сайта. К счастью, приложения ASP.NET не нуждаются в выполнении компиляции при каждом запрашивании веб-страницы. Вместо этого код на IL в них создается один раз и генерируется заново только в случае изменения исходного кода, а файлы, содержащие машинный код, кэшируются в системном каталоге, путь к которому выглядит примерно так:



**Рисунок 2.1.2. Путь в системном каталоге**

Конкретный момент, когда код компилируется в IL [16, стр.113-114], зависит от того, каким образом создается и развертывается веб-приложение. В случае создания веб-приложения в виде веб-проекта в Visual Studio код компилируется в IL при компиляции проекта. В случае его создания в виде облегченного веб-сайта без проекта, код для каждой из его страниц компилируется при первом запросе соответствующей страницы. Но и в том, и в другом случае через второй этап компиляции (из IL в машинный код) код проходит лишь при первом его запуске.

В ASP.NET также включены инструменты для выполнения предварительной компиляции, с помощью которых можно делать так, чтобы приложение компилировалось сразу же в машинный код прямо после его развертывания на производственном веб-сервере. Это позволяет избежать накладных расходов, связанных с выполнением первого этапа компиляции при развертывании готового приложения (и исключить возможность подделки или изменения кода другими людьми).

**Модели разработки**

ASP.NET Core позволяет создавать веб-приложений с помощью различных моделей разработки.

* Прежде всего это **базовый ASP.NET Core**, который поддерживает все основные моменты, необходимые для работы современного веб-приложения: маршрутизация, конфигурация, логгирования, возможность работы с различными системами баз данных и т.д.
* **ASP.NET Core MVC** представляет в общем виде построения приложения вокруг трех основных компонентов - Model (модели), View (представления) и Controller (контроллеры), где модели отвечают за работу с данными, контроллеры представляют логику обработки запросов, а представления определяют визуальную составляющую.
* **Razor Pages** представляет модель, при котором за обработку запроса отвечают специальные сущности - страницы Razor Pages. Каждую отдельную такую сущность можно ассоциировать с отдельной веб-страницей [13, стр. 209-264].
* **ASP.NET Core Web API** представляет реализацию паттерна REST, при котором для каждого типа http-запроса (GET, POST, PUT, DELETE) предназначен отдельный ресурс. Подобные ресурсы определяются в виде методов контроллера Web API.
* **Blazor** -фреймворк, который позволяет создавать интерактивные приложения как на стороне сервера, так и на стороне клиента и позволяет задействовать на уровне браузера низкоуровневый код WebAssembly.

**2.2. Фреймворк ASP.NET Core MVC и его преимущества**

Фреймворк ASP.NET Core MVC является частью платформы ASP.NET Core, его отличительная особенность - применение паттерна MVC. Преимуществом использования фрейморка ASP.NET Core MVC по сравнению с "чистым" ASP.NET Core является то, что он упрощает в ряде ситуаций и сценариев организацию и создание приложений, особенно это относится к большим приложениям.

Стоит отметить, чам паттерн MVC не является исключительной особенностью ASP.NET Core MVC, данный паттерн появился еще в конце 1970-х годов в компании Xerox как способ организации компонентов в графическом приложение на языке Smalltalk и в настоящее время применяется во многих платформах и для различных языках программирования. Особенно популярен паттерн MVC в веб-приложениях.

Концепция паттерна MVC предполагает разделение приложения на три компонента:

Модель (model): описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными, например, логику валида­ции данных. Как правило, объекты моделей хранятся в базе данных.

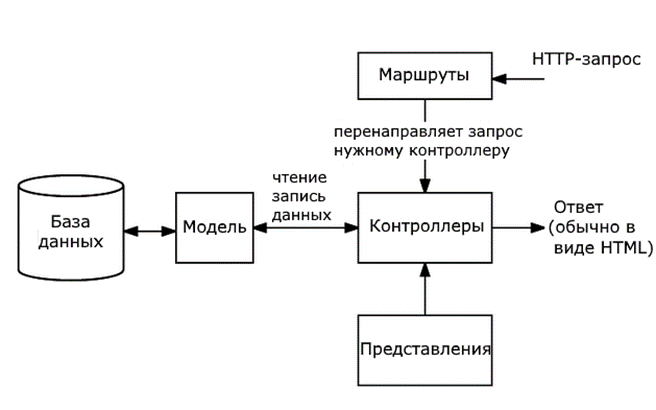
В MVC модели представлены двумя основными типами: модели представлений, которые используются представлениями для отображения и передачи данных, и модели домена, которые описывают логику управления данными.

Модель может содержать данные, хранить логику управления этими данными. В то же время модель не должна содержать логику взаимодействия с пользователем и не должна определять механизм обработки запроса. Кроме того, модель не должна содержать логику отображения данных в представлении.

Представление (view): отвечают за визуальную часть или пользователь­ский интерфейс, нередко html-страница, через который пользователь взаимодействует с приложением. Также представление может содержать логику, связанную с отображением данных. В то же время представление не должно содержать логику обработки запроса пользователя или управления данными.

Контроллер (controller): представляет центральный компонент MVC, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления, наполненного данными моделей.

Отношения между компонентами паттерна можно описать следующей схемой:



**Рисунок 2.2.1Модель MVC**

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления никак не влияют на модель. Контроллер и представление являются относительно независимыми компонентами. Так, из представления можно обращаться к определенному контроллеру, а из контроллера генерировать представления, но при этом нередко их можно изменять независимо друг от друга [10, стр. 72].

Такое разграничение компонентов приложения позволяет реализовать концепцию разделение ответственности, при которой каждый компонент отвечает за свою строго очерченную сферу. В связи с чем легче построить работу над отдельными компонентами. И благодаря этому приложение легче разрабатывать, поддерживать и тестировать отдельные компоненты. Допустим, если нам важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер..

**Архитектурный шаблон MVC**

Важно различать архитектурный шаблон MVC и инфраструктуру ASP.NET MVC Framework. Шаблон MVC далеко не нов (его появление датируется 1978 г. и связано с проектом Smalltalk в Xerox PARC), но в наши дни он завоевал огромную популярность в качестве шаблона для веб-приложений по перечисленным ниже причинам:

* Взаимодействие пользователя с приложением MVC осуществляется в соответствии с естественным циклом: пользователь предпринимает действие, в ответ на которое приложение изменяет свою модель данных и доставляет обновленное представление пользователю. Затем цикл повторяется. Это хорошо укладывается в схему веб-приложений, предоставляемых в виде последовательностей запросов и ответов HTTP [15, стр.29].
* Веб-приложения, нуждающиеся в комбинировании нескольких технологий (например, баз данных, HTML-разметки и исполняемого кода), обычно разделяются на ряд слоев или уровней. Полученные в результате шаблоны естественным образом вписываются в концепции MVC.

Инфраструктура ASP.NET MVC Framework реализует шаблон MVC и при этом обеспечивает существенно улучшенное разделение ответственности. На самом деле в ASP.NET MVC внедрен современный вариант MVC, который особенно хорошо подходит для веб-приложений.

За счет принятия и адаптации шаблона MVC инфраструктура ASP.NET MVC Framework составляет сильную конкуренцию Ruby on Rails и аналогичным платформам, выводя модель MVC в авангард развития мира .NET. Обобщая опыт и наиболее рекомендуемые приемы, обнаруженные разработчиками, которые используют другие платформы, ASP.NET MVC во многих отношениях превзошла даже то, что может предложить Rails.

**Расширяемость**

Инфраструктура MVC Framework построена в виде набора независимых компонентов, которые удовлетворяют интерфейсу .NET или созданы на основе абстрактного базового класса. Компоненты, подобные системе маршрутизации, механизму визуализации и фабрике контроллеров, можно легко заменять другими компонентами с собственной реализацией. В общем случае для каждого компонента MVC Framework предлагает три возможности:

* Использование стандартной реализации компонента в том виде, как она есть (этого должно быть достаточно для большинства приложений).
* Создание подкласса из стандартной реализации с целью корректировки существующего поведения.
* Полная замена компонента новой реализацией интерфейса или абстрактного базового класса.

Различные компоненты, а также способы и причины их возможной настройки или замены будут рассматриваться в следующих статьях.

**Жесткий контроль над HTML и HTTP**

Инфраструктура ASP.NET MVC генерирует ясный и соответствующий стандартам код разметки. Ее встроенные вспомогательные методы HTML производят соответствующий стандартам вывод, но существует также гораздо более значимое философское изменение по сравнению с Web Forms. Вместо генерации громадного объема трудно поддающейся управлению HTML-разметки инфраструктура MVC Framework стимулирует создание простых и элегантных элементов, оформленных стилями CSS.

Конечно, если действительно требуется использовать некоторые готовые виджеты для таких сложных элементов пользовательского интерфейса, как окна выбора даты или каскадные меню, применяемый в ASP.NET MVC подход "никаких специальных требований" к разметке, позволяет легко использовать наилучшие библиотеки для построения пользовательских интерфейсов, подобные jQuery или Bootstrap CSS. К примеру, библиотека jQuery настолько эффективно поддерживается, что поставляется в качестве встроенной части стандартного шаблона проекта ASP.NET MVC в Visual Studio наряду с другими популярными библиотеками, такими как Bootstrap, Knockout и Modernizr.

Сгенерированные ASP.NET MVC страницы не содержат никаких данных View State, поэтому они меньше типовых страниц ASP.NET Web Forms по размеру. Несмотря на современные быстрые соединения, такая экономия трафика по-прежнему повышает комфорт конечного пользователя и помогает сократить затраты, связанные с запуском популярных веб-приложений.

Инфраструктура ASP.NET MVC работает в тесном сотрудничестве с HTTP. При этом имеется контроль над запросами, передаваемыми между браузером и сервером, что позволяет очень точно настраивать пользовательский интерфейс по своему усмотрению. Технология AJAX проста, и ей не нужны какие-то автоматические обратные отправки запросов для взаимодействия с состоянием клиентской стороны.

**Тестируемость**

Естественное разделение различных ответственностей приложения по независимым друг от друга частям программного обеспечения, которое поддерживается архитектурой MVC, позволяет изначально строить легко сопровождаемые и тестируемые приложения. Однако проектировщики ASP.NET MVC на этом не остановились. Для каждого фрагмента компонентно-ориентированного проекта инфраструктуры они обеспечили структурированность, необходимую для удовлетворения требований модульного тестирования и средств имитации.

В среду Visual Studio добавлен набор мастеров для автоматизированного создания проектов модульного тестирования, которые могут быть интегрированы с такими инструментами модульного тестирования с открытым кодом, как NUnit и xUnit. Даже если вам никогда ранее не приходилось создавать модульные тесты, у вас будет все необходимое для успешного старта.

Далее, в этом руководстве, вы увидите примеры написания ясных и простых модульных тестов для контроллеров и действий ASP.NET MVC, предоставляющих фиктивные либо имитированные реализации компонентов инфраструктуры для эмуляции любого сценария с использованием разнообразных стратегий тестирования и имитации.

Тестируемость касается не только модульного тестирования. Приложения ASP.NET MVC успешно работают также с инструментами тестирования, встроенными в средства автоматизации пользовательского интерфейса. Можно создавать тестовые сценарии, которые имитируют взаимодействие с пользователем, не беспокоясь о том, какие структуры HTML-элементов, классы CSS или идентификаторы будут сгенерированы инфраструктурой, равно как и о неожиданных изменениях структуры.

**Мощная система маршрутизации**

Совершенствование технологии построения веб-приложений сопровождалось развитием стиля URL-адресов. Адреса URL, подобные приведенному ниже:



**Рисунок 2.2.2. URL адрес**

встречаются все реже, а на смену им приходит более простой и понятный формат следующего вида:



**Рисунок 2.2.3. Понятный формат URL адреса**

Существует ряд веских причин для того, чтобы заботиться о структуре URL-адресов.

Во-первых, поисковые механизмы придают ключевым словам, содержащимся в URL, больший вес.

Во-вторых, многие веб-пользователи достаточно сообразительны, чтобы понять URL, и ценят возможность осуществления навигации путем ввода запроса в адресной строке своего браузера.

В-третьих, когда кто-то понимает структуру URL-адреса, они с большей вероятностью пройдут по нему, поделятся им с другими или даже зачитают его вслух по телефону.

В-четвертых, при таком подходе в Интернете не раскрываются технические нюансы, структура каталогов и имен файлов приложения; следовательно, вы вольны изменять лежащую в основе сайта реализацию, не нарушая работоспособности всех входящих ссылок.

В более ранних инфраструктурах понятные URL-адреса реализовать было трудно, но в ASP.NET MVC применяется средство, известное как маршрутизация URL, которое обеспечивает предоставление понятных URL-адресов по умолчанию. Это дает контроль над схемой URL и ее взаимосвязью с приложением, обеспечивая свободу создания понятного и удобного для пользователей шаблона URL без необходимости следования какому-то заранее определенному шаблону. И, разумеется, это означает также простоту определения современной схемы URL в стиле REST, если это требуется.

**Построение на основе лучших частей платформы ASP.NET**

Существующая платформа ASP.NET производства Microsoft предлагает зрелый, хорошо проверенный набор компонентов и средств для разработки эффективных и высокопроизводительных веб-приложений. Первое и наиболее очевидное преимущество заключается в том, что поскольку инфраструктура ASP.NET MVC построена на основе платформы .NET, вы вольны писать код на любом языке .NET и при этом иметь доступ к одним и тем же функциям API-интерфейсов, которые определены не только в MVC Framework, но и в обширной библиотеке классов .NET, а также в широком множестве библиотек .NET от независимых разработчиков.

Во-вторых, готовые средства платформы ASP.NET, такие как аутентификация, членство, роли, профили и интернационализация, могут существенно сократить объем кода, который придется писать и поддерживать в любом веб-приложении, и в проекте MVC Framework они столь же эффективны, как в классическом проекте Web Forms. Лежащая в основе платформа ASP.NET предоставляет развитый набор инструментов, на базе которых строятся веб-приложения с помощью MVC Framework.

**Современный API-интерфейс**

Платформа Microsoft .NET развивалась с каждым крупным выпуском, поддерживая - и даже определяя - многие передовые аспекты современного программирования.

Версия ASP.NET MVC 5 построена для .NET Framework 4.5.1, поэтому ее API-интерфейс может в полной мере задействовать последние новшества языка и исполняющей среды, в том числе ключевое слово await, расширяющие методы, лямбда-выражения, анонимные и динамические типы, а также язык интегрированных запросов (Language Integrated Query - LINQ). Многие методы и шаблоны кодирования API-интерфейса MVC Framework следуют более четкой и выразительной композиции, чем это было возможно в ранних платформах.

**Инфраструктура ASP.NET MVC имеет открытый код**

В отличие от предшествующих платформ веб-разработки производства Microsoft, первоначальный исходный код ASP.NET MVC доступен для свободной загрузки и даже для модификации и компиляции с целью получения собственной версии этой инфраструктуры [18, стр.573]. Это буквально неоценимо при отладке кода, обращающегося к системному компоненту, когда требуется пошагово выполнить его код (и даже ознакомиться с комментариями программистов, написавших этот код). Это также полезно, если вы создаете усовершенствованный компонент и хотите видеть, какие существуют возможности разработки, или узнать, как действительно работают встроенные компоненты.

Вдобавок упомянутая возможность удобна и тогда, когда не устраивает работа того или иного компонента, когда требуется найти ошибку или когда необходимо получить доступ к тому, что недоступно с помощью других средств - интересующий компонент можно просто изменить самостоятельно. Однако при этом придется отслеживать свои изменения и повторно их применять при модернизации до более новой версии инфраструктуры.

Исходный код ASP.NET MVC распространяется в соответствии с открытой лицензией Microsoft (Microsoft Public License - Ms-PL) - лицензией программного обеспечения с открытым кодом, одобренной Инициативой по поддержке ПО с открытым кодом (Open Source Initiative - OSI). Это значит, что исходный код можно изменять, развертывать и даже распространять в виде производного проекта.

**2.3. СУБД и их виды**

В рамках данной диссертационной работы нам помимо разработки системы, которой мы будем оперировать, понядобятся данные которые будут в ней содержаться, храниться и изменяться. Здесь у нас появляется потребность в использовании БД, ибо для правильной работы сайта нужны не только файлы с кодом страниц, но и базы данных. Для взаимодействия с БД используются системы управления базами данных (СУБД). В последующей главе мы побробно рассмотрим применение конкретной СУБД в системе, но прежде чем приступить к её реализации дадим объяснение СУБД и подробно расскажем об их разновидностях.

**Система управления базами данных (СУБД)**

Система управления базами данных (сокращенно**СУБД**)– это программное обеспечение для создания и работы с базами данных.

**Главная функция СУБД** – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, а также отвечает за копирование и восстановление информации после каких-либо сбоев.

**Для каких целей подходит СУБД**

СУБД позволяет полностью контролировать работу баз данных, в которых содержится детальная информация о клиентах компании. Например, это могут быть совершенные транзакции в интернет-магазине. Чтобы создать базу данных для такой крупной системы, необходимо организовать таблицу с различными значениями. Чаще всего в них входит: название продукции, ее стоимость, количество и многое другое. Для поддержания таких крупных таблиц как раз используется система управления базами данных. Она обеспечивает надежность и высокую скорость работы БД.

Помимо работы с крупными данными, СУБД также помогает:

* модифицировать базу данных – удалять, изменять или объединять данные внутри системы;
* восстанавливать утерянную информацию после непредвиденных сбоев;
* настраивать резервное копирование данных;
* получать все необходимые данные из БД через язык запросов SQL;
* администрировать систему – выдавать ограниченные права доступа для пользователей.

**Из чего состоят системы управления базами данных**

СУБД состоит из нескольких компонентов:

* **Язык запросов**– используется для создания запросов к БД, например, SQL.
* **Ядро СУБД**– основной компонент, который обеспечивает выполнение запросов и доступ к данным.
* **Драйверы**– программное обеспечение, необходимое для обеспечения взаимодействия между СУБД и приложениями, например, ODBC.
* **Административная консоль**– графический интерфейс для управления БД, включая создание таблиц, пользователей и управление правами доступа.
* **Библиотеки**– наборы программных модулей, которые можно использовать для создания приложений, использующих базу данных.
* **Хранилище данных**– физическое устройство или набор устройств, где содержится вся информация, например, жесткий диск.

Рассмотренные выше компоненты работают вместе для обеспечения эффективного управления базами данных.

**Основные виды системы управления базами данных**

Системы управления БД принято разделять на несколько классификаций. Ниже мы рассмотрим каждый из видов и разберемся, чем они отличаются друг от друга.

**По расположению баз данных**

В этом виде СУБД разделяются на:

* **Локальные**– установлены на локальном компьютере и работают только на нем.
* **Централизованные**– находятся на центральном сервере и обслуживают клиентские приложения, установленные на других ПК.
* **Распределенные**– состоят из нескольких серверов, расположенных в разных местах, и обрабатывают данные, которые были разделены между устройствами.

**По хранению и обработке данных и запросов**

Здесь также СУБД делятся на несколько подвидов:

* **Клиент-серверные**– разделяют работу между клиентской и серверной частями. Первая отвечает за пользовательский интерфейс и запросы к серверу, вторая – за хранение и обработку данных.
* **Файл-серверные**– хранят базу данных на центральном сервере, но обработка запросов происходит на устройствах пользователей через сетевые пути.
* **Встраиваемые**– предназначены для использования внутри приложений. Обеспечивают быстрое и экономичное хранение данных, не требуют установки и настройки на компьютере клиента.

**По языку запросов**

Системы управления БД также классифицируются по языку запросов:

* **SQL-ориентированные** – используют язык SQL для запросов к базе данных.
* **NoSQL-ориентированные** – используют различные языки запросов, не связанные с SQL, такие, как MongoDB Query Language или Cassandra Query Language.

**По структуре и организации данных**

Последняя классификация – в ней системы управления базами данных делятся на то, как они представляют информацию внутри БД.

* **Реляционные**– содержат информацию в виде таблиц, которые могут быть связаны между собой. У каждой строки есть уникальный идентификатор, помогающий легко находить нужные данные. Реляционные СУБД используются, например, в MySQL и PostgreSQL.
* **Ключ-значение**– разновидность NoSQL-ориентированных СУБД, которые используют пары ключ-значение для хранения данных. Такая организация данных встречается в ПО Redis и Memcached.
* **Документные**– еще одна разновидность NoSQL. Она использует документы, содержащие различные поля и их значения для хранения данных. Увидеть такую классификацию можно в Amazon DocumentDB и CouchDB.
* **Графовые**– используют граф для хранения и организации данных, где каждый узел представляет собой объект, а ребра между узлами – отношения между объектами. Такие СУБД позволяют выполнять всевозможные запросы, которые не могут быть исполнены в реляционных системах. Пример: Amazon Neptune, Neo4j, InfoGrid.
* **Колоночные**– разновидность реляционных СУБД, которые хранят данные в виде колонок, а не строк. Каждая колонка содержит информацию только одного типа, что позволяет сэкономить размер БД и ускорить выполнение запросов. Примеры таких СУБД: Vertica и ClickHouse.

**Реляционные СУБД и язык SQL**

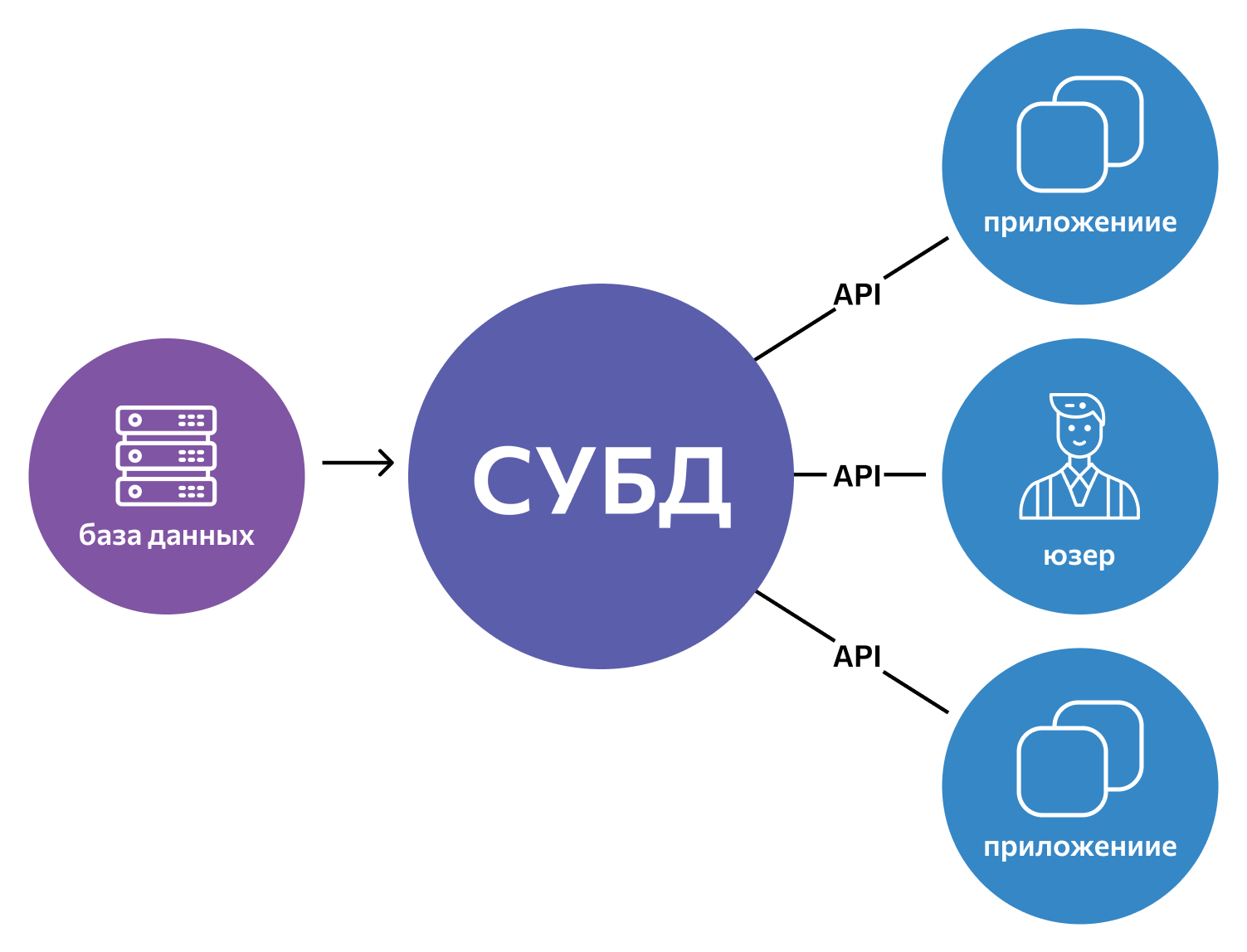
**Реляционные и объектно-реляционные СУБД**являются одними из самых распространенных систем. Они представляют собой таблицы, в которых каждый столбец (он называется «field» или «поле») упорядочен и имеет определенное уникальное название. Последовательность строк (их называют «records» или «записи») определяется последовательностью ввода информации в таблицу. При этом обрабатывание столбцов и строк может происходить в любом порядке. Таблицы с данными связаны между собой специальными отношениями, благодаря чему с данными из разных таблиц можно работать – к примеру, объединять их при помощи одного запроса.

Для управления реляционными базами данных применяется особый язык программирования–**SQL.** Сокращение расшифровывается как «**Structured query language**», в переводе на русский – «язык структурированных запросов» [14, стр.35].

Команды, которые используются в SQL, делятся на:

* манипулирующие данными,
* определяющие данные,
* управляющие данными.

Схема работы с базой данных выглядит следующим образом**:**

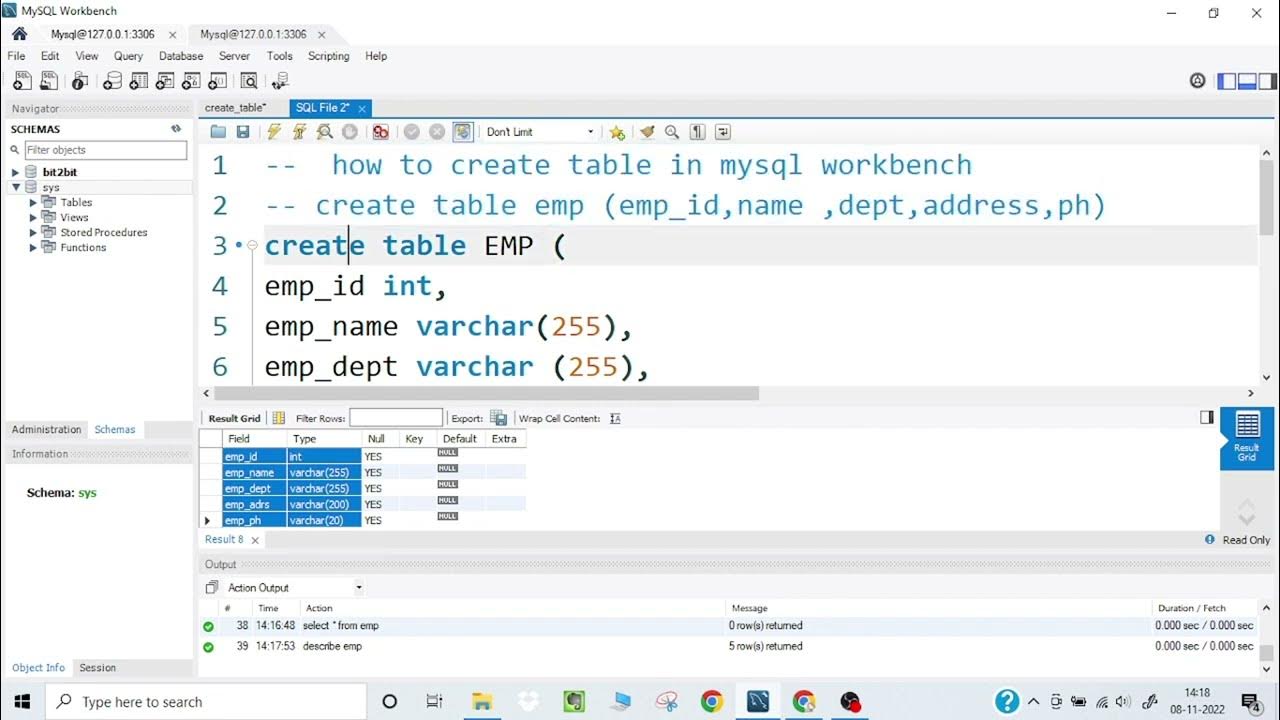


**Рисунок 2.3.1.Схема работы СУБД с базой данных**

**Примеры современных СУБД**

Далее, я кратко расскажу о лучших СУБД, которые чаще всего используются при создании веб-проектов.

**MySQL**

****

**Рисунок 2.3.2. MySql Workbench**

MySQL является одной из самых популярных и распространенных СУБД, которая используется во многих компаниях (например, Facebook, Wikipedia, Twitter, LinkedIn, Alibaba и других). MySQL представляет собой реляционную СУБД, которая относится к свободному программному обеспечению: она распространяется на условиях GNU Public License. Как правило, эту систему управления базами данных определяют как хорошую, быструю и гибкую, рекомендованную к применению в небольших или средних проектах.

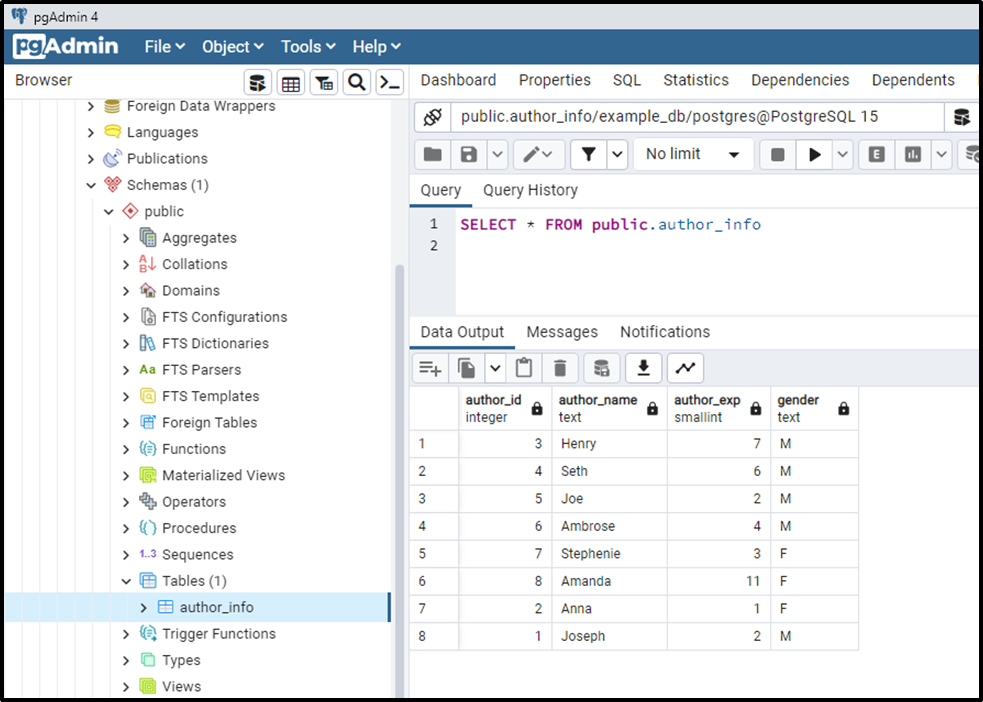
У MySQL есть множество различных преимуществ. Например, она поддерживает различные типы таблиц – как известные MyISAM и InnoDB, так и более экзотичные HEAP и MERGE. Кроме того, количество поддерживаемых типов постоянно растет. MySQL выполняет все команды быстро – возможно, сейчас это самая быстрая СУБД из всех существующих. С этой системой управления базами данных может одновременно работать неограниченное количество пользователей, а число строк в таблицах может достигать 50 миллионов.

Так как в сравнении с некоторыми другими системами MySQL поддерживает меньшее количество возможностей, то и работать с ней значительно проще, чем, к примеру, с PostgreSQL, о которой будет рассказано ниже.

Для работы с MySQL используется не только текстовый, но и графический режим. Это становится реальным благодаря приложению phpMyAdmin: для работы в приложении вам даже не нужно знать SQL-команды, а администрировать свою базу данных можно прямо через браузер.

MySQL – это выбор тех, кому необходима СУБД для проекта небольшого или среднего размера, быстрая и удобная в работе и без сложностей с администрированием.

**PostgreSQL**



**Рисунок 2.3.3. Интерфейс PgAdmin4**

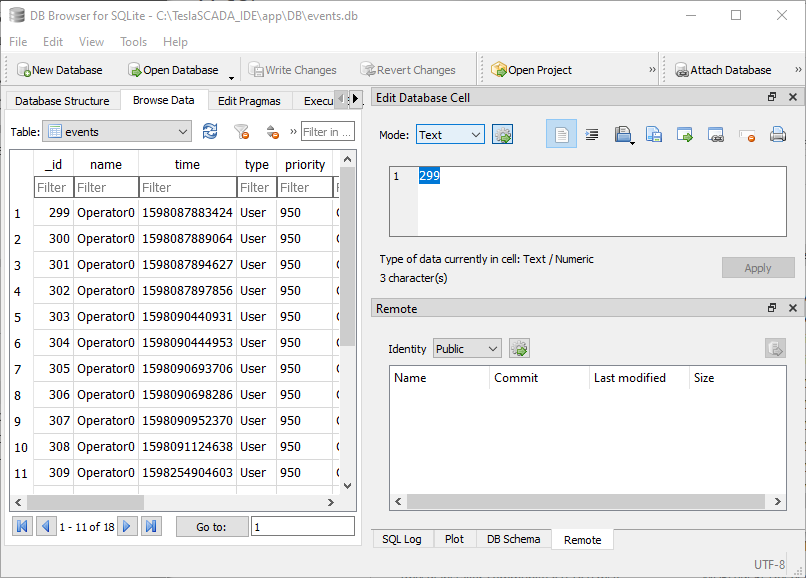
Эта свободно распространяемая система управления базами данных относится к объектно-реляционному типу СУБД. Как и в случае с MySQL, работа с PostgreSQL основывается на языке SQL, однако, в отличие от MySQL, PostgreSQL поддерживает стандарт SQL - 2011. Эта СУБД не имеет ограничений ни по максимальному размеру базы данных, ни по максимуму записей или индексов в таблице.

Если говорить о преимуществах PostgreSQL, то в первую очередь это надежность транзакций и репликаций, возможность наследования и легкая расширяемость. PostgreSQL поддерживает различные расширения и варианты языков программирования, такие как PL/Perl, PL/Python и PL/Java. Также есть возможность загружать C-совместимые модули [2 , стр. 9-24].

Многие отмечают, что в отличие от MySQL данная СУБД имеет хорошую и подробную документацию, которая дает ответы практически на все вопросы.

О том, что это более масштабная, чем MySQL, СУБД, говорит и тот факт, что PostgreSQL периодически сравнивают с такой мощной системой управления данных, как Oracle. Все это позволяет говорить о PostgreSQL как об одной из самых продвинутых СУБД на данный момент.

**SQLite**



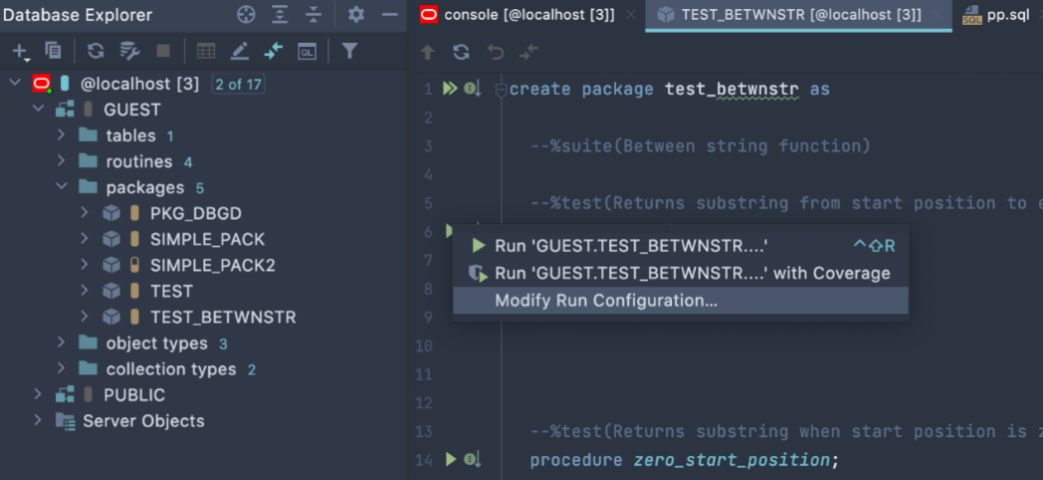
**Рисунок 2.3.4. Интерфейс браузера баз данных для SQLite**

На данный момент это одна из самых компактных СУБД. Также она является встраиваемой и реляционной.

SQLite позволяет хранить все данные в одном файле и, благодаря своему небольшому объему, отличается завидным быстродействием. SQLite значительно отличается от MySQL и PostgreSQL своей структурой: движок и интерфейс этой СУБД находятся в одной библиотеке – и именно это позволяет выполнять все запросы очень быстро. Другие СУБД (MySQL, PostgreSQL, Oracle и т.д.) используют парадигму «клиент-сервер», когда взаимодействие происходит через сетевой протокол.

Из недостатков можно отметить отсутствие системы пользователей и возможности увеличения производительности.

**Oracle**



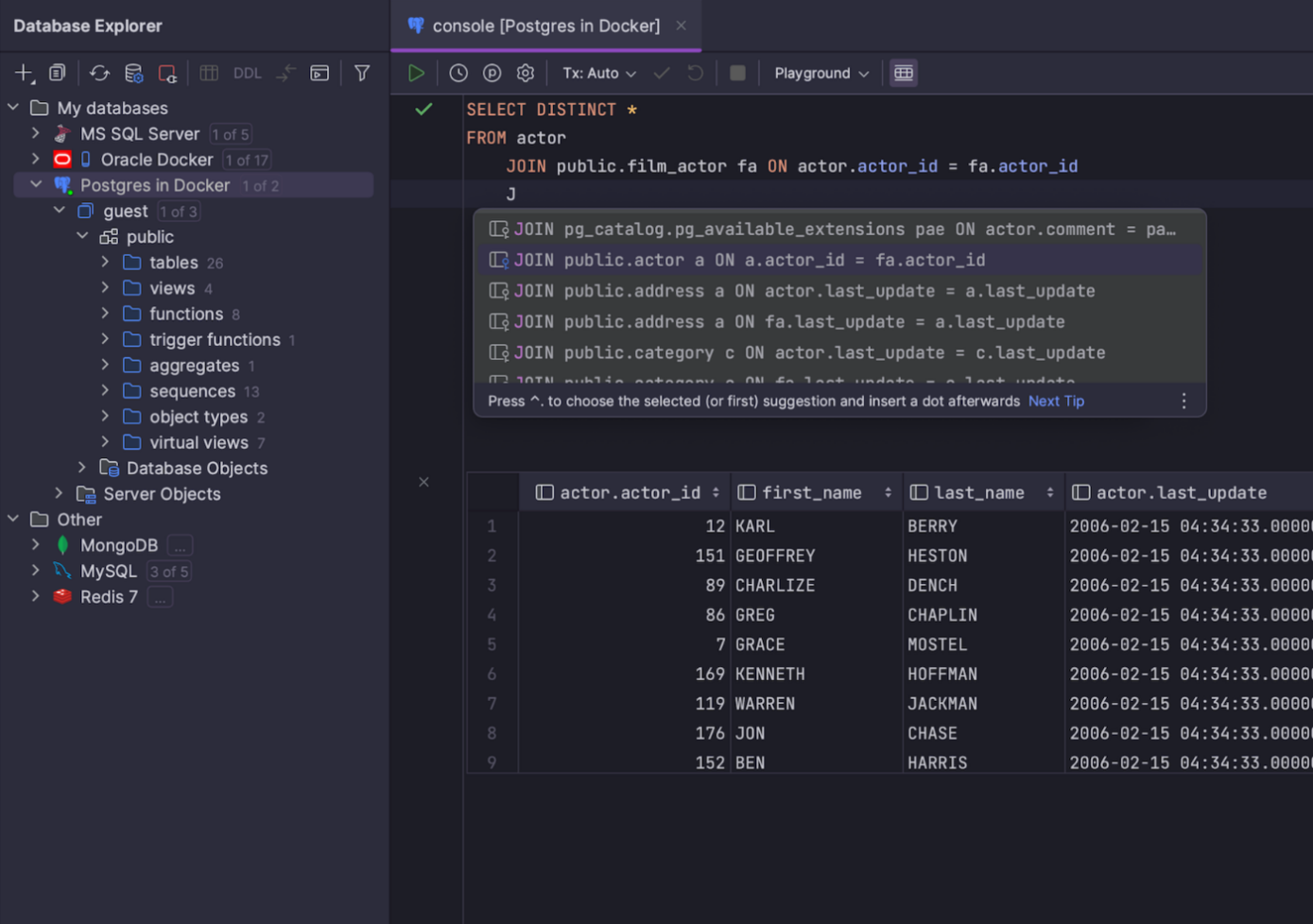
**Рисунок 2.3.5. Интерфейс DataGrip при работе с Oracle**

Эта СУБД относится к объектно-реляционному типу. Название произошло от названия разработавшей эту систему фирмы Oracle. Наравне с SQL СУБД использует процедурное расширение под названием PL/SQL, а также язык Java.

Oracle – это система, отличающаяся стабильностью уже не один десяток лет, поэтому ее выбирают корпорации, для которых важна надежность восстановления после сбоев, отлаженная процедура бэкапа, возможность масштабирования и другие ценные возможности. К тому же эта СУБД обеспечивает отличную безопасность и эффектную защиту данных [11, стп 183-320].

В отличие от других СУБД, стоимость покупки и использования Oracle достаточно высока, и именно это зачастую является значимым препятствием к ее использованию в небольших фирмах. Вероятно, именно это также является причиной того, что в рейтинге лучших СУБД на 2016 год в России Oracle находится лишь на 6-м месте.

**MongoDB**



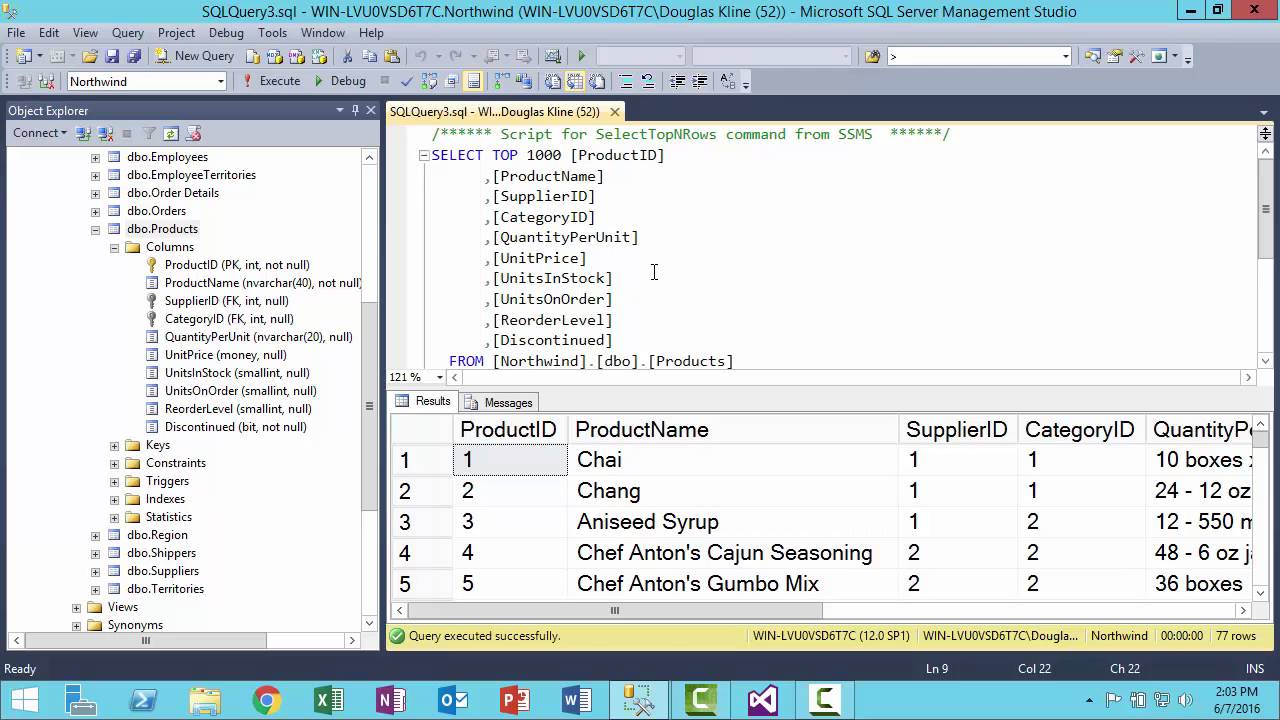
**Рисунок 2.3.6. Интерфейс DataGrip при работе с MongoDB**

Эта СУБД отличается тем, что она предназначена для хранения иерархических структур данных, и поэтому ее называют документоориентированной (она представляет собой документное хранилище без использования таблиц или схем). MongoDB имеет открытый исходный код.

Используя идентификатор, вы можете производить быстрые операции над объектом [8, стр. 81-134]. Также эта СУБД хорошо показывает себя и при сложных взаимодействиях. В первую очередь речь идет о быстродействии – в некоторых случаях приложение, написанное на MongoDB, будет работать быстрее, чем такое же приложение, использующее SQL, т.к. MongoDB относится к классу СУБД NoSQL и пользуется объектным языком запросов, который значительно легче SQL.

Однако этот язык имеет и свои ограничения,  и потому MongoDB следует использовать в случаях, когда нет необходимости в сложных и нетривиальных выборках.

**Microsoft SQL Server**

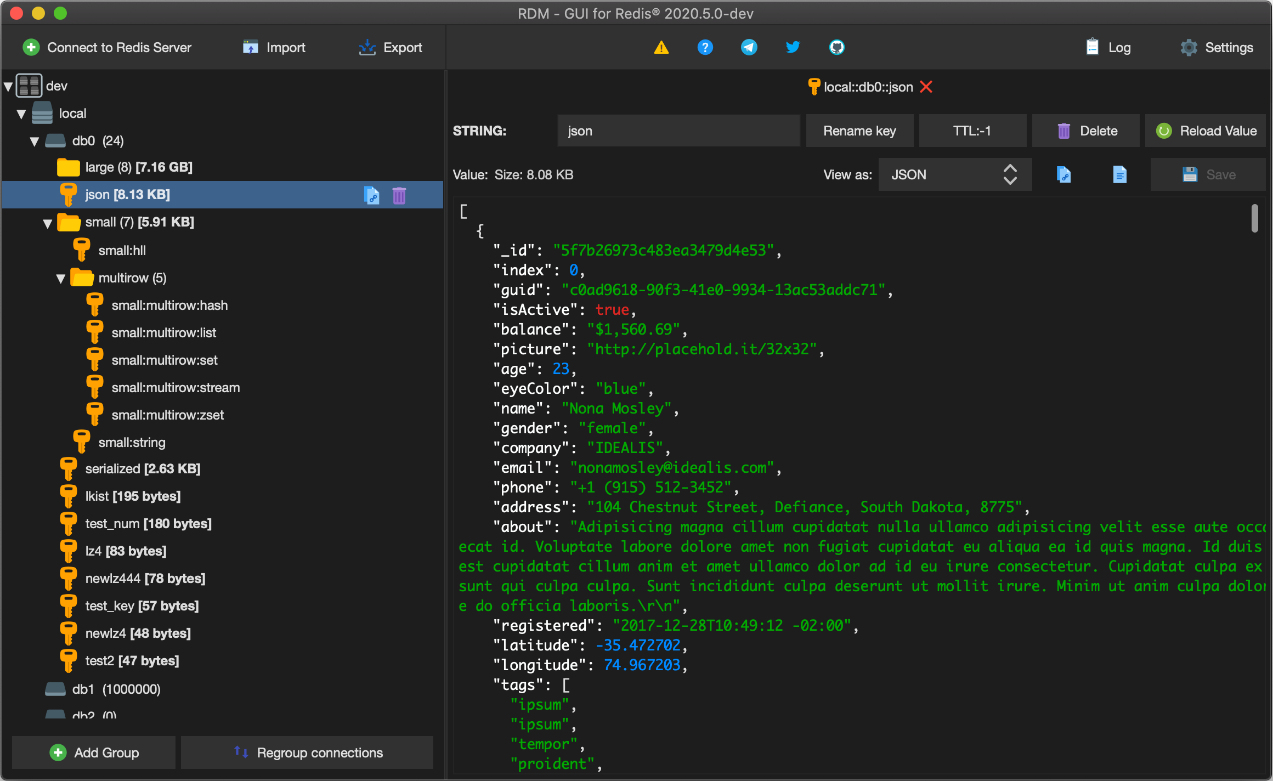
****

**Рисунок 2.3.7. Интерфейс работы SQL Server Management Studio**

SQL Server – это реляционная СУБД, разработанная компанией Microsoft. Она использует язык SQL для запросов к базе данных и может работать на различных операционных системах, включая Windows и Linux. SQL Server имеет широкий спектр функций и инструментов, включая поддержку транзакций, индексацию, репликацию и многое другое [12, стр.146].

Основное преимущество Microsoft SQL Server в добавлении автоматизации задач, например, можно интегрировать код, который будет управлять памятью.

**Redis**

****

**Рисунок 2.3.8. Интерфейс работы RDM (Redis Desktop Manager)**

Redis – использует пары ключ-значение для хранения данных. Она отличается высокой производительностью и низкой задержкой при выполнении операций. Redis поддерживает множество типов данных, включая строки, списки, множества и хеш-таблицы. Дополнительно СУБД  поддерживает распределенные системы и может использоваться для кэширования данных и реализации очередей сообщений.

**Что такое NoSQL-системы**

Выше мы уже немного упоминал о NoSQL-системах – это один из видов СУБД, в котором не используется традиционная модель данных с табличным представлением информации. Здесь применяются другие способы хранения данных, такие как документо-ориентированные, ключ-значение, графовые или колоночные модели [21, стр.35-58].

Такие системы позволяют эффективно хранить и обрабатывать большие объёмы разнородных данных, обеспечивая высокую производительность и масштабируемость. NoSQL-системы часто применяются для хранения и использования данных веб-приложений, социальных сетей и других приложений, где необходимо обработать большое количество данных в режиме реального времени.

**ГЛАВА 3. ПРАКТИКА СИСТЕМ МЕНЕДЖМЕНТА УНИВЕРСИТЕТА НА ОСНОВЕ АИС В РАЗЛИЧНЫХ СТРАНАХ, А ТАКЖЕ ПРАКТИЧЕСКАЯ РАЗРАБОТКА ПРОЕКТА**

**3.1.Системы менеджмента университета на АИС в различных странах**

Использование АИС в управлении университетами стало общепринятой практикой во всем мире [6, стр.45]. Эти системы автоматизируют различные аспекты университетской деятельности, от приёма студентов и регистрации курсов до управления расписанием занятий и выставления оценок. Данные системы, могут существенно различаться в разных странах в зависимости от местных потребностей, законодательства и технологического развития. Вот несколько основных элементов, которые могут быть включены в такие системы:

* **Учет студентов:** В АИС входят данные о студентах, их учебных планах, академической успеваемости, финансовых обязательствах и другой сопутствующей информации [4, стр.316].
* **Управление учебными программами:** Система может предоставлять средства для создания, изменения и отслеживания учебных программ, а также расписаний занятий.
* **Финансовый учет:** Включает в себя управление финансовыми операциями, такими как оплата за обучение, стипендии, финансовая помощь, бухгалтерский учет и отчетность.
* **Управление персоналом:** В системе хранятся данные о преподавателях, их расписаниях, квалификации, заработной плате, а также информация о другом административном персонале.
* **Академические ресурсы:** Включает в себя библиотечные ресурсы, онлайн-курсы, электронные библиотеки и другие академические ресурсы, доступные для студентов и преподавателей [22, стр.72]..
* **Управление исследованиями и проектами:** Для университетов, активно занимающихся исследовательской деятельностью, АИС может содержать информацию о научных проектах, финансировании и научных публикациях.
* **Аналитика и отчетность:** Важная часть системы, позволяющая администрации университета анализировать данные о студентах, финансах, академической производительности и принимать обоснованные решения на основе этих данных.

Хотя основные элементы могут быть схожими, тем не менее, конкретные решения и функциональность АИС могут значительно различаться в разных странах и даже в разных университетах внутри одной страны [17, стр. 71-75]. Например, в некоторых развитых странах университеты могут иметь высоко-интегрированные системы, которые объединяют в себе все вышеперечисленные аспекты, в то время как в развивающихся странах университеты могут использовать менее развитые системы или даже полагаться на ручные процессы.

Вот несколько примеров:

**США:** Системы ERP (Enterprise Resource Planning): Широко используются комплексные системы ERP, такие как SAP и Oracle, которые интегрируют различные функции управления, такие как финансы, кадры, студенческие записи и т. д.

Системы LMS (Learning Management Systems): Популярны системы LMS, такие как Blackboard и Moodle, которые используются для создания и доставки онлайн-курсов, а также для управления заданиями, оценками и общением между студентами и преподавателями.

**Великобритания:** Системы SITS (Student Information Systems): Распространены специализированные системы SITS, такие как Banner и Unicom, которые ориентированы на управление записями о студентах, расписанием занятий, оценками и т. д.

Системы VLE (Virtual Learning Environments): Используются системы VLE, такие как Blackboard и Brightspace, для создания онлайн-курсов и управления учебным процессом.

**Россия:** Системы УИС (Университетские информационные системы): Разработаны и внедрены университетские информационные системы, такие как Университетская информационная система (УИС) МГУ, АИС "Университет", которые охватывают широкий спектр функций управления университетом.

Системы LMS: Используются системы LMS, как отечественные (Moodle, iSpring Learn), так и зарубежные (Blackboard, Brightspace).

**Китай:** Системы ERP: Внедрены системы ERP, разработанные китайскими компаниями, такие как SAP Business One и Kingdee.

Системы LMS: Используются как отечественные системы LMS (SuperMap GIS, eSchool), так и зарубежные (Blackboard, Moodle).

Помимо этих примеров, существует множество других АИС, используемых университетами во всем мире.

Выбор АИС зависит от различных факторов, таких как:

* Размер и тип университета
* Бюджет
* Потребности пользователей
* Техническая инфраструктура

Важно отметить, что внедрение АИС может быть сложным и трудоёмким процессом [19, стр. 125].

Необходимо тщательно планировать и управлять процессом внедрения, чтобы обеспечить его успешное завершение.

**3.2. Построение файловой структуры проекта и установка связи между нашим проектом и системой управления базами данных**

В предыдущих главах мы подробно разобрали все особенности и преимущества технологий, которые мы будем применять в построении нашего проэкта, а в данной, заключительной главе будет рассматриваться лишь их практическое применение на проекте. Приступим к разработке нашего MVC проекта с использованием Visual Studio 2022.

При создании нового проекта ASP.NET среда Visual Studio предоставляет несколько вариантов первоначального содержимого, которое требуется для проекта. Идея состоит в том, чтобы упростить процесс изучения разработчикам-новичкам и применить экономящие время рекомендуемые приемы для распространенных средств и задач. Такая поддержка воплощена через шаблоны, используемые для создания контроллеров и представлений, которые создаются с помощью шаблонного кода для вывода списков объектов данных, редактирования свойств модели и т.д.

В версиях Visual Studio 2013 и MVC 5 разработчики из Microsoft обновили шаблоны и то, что известно как формирование шаблонов, чтобы размыть границы между разными видами проектов ASP.NET и предоставить более широкий диапазон шаблонов проектов и конфигураций кода.

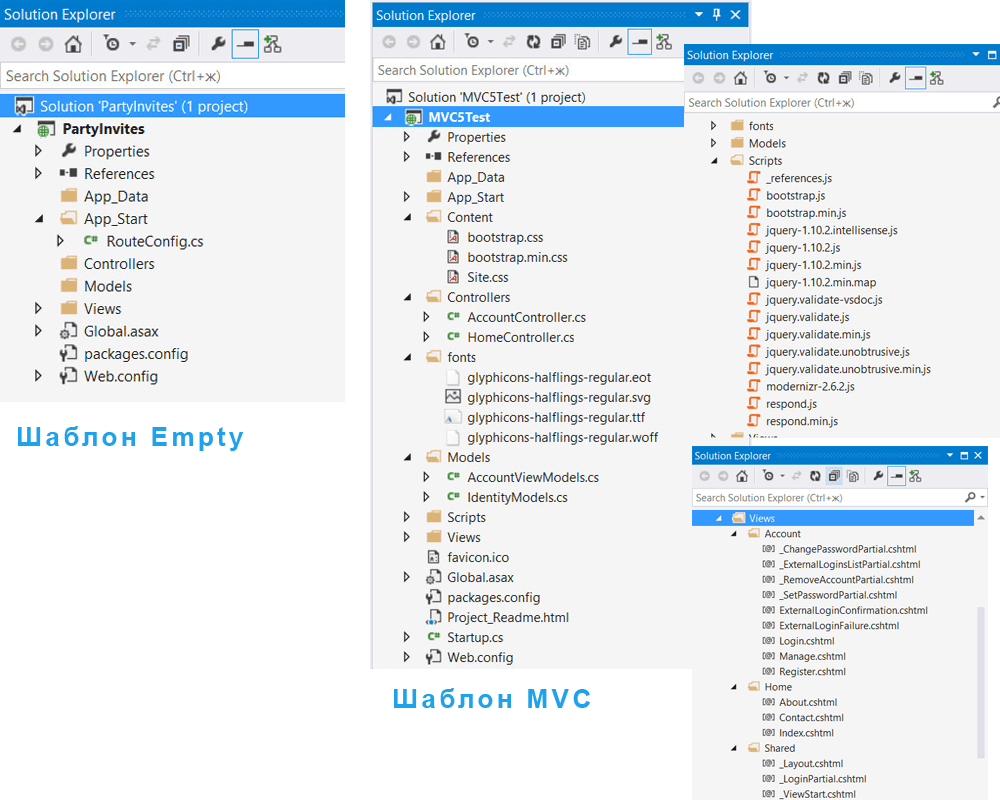
В ходе моего исследования я поняла, что лучше всего начинать с пустого проекта и по мере необходимости добавлять файлы и пакеты. Поступая так, вы не только лучше изучите работу MVC Framework, но также будете иметь полный контроль над тем, что содержит ваше приложение.

**Создание проекта**

Когда вы впервые создаете новый проект MVC Framework, у вас есть на выбор две базовых отправных точки: шаблон Empty (Пустой) и шаблон MVC. Эти имена несколько обманчивы, т.к. добавить базовые папки и сборки, требуемые для MVC Framework, можно в любой проект, отметив флажок MVC в разделе Add folders and core references for (Добавить папки и основные ссылки для) диалогового окна New ASP.NET Project (Новый проект ASP.NET), как показано на рисунке ниже. В случае выбора шаблона MVC этот флажок отмечается автоматически.

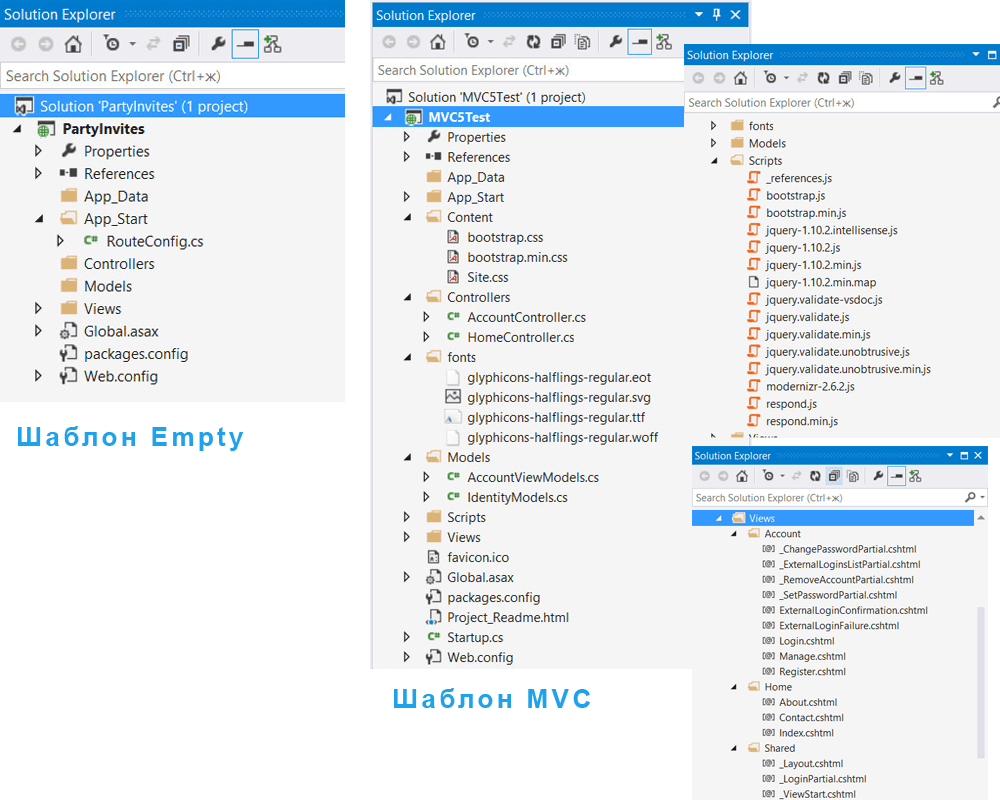
Реальное отличие связано с дополнительным содержимым, которое шаблон MVC добавляет в новые проекты. Оно предоставляет готовую отправную точку, включающую стандартные контроллеры и представления, конфигурацию безопасности, ряд популярных пакетов JavaScript и CSS (таких как jQuery и Bootstrap), а также компоновку - которая применяет библиотеку Bootstrap для построения темы, оформляющей пользовательский интерфейс приложения.

Вариант Empty проекта содержит только базовые ссылки, требуемые для MVC Framework, и базовую структуру папок.



**Рисунок 3.2.1. Шаблон Empty проекта**

Шаблон MVC добавляет довольно много содержимого, и разницу можно видеть на рисунке ниже. Проект ниже был создан с использованием шаблона MVC, и чтобы продемонстрировать все файлы в нем, пришлось открывать разные папки в окне Solution Explorer, т.к. список файлов оказался бы слишком длинным.



**Рисунок 3.2.2. Шаблон MVC проекта**

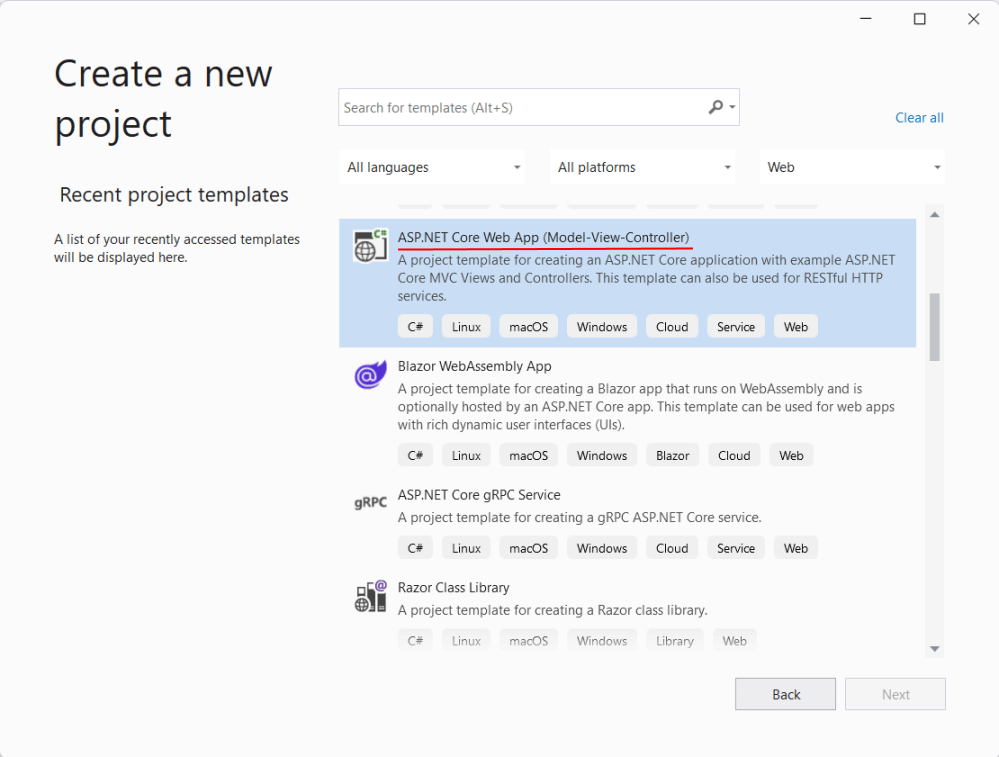
Дополнительные файлы, добавляемые в проект шаблоном MVC, выглядят хуже, чем есть на самом деле. Некоторые из них относятся к аутентификации, а другие представляют собой файлы JavaScript и CSS, для которых предусмотрены обычные и минимальные версии.

Среда Visual Studio собирает проект, созданный с помощью шаблона MVC, используя пакеты NuGet, а это значит, что вы можете просмотреть, какие пакеты применяются, выбрав пункт Manage NuGet Packages for Solution (Управление пакетами NuGet для решения) в меню Tools => Library Package Manager (Сервис => Диспетчер библиотечных пакетов). Это также означает возможность добавления тех же самых пакетов в любой проект, в том числе и созданный с использованием шаблона Empty.

Какой бы шаблон вы ни выбрали, вы заметите, что результирующие проекты имеют очень похожие структуры папок. Некоторые из элементов в проекте MVC играют особые роли, жестко закодированные в ASP.NET или MVC Framework. Другие имеют отношение к соглашениям об именовании.

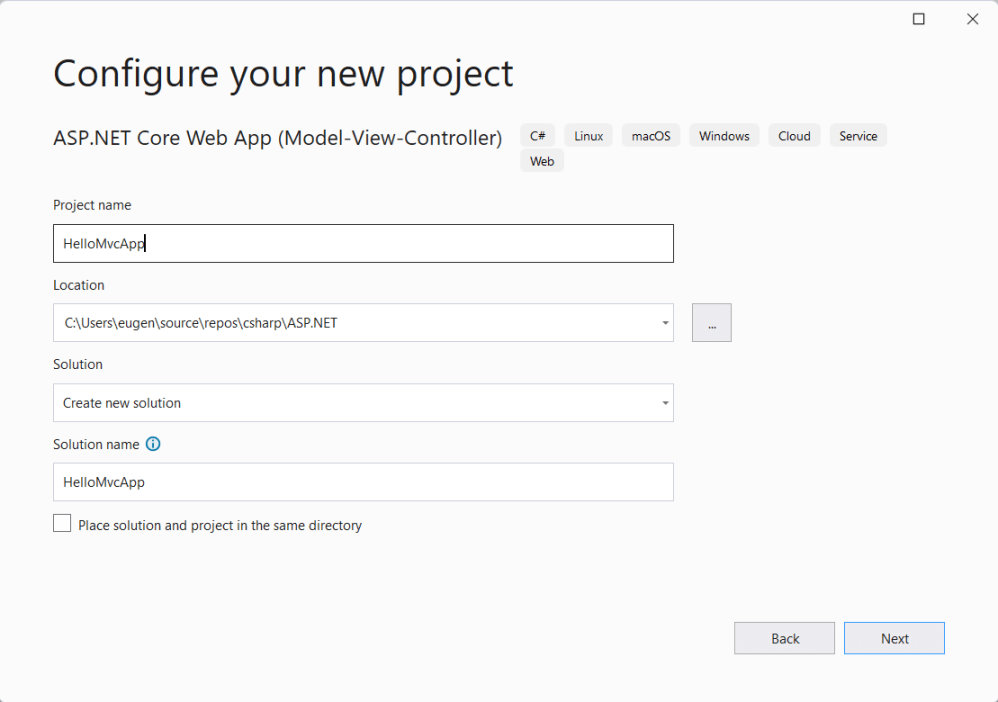
**Создание нашего проекта при помощи шаблона ASP.NET MVC**

Как мы уже отметили выше, в Visual Studio 2022 для создания проекта на ASP.NET Core MVC мы можем выбрать любой тип проекта на ASP.NET Core и в нем уже добавлять необходимые компоненты (Рисунок 3.2.3). Однако для упрощения Visual Studio уже по умолчанию предоставляет для этого шаблон **ASP.NET Core Web App (Model-View-Controller)**:



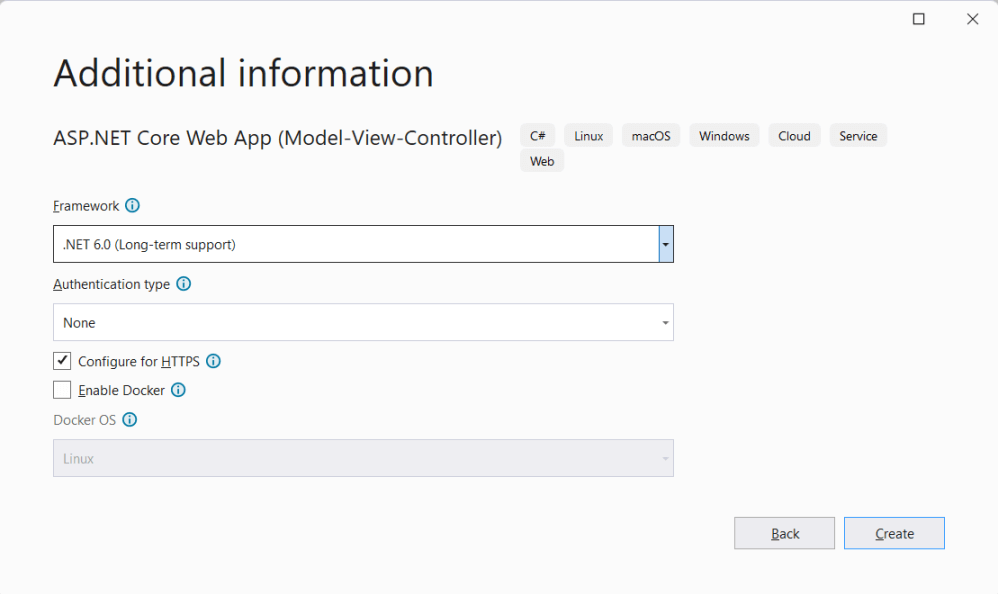
**Рисунок 3.2.3.**

Выберем данный шаблон для создания проекта. Дальше нам откроется окно для установки имени проекта. Допустим, проект будет называться HelloMvcApp (Рисунок 3.2.4). Данный шаблон будет продемонстрирован в качестве примера начального этапа создания проекта, а в действительности наш проект будет называть **University-Course-And-Result-Management-System.**



**Рисунок 3.2.4.**

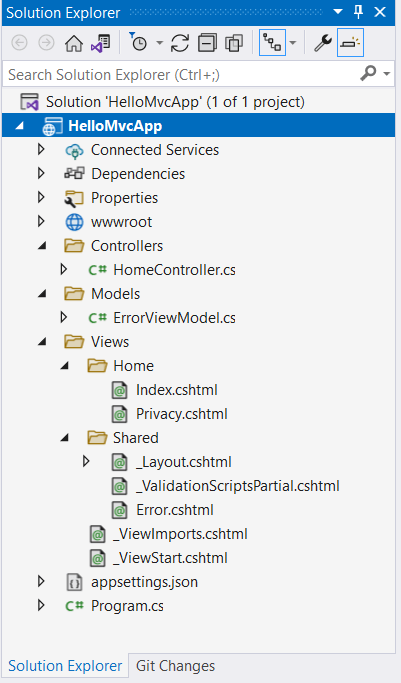
Далее нам надо будет настроить стандартные настройки для ASP.NET Core (Рисунок 3.2.5):



**Рисунок 3.2.5.**

Оставим все настройки по умолчанию и нажмем на ОК. И Visual Studio создаст новый проект MVC.

Структура папок созданного нами проэкта будет выглядеть следующим образом, где каждая папка будет содержатиь данные и файлы определенного специального типа:

* **Dependencies**: все добавленные в проект пакеты и библиотеки
* **wwwroot:** этот узел предназначен для хранения статических файлов - изображений, скриптов javascript, файлов css и т.д., которые используются приложением.
* **Controllers:** папка для хранения контроллеров, используемых приложением. По умолчанию здесь уже есть один контроллер - Homecontroller
* **Models**: каталог для хранения моделей. По умолчанию здесь создается модель ErrorviewModel
* **Views**: каталог для хранения представлений. Здесь также по умолчанию добавляются ряд файлов - представлений
* **appsettings. json:** хранит конфигурацию приложения
* **Program.cs:** файл, который определяет входную точку в приложение ASP.NET Core

**Рисунок 3.2.6. Стуктура папок созданного шаблонного MVC проэкта**

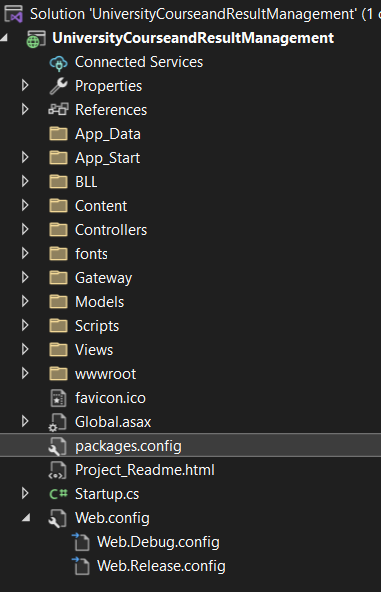
Как было уже отмечено ранее, эта та же структура, что и у проекта по типу Empty за тем исключением, что здесь также добавлены по умолчанию папки для ключевых компонентов фреймворка MVC: контроллеров и представлений. А также есть дополнительные узлы и файлы для управления зависимостями клиентской части приложения.

И если мы запустим проект на выполнение, то сработает запрос к контроллеру по умолчанию - классу HomeController, который выберет для генерации ответа нужное представление. И в итоге из представления будет создана html-страница, которую мы увидим в своем веб-браузере.

**Построение файловой структуры всех бизнес процессов проекта**

На данном этапе мы приступим к разработке фронт и бэк частей нашего проекта, а именно, мы будем редактировать различные папки как Views, Content , Controllers, BLL, Model и соответствующие им стилистики в папке wwwroot.

Для большего удобства и систематизации бизнес процессов нашего проекта, нам следует создать все необходимые файлы и папки нашего проекта, внутри которых будут содержаться валидации, аутентификация, авторизация, ветвления бизнеса, передача данных между этапами, файлы со скриптами загруженные из официальных документаций и так далее [22, стр.63-100]. В конечном итоге файловая система проекта будет выглядеть следующим образом :



**Рисунок 3.2.7. Стуктура папок созданного нами MVC проекта**

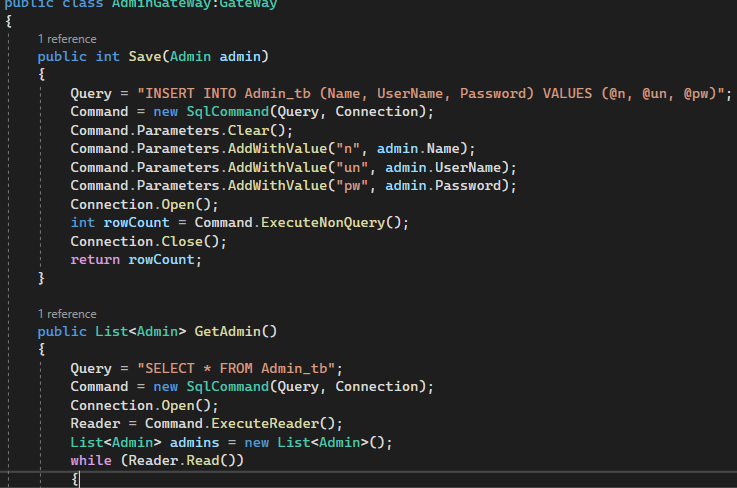
Дадим пояснения относительно содержания каждой папки:

**App\_Start** - здесь будут находиться скрипты роутинга и конфигурации для подключения всех необходимых библиотек

**BLL (Business Logic Layer)** - будет содержать скрипты касательно бизнес логики, а именно здесь будут вызываться отдельные файлы из папки с, перед изменением, удалением, передачей или получением данных из базы

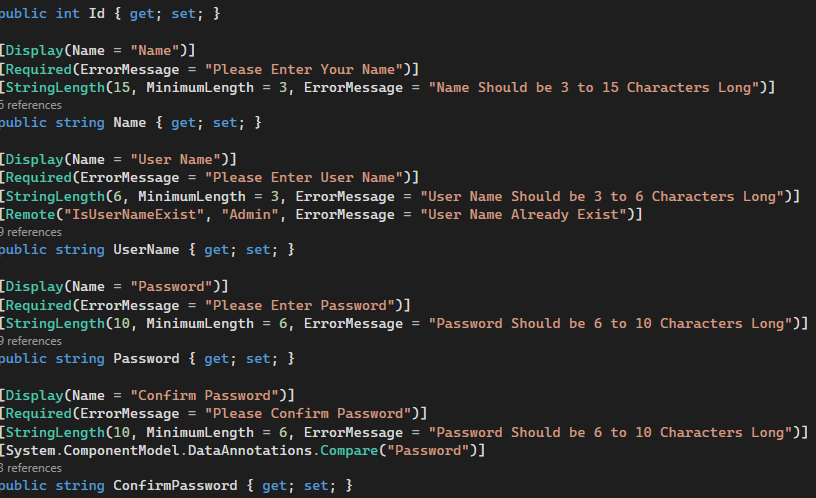
**fonts** - здесь находятся файлы шрифтов, которые будут нами использоваться(файлы были загружены из официальных документаций)

**Gateway** - из названия можно понять, что здесь будут находиться “пути” логики каждого из этапов. Один из наглядных примеров выглядит следующим образом :

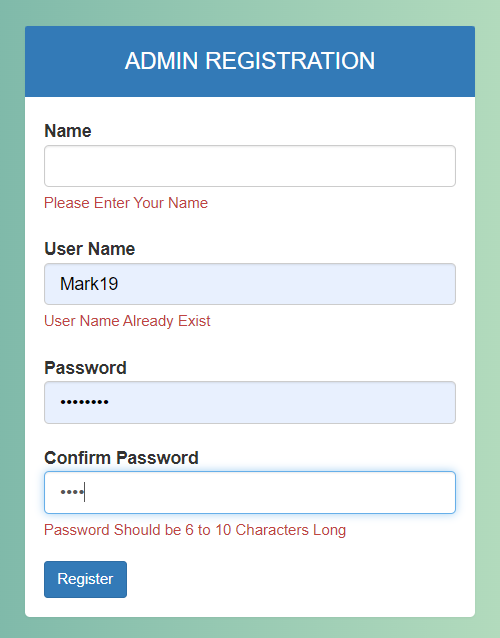


**Рисунок 3.2.8. Стуктура Gateway файла для объекта Admin**

**Models -** содержит модели всех Entity с которыми мы будет работать.Здесь имеется ввиду какие поля и валидации к данным полям будут содержаться в данных объектах.Один из наглядным примеров будет выглядеть следующим образом, а именно на первом изображении скриптовое отображение (Рисунок 3.2.9), а на втором как это наглядно будет изображено в проекте (Рисунок 3.2.10):

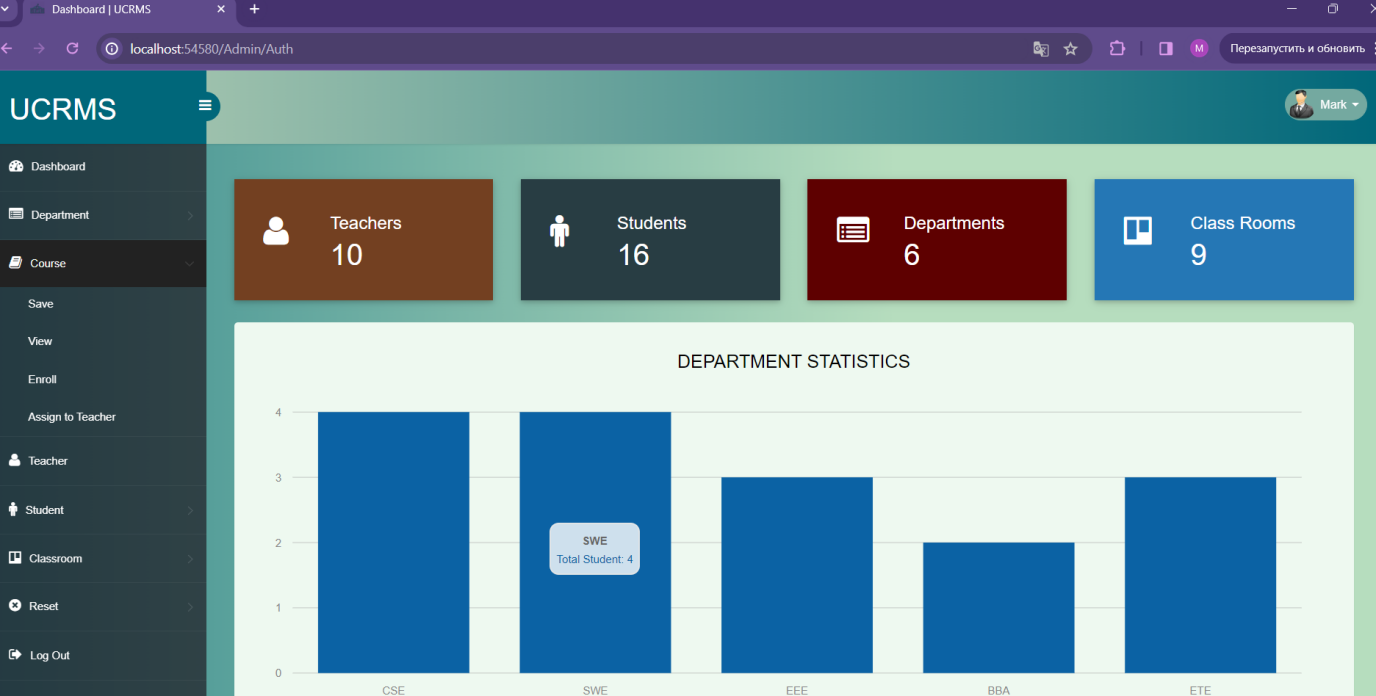


**Рисунок 3.2.9.**



**Рисунок 3.2.10.**

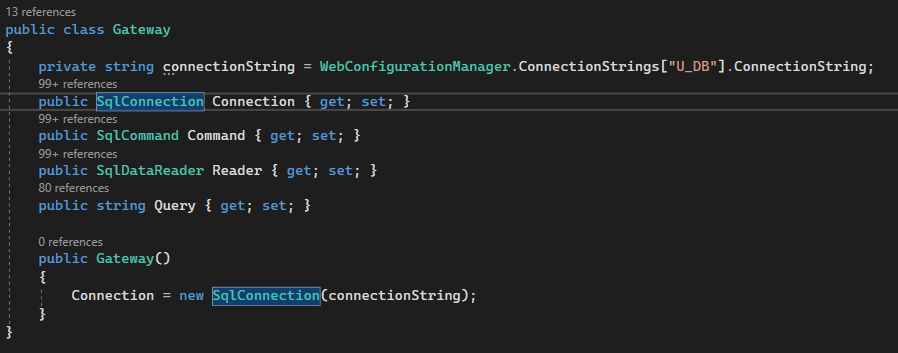
Views - будет содержать файлы представлений каждого из элементов проекта, но данная папка не сможет реализовать свою деятельность без папок Controller и wwwroot. Внутри данной папки мы можем заметить содержание файла \_ViewStart.cshtml внутри которого указан путь в файл \_Layout.cshtml, внутри которого будут находиться статические части проекта , а также меняющиеся части в зависимости от того на каком этапе мы находимся (они определяются по аннотации @RenderBody() ).Он в свою очередь перенаправляет нас \_LayoutPartial.cshtml ,который будет перенаправлять нас между этапами входа в систему и регистрации. В данной папке также здесь будет находиться файл внутреннего роутинга ( \_MasterLayoutPage.cshtml ) внутри системы, который наглядно выглядит следующим образом (Рисунок 3.2.11) :



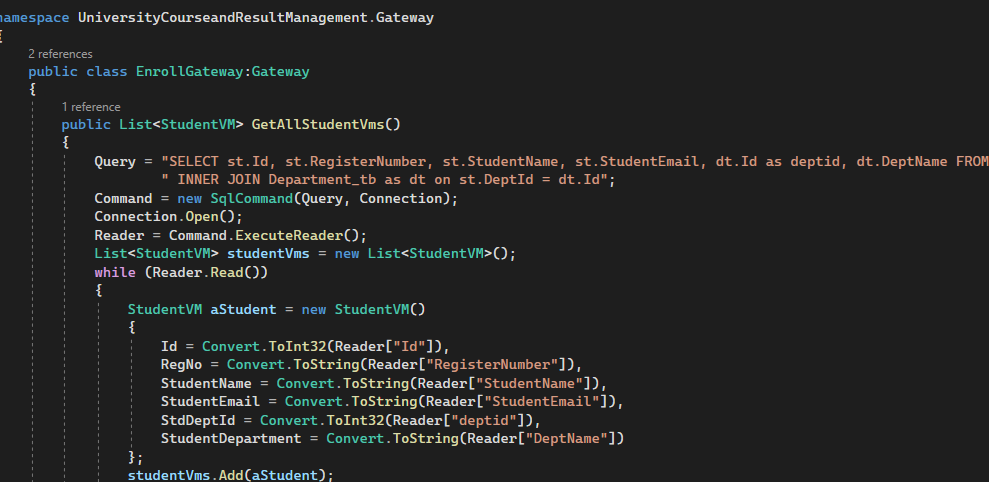
**Рисунок 3.2.11.**

**Установка связи между нашим проектом и СУБД**

В качестве СУБД, которую мы будем использовать в нашем проекте я решила дать предпочтение MS SQL Server, но для установки связи между проектом и СУБД, необходимо настроить строку подключения, иначе говоря Connection String [10, стр.135]. Для этого в папке Gateway , нам необходимо создать файл Gateway , внутри которого и будет конфигурация строки подключения к базе данных.Помимо этого от данного класса будут наследоваться классы , которые также находятся в папке Gateway . Как было упомянуто ранее в них находятся “пути” логики каждого из этапов, реализованные при помощи SQL скриптов [14, стр. 501] и объектами Models.



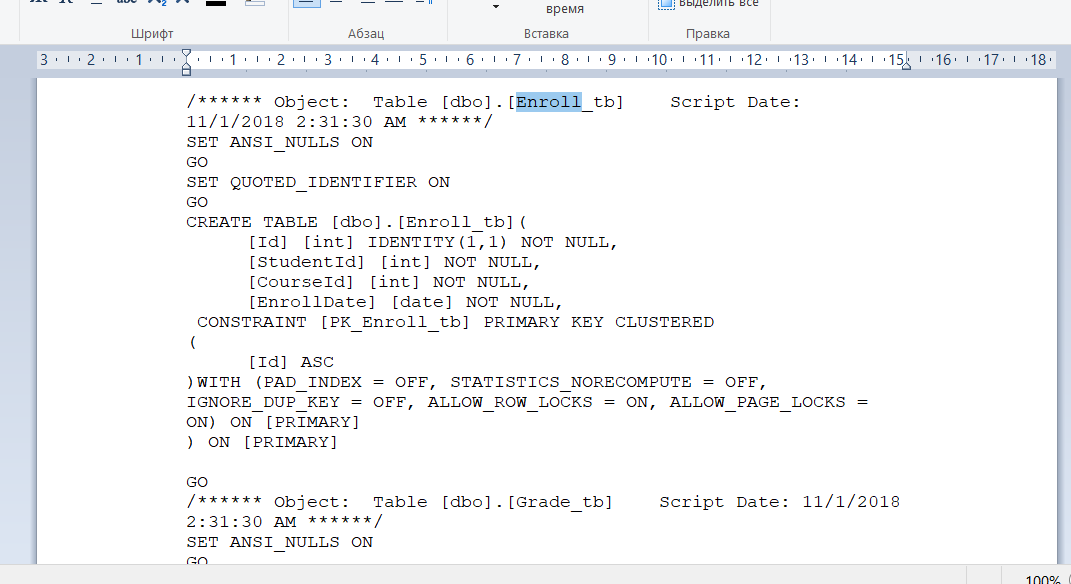
**Рисунок 3.2.12. Конфигурация строки подключения**



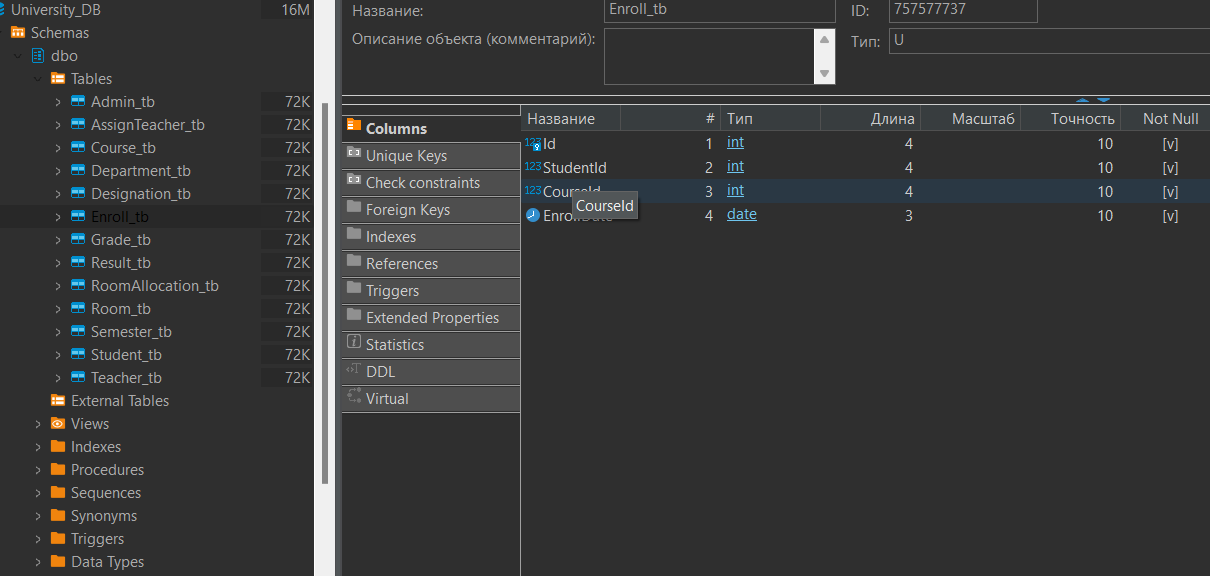
**Рисунок 3.2.13. Наследование класса записи струдентов на курсы от класса Gateway**

Здесь хотелось бы отметить, что для дальнейшего более удобного способа реализации всех SQL скриптов мы можем их вывести в отдельную внешнюю папку под названием DataBase.sql (Рисунок 3.2.14).

Далее мы можем использовать нашу БД, открыв её при помощи универсального инструмента для различных СУБД DBeaver.Реализация этого достаточно проста в связи с удобствами интерфейса которого мы используем, и в результате в окне нашего инструмента мы можем увидеть наглядно увидеть информационно базовую настройку наших файлов(Рисунок 3.2.15)



**Рисунок 3.2.14.**



**Рисунок 3.2.15.**

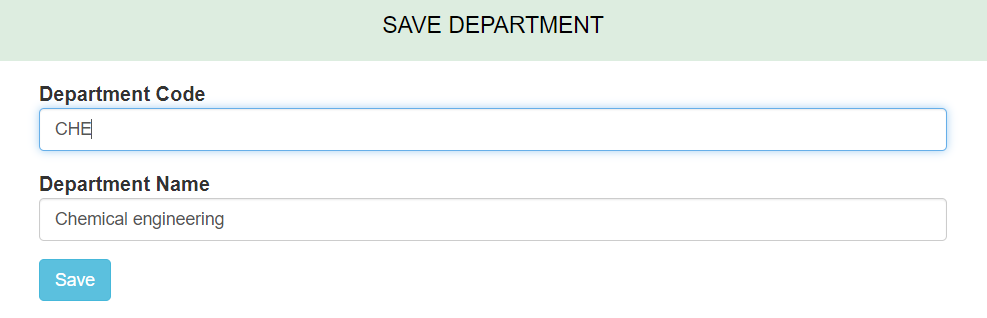
**3.3. Демонстрация основных функциональностей системы**

В данной заключительной подглаве, мы наглядно продемонстрируем реализованные нами функциональности проекта при помощи подготовленноговеб интерфейса и дадим разъяснение и описание каждому из пунктов.

Приступим к их описанию:

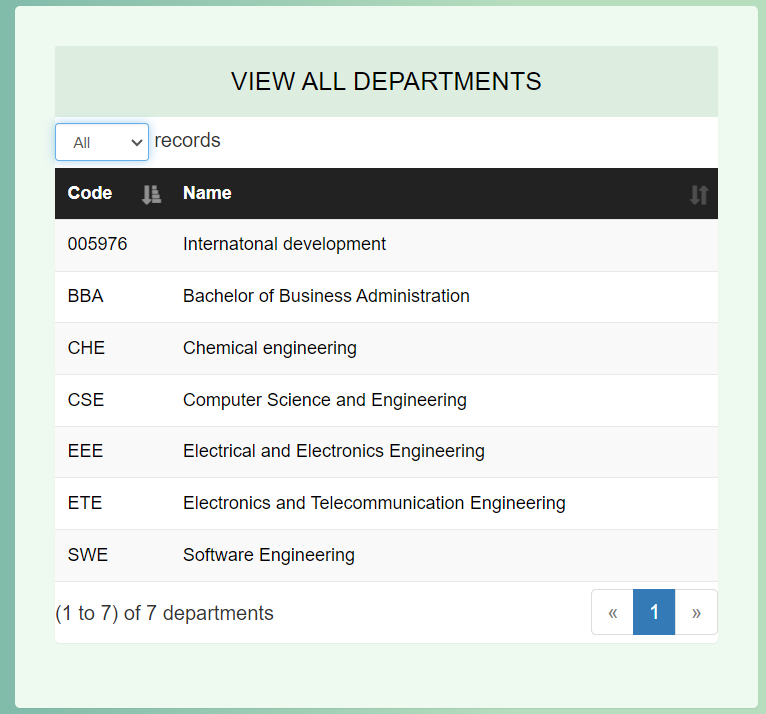
**Департаменты**

1. **Сохранение департамента**. При сохранении департамента необходимо обеспечить уникальность кода и названия. Код должен состоять из двух (2) до семи (7) символов.(Рисунок 3.3.1)



**Рисунок 3.3.1.**

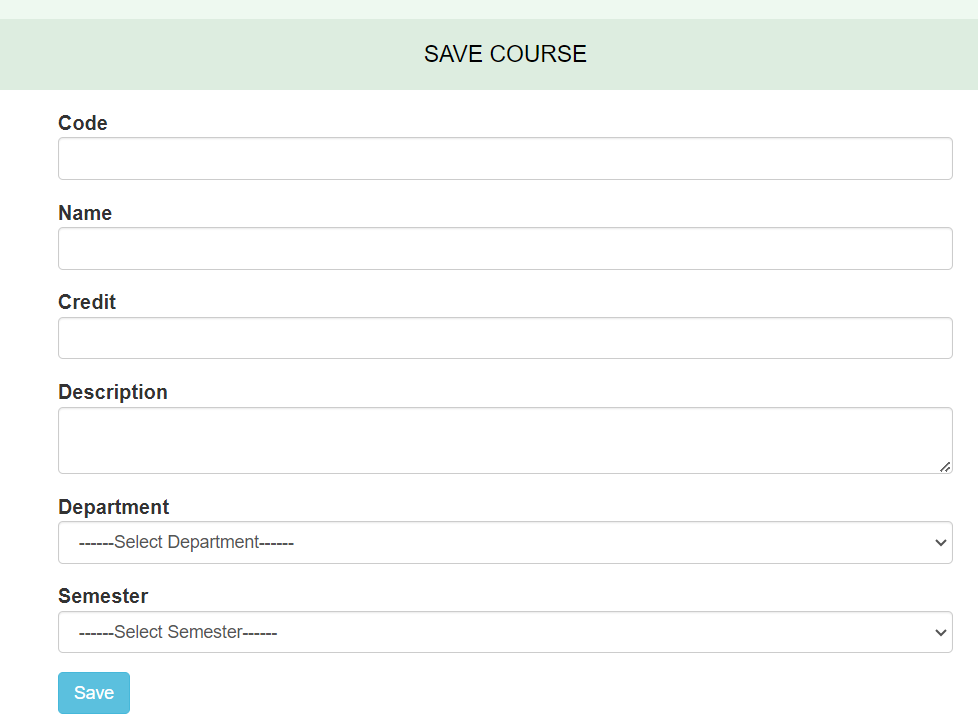
1. **Просмотр списка всех департаментов.** На этой странице будут отображаться все существующие данные об отделах.(Рисунок 3.3.2)



**Рисунок 3.3.2.**

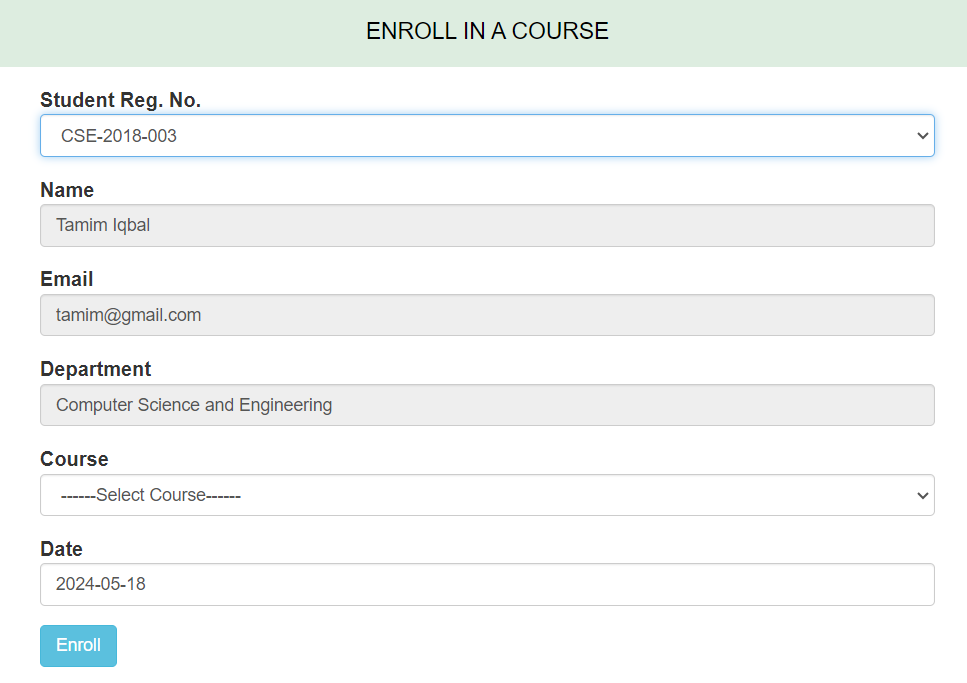
**Курсы**

1. **Сохранение курса**. Здесь код и название должны быть уникальными. Код должен содержать как минимум пять (5) символов. Диапазон кредитов составляет от 0,5 до 5,0, т. е. кредит не может быть меньше 0,5 и больше 5,0. В списке раскрывающегося списка отделов будут загружены существующие данные об отделах из базы данных. В базе данных должны храниться данные об восьми (8) семестрах, и список раскрывающегося списка семестров будет загружен этими данными.( Рисунок 3.3.3)



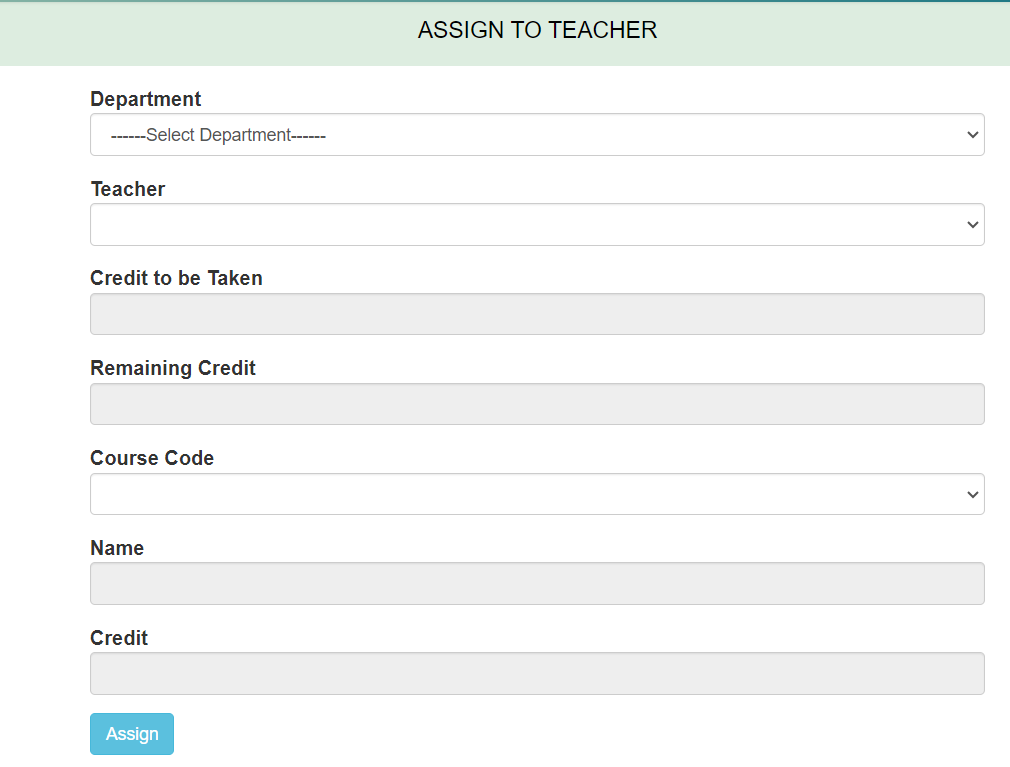
**Рисунок 3.3.3.**

1. **Зачисление на курс**. В раскрывающемся списке номеров регистрации студента будут загружены существующие номера регистрации студентов. Пользователь выберет номер регистрации, и будут отображены имя, адрес электронной почты и отдел этого конкретного студента. В раскрывающемся списке "Выбор” (Рисунок 3.3.4)



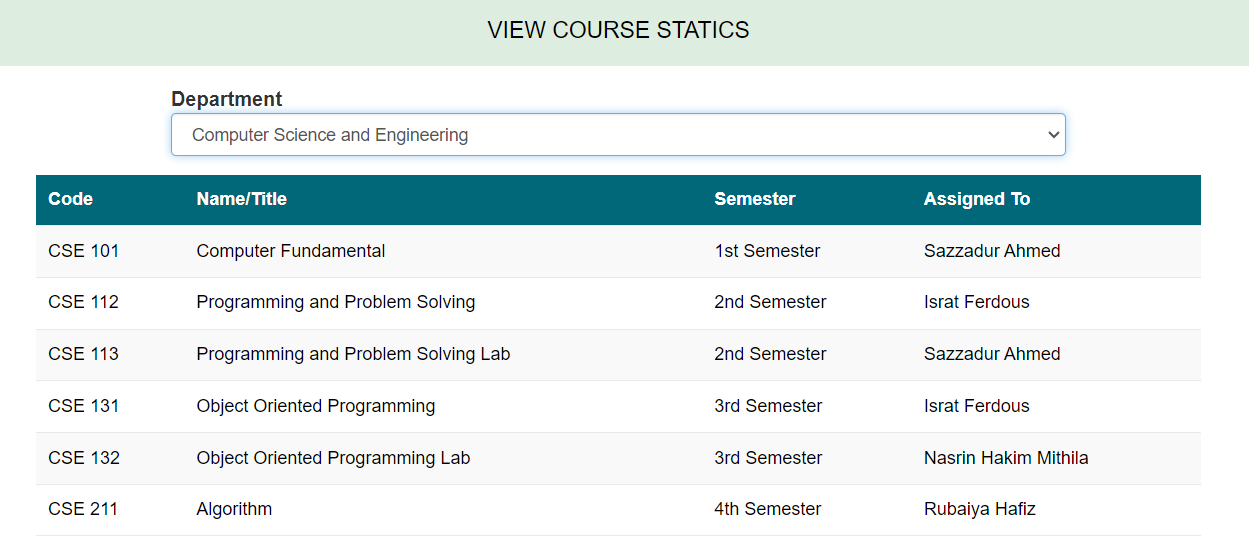
**Рисунок 3.3.4.**

1. **Назначение курса преподавателю.** Пользователь выберет отдел из раскрывающегося списка, и имена всех преподавателей и код курса этого конкретного отдела будут загружены в раскрывающиеся списки "Преподаватель" и "Код курса". При выборе пользователем преподавателя отображается количество зачисленных кредитов и оставшийся кредит соответственно. При выборе пользователем кода курса отображаются название и количество кредитов этого курса. Необходимо избежать проблем с перекрытием. Курс не может быть назначен более чем одному учителю, т. е. назначенный курс не может быть назначен снова. Если пользователь пытается назначить курс, у которого количество кредитов больше, чем оставшийся кредит у учителя, система покажет диалоговое окно с вариантами ответа (Да/Нет) и действует соответственно.( Рисунок 3.3.5)



**Рисунок 3.3.5.**

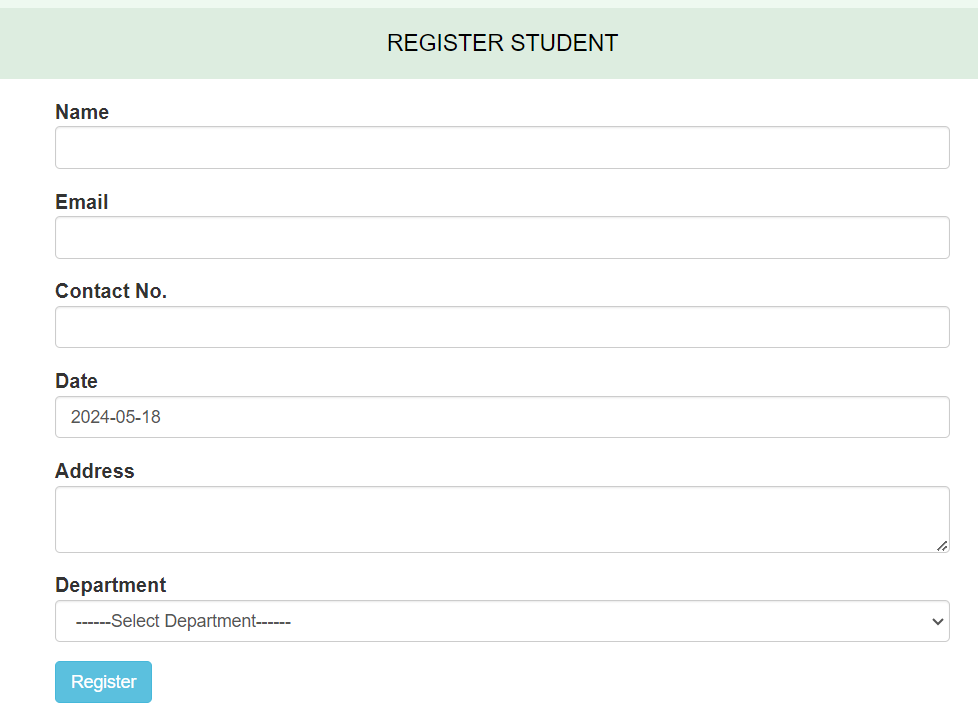
1. **Просмотр статистики курса**. Пользователь выбирает отдел, и информация о всех курсах (Код, Название, Семестр и Назначено) будет отображена соответственно. Если есть курс, который еще не назначен ни одному учителю, то в столбце "Назначено" должно быть написано "Еще не назначено"( Рисунок 3.3.6).



**Рисунок 3.3.6.**

**Студенты**

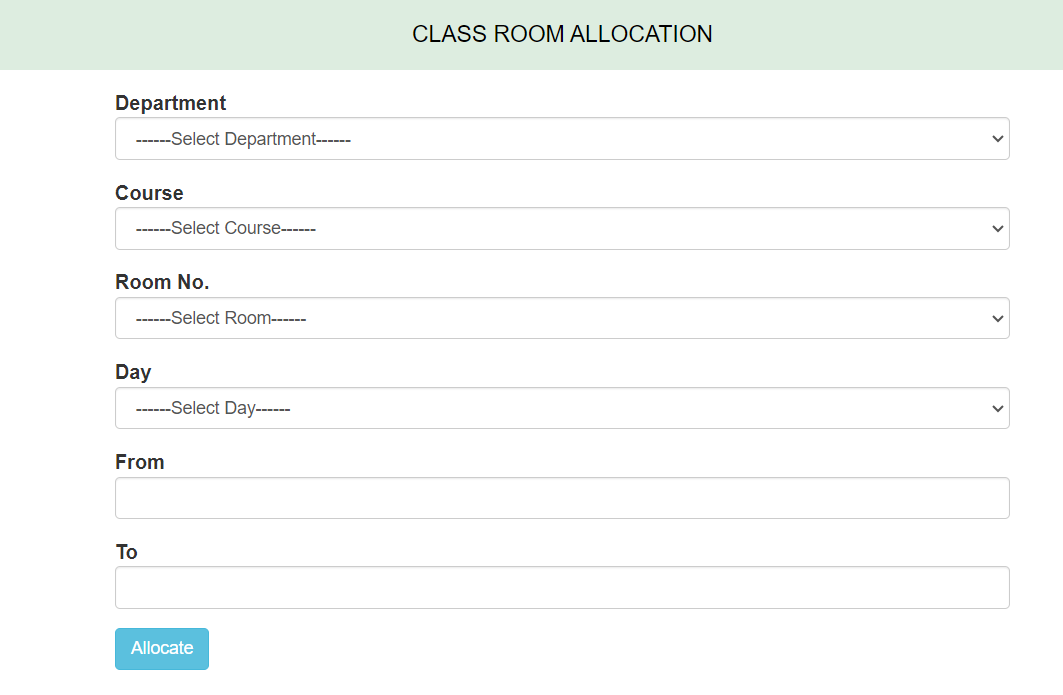
**1.Регистрация студента.** При регистрации студента необходимо обеспечить, чтобы адрес электронной почты был в правильном формате и был уникальным. В поле даты должен быть элемент DatePicker, где по умолчанию выбрана текущая дата. После успешной регистрации отображаются все данные, а также номер регистрации. Для номера регистрации есть фиксированный формат. Формат номера регистрации: --XXX. Например, CSE-2012-001, CSE-2012-002, EEE-2012-001, EEE-2013-001, CSE-2013-001 (Рисунок 3.3.7).



**Рисунок 3.3.7.**

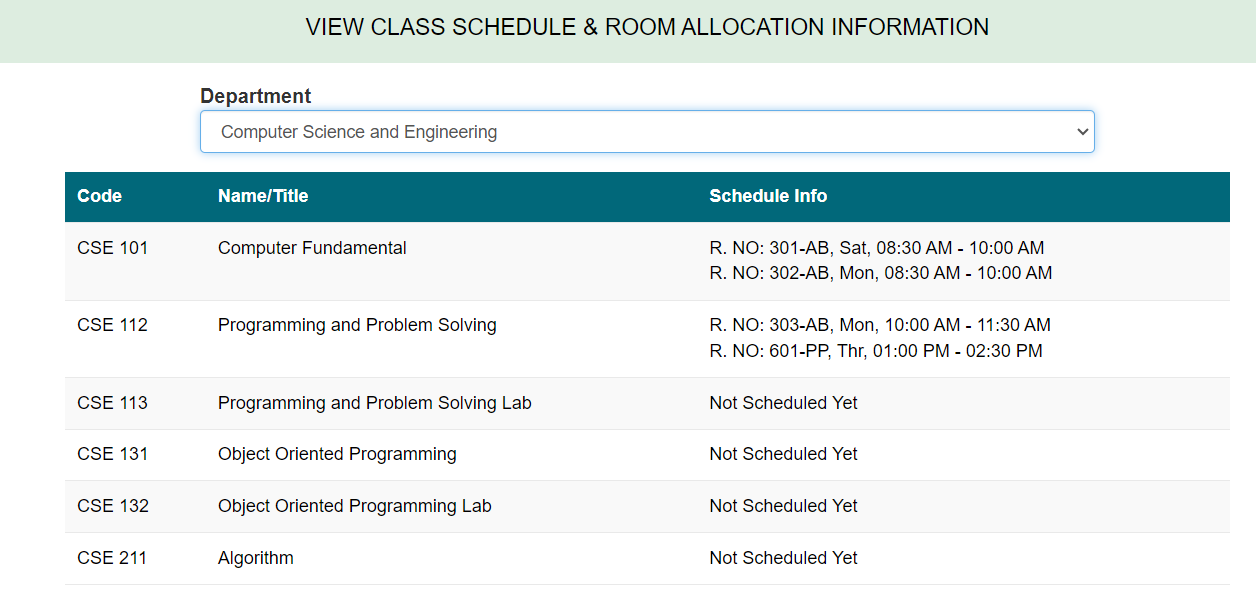
**Аудитории**

**2.Выделение аудиторий**. В раскрывающемся списке дней будет загружено семь (7) названий дней недели. Сохраните некоторые данные о комнатах в базе данных, и список раскрывающегося списка номеров комнат будет загружен этими данными. Необходимо избежать перекрытия проблемы здесь. Необходимо избежать как полного, так и частичного перекрытия (Рисунок 3.3.8).



**Рисунок 3.3.8.**

**3. Просмотр расписания занятий и информации о выделении комнат.** Пользователь выбирает отдел, и будет отображена информация о расписании занятий и выделении комнат по курсам этого конкретного отдела. Обратите внимание, что для одного курса будет создана одна строка, т. е. нельзя создавать несколько строк для нескольких расписаний одного курса. Если есть курс, который еще не запланирован, то в столбце "Информация о расписании" должно быть написано "Еще не запланировано" (Рисунок 3.3.9).



**Рисунок 3.3.9.**

# **ЗАКЛЮЧЕНИЕ**

Процесс автоматизации системы менеджмента университета с применением C#, представляет собой стандартную процедуру, имеющую чёткое описание в стандартах, инструкциях, но в то же время, довольно сложная, требующая учёта множества нюансов и специфики системы, в рамках которой она создаётся, ибо система всегда имеет индивидуальный комплект рабочих блоков.

Разработанный C# ASP.NET Core MVC проект "University-Course-And-Result-Management-System" позволяет автоматизировать работу во внутренней системе менеджмента университета. Несмотря на то, что данный проект не велик, он облегчает и ускоряет работу с управлением данных студентов, ведёт контроль за движением всех учебных графиков университета (как студентов так и преподавателей), обеспечивает четкую и слаженную работу всех администраторов, формирует диаграмму всего состава университета в зависимости от происходящих в нем процессов .

В результате выполненной диссертационной работы была решена задача проверки существования пользователя “администратора” в базе данных университета, его создание при случае его отсутствия в базе, добавления новых предметов и учебных часов, их удаление и модификация из базы данных. Как упоминалось ранее для реализации данного проекта была использована инфраструктура ASP.NET MVC 5, которая была связана с отмеченной выше СУБД, а для его визуального представления были использованы технологии как HTML, CSS, JavaScript, Bootstrap.

Основным преимуществом созданной системы является простота. Таким образом, системой сможет управлять один администратор, не прибегая к помощи различных дополнительных вычислений вручную. Автоматизация процессов обработки и хранения информации в системе менеджмента университета даёт возможность обрабатывать большее количество информации за меньшее количество времени и с большей надёжностью. Благодаря данной системе, все бизнес и прочие процессы будут хорошо отлажены.

В данной диссертационной работе было использовано программирование на языке С#. Программный продукт содержит форму авторизации администраторов, которая открывается после запуска приложения. Данная форма содержит поля для ввода администратором своего логина и пароля, где введённые в свою очередь будут подложены на проверку процедурам авторизации и аутентификации.

Вместе с тем, программа реализует "дружественный интерфейс" пользователя, широко использует подстановки данных и другие эффективные приёмы. Интерфейс работы с экранными формами интуитивно понятен и практически не требует процесса обучения.

Я многому научилась, завершив данный проект. В процессе создания проекта по автоматизации работы системы для менеджмента университета я улучшила свои знания по языку программирования С#, фреймворку ASP.NET Core MVC, системой управления баз данных, узнала много нового об валидации входных данных и их аннотации, языке запросов LINQ, а также узнала об улучшенных технологиях передачи данных из контроллеров в представления.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Албахари, Д. C# 8.0 in a Nutshell / Д. Албахари, - 2020. - 832 стр.
2. Афанасьев, И. PostgreSQL. Полное руководство / И. Афанасьев, - 2023. - 1104 стр.
3. Афанасьев, С.Н. Информационные системы: Учебное пособие. - М.: Финансы и статистика / С.Н. Афанасьев, - 2023. - 320 стр.
4. Ахундова М.А, “Применение автоматизированных информационных технологий в системе менеджмента университета с применением C#” // III Международная научно-практическая конференция «Общество и наука: взаимосвязь», -16-18 ноября, - Порто, Португалия, - стр. 315-317
5. Ахундова М.А, “Важность цифрового правительства в развитии цифровой трансформации мира” // Республиканская научно – практическая конференция «Гейдар Алиев - основатель молодежной политики в Азербайджане», посвященная 100-летию со дня рождения Общенационального Лидера азербайджанского народа Гейдара Алиева, - Баку, Азербайджан: - 23 декабря, - 2023, - стр. 410-413
6. Беляев, А.П. Автоматизация обработки информации: Учебное пособие / А.П. Беляев, – Санкт-Петербург, Россия, - 2021. - 416 стр.
7. Гриффитс, Й. Программирование на C# 8.0 / Й. Гриффитс, - 2021. - 720 стр.
8. Дейл, К. MongoDB: Полное руководство / К. Дейл, Д. Монте, М. Бином, - 2019. - 720 стр.
9. Джозеф, Д. C# 9.0 и .NET 6: Продвинутые темы, шаблоны и архитектура / Д. Джозеф, - 2023. - 960 стр.
10. Джозеф С. ASP.NET MVC 5: Разработка веб-приложений с использованием C# и .NET Framework / С. Джозеф , - 2014. - 1072 стр.
11. Кайзер, К. Oracle 12c: Администратор для начинающих / К.Кайзер, Дж. Прайс, - Санкт-Петербург, Россия,  - 2016. 816 стр.
12. Карвин, Б. SQL Server 2022 Высокопроизводительная настройка / Б. Карвин, - 2022. - 1248 стр.
13. Лазарев, С. C# 10 и .NET 6. Разработка приложений / С. Лазарев, - 2023. - 560 стр.
14. Льюи М. SQL для чайников. 8-е издание / М. Льюи, Д. Морган, - 2023. - 832 стр.
15. Протасов, Е. ASP.NET Core MVC 5 для начинающих / Е. Протасов, - 2022. - 416 стр.
16. Ричтер, Д. CLR с помощью C#. Программирование на платформе Microsoft .NET / Д. Ричтер, - 2018. - 912 стр.
17. Сидоров, А.П. Информационные технологии управления: Учебное пособие / А.П.Сидоров, - 2021. - 320 стр.
18. Смит, К. ASP.NET MVC 5: Создание веб-приложений с использованием C# и .NET Framework / К. Смит , - 2014. - 832 стр.
19. Соколов, А.А. Автоматизированные системы обработки информации: Учебник / А.А. Соколов, В. В. Еремин, - 2022. - 400 стр.
20. Фуллер, Д. Pro C# 7 с .NET Core 2.0 / Д. Фуллер, - 2018. - 1216 стр.
21. Чапман, К. NoSQL для чайников / К. Чапман, - 2019. - 528 стр
22. Щеголев, А.П. Информационные технологии в образовании: Учебное пособие / А.П. Щеголев, - 2021. - 256 стр.

**О Т З Ы В**

нa магистерскую диссертацию магистранта II курса по специализации «Цифровое правительство» Aкадемии Государственного Управления при Президенте Азербайджанской Республики Ахундовой Миниры Азад гызы по теме:«Применение автоматизированных информационных технологий в системе менеджмента университета с применением с C#»

В современном обществе количество информации огромно, и обработка и хранение ее становятся все более сложными с использованием традиционных методов. В связи с этим возникает потребность в автоматизации обработки информации в различных областях человеческой деятельности.

Автоматизированные информационные технологии представляют собой область, где компьютеры и программное обеспечение используются для автоматизации процессов, связанных с сбором, хранением, обработкой, анализом и передачей информации. Эти технологии играют важную роль в современных организациях и предприятиях, способствуя увеличению эффективности и принятию более обоснованных решений. АИТ можно рассматривать как систему автоматизации информационных ресурсов. Автоматизированная информационная система (АИС) представляет собой совокупность программных и аппаратных средств, разработанных для сбора, хранения, обработки и передачи информации внутри организации или для решения конкретных задач.

Использование автоматизированных информационных технологий (АИТ) в управлении университетом с применением C# может существенно улучшить эффективность и точность администрирования университетской деятельности. Автоматизация позволит упростить рутинные задачи, обеспечить более удобный доступ к данным и оптимизировать взаимодействие между учебными заведениями, факультетами, студентами и другими участниками образовательного процесса. Реализовать это можно различными способами, такими как : учет студентов и управление их данными, управление учебными курсами, оптимизация управления финансами, мониторинг и анализ данных, управление ресурсами университета, электронное обучение и дистанционное образование.

Связи с этим тема магистерской диссертации посвященная применению автоматизированных информационных технологий в системе менеджмента университета с применением с C# является актуальной. В работе рассматриваются роль и преимущества автоматизации систем менеджмента, демонстрация актуальности данной темы в системе университета, при помощи современных инструментов автоматизации, таких как языки программирования, библиотеки, платформы, а также рейтинговых СУБД. В рамках темы диссертационной работы были реализованы современные подходы построения системы бизнес логики проекта, такие как вход в систему при помощи аутентификации и идентификации, формирование и фильтрация студентов записанных на определенные предметы, часов преподавателей, свободных и занятых аудиторий, настройка различной информации (создание, модификайия) и т.д...

Можно отметить, что при выпольнении данной работы автор проявила себя с хорошей стороны, т.к. изучая и анализируя указанные литературы и источники она всесторонне охватила важнейшие преимущества применения таких автоматизированных систем, со стороны современных организаций и предприятий, ибо они облегчаеют и ускоряет работу, обеспечивает четкую и слаженную работу, и подготовила законченное научное исследование по данной тематике.

Исходя из вышеизложенного можно сказать, что магистерская диссертация Ахундовой Миниры Азад гызы по теме: «Применение автоматизированных информационных технологий в системе менеджмента университета с применением с C#» отвечает всем требованиям, предъявляемым к магистерским диссертациям и может быть допущена к защите на специализированном совете.

*Научный руководитель: к.п.н., доц. Гаджиева Рена Джавадхан гызы*

*Ученый секретарь: доц. С.Г.Алиев*

*27.05.2024-ый год.*