МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И**

**СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащийся группы ПО-455

В. М. Ермоленко

Шифр 09

2023

# 25 Объясните особенности интерфейса среды Delphi

Embarcadero Delphi, ранее — Borland Delphi и CodeGear Delphi, — интегрированная среда разработки ПО для Microsoft Windows, [macOS](https://ru.wikipedia.org/wiki/MacOS), [iOS](https://ru.wikipedia.org/wiki/IOS) и Android на языке Delphi (ранее носившем название Object Pascal), созданная первоначально фирмой Borland и на данный момент принадлежащая и разрабатываемая Embarcadero Technologies. Embarcadero Delphi является частью пакета Embarcadero RAD Studio и поставляется в четырёх редакциях: Community (распространяется бесплатно и имеет ограниченную лицензию на использование в коммерческих целях), Professional, Enterprise и Architect (подробнее в Таблице редакций ниже). Координирующий офис Embarcadero, ответственный за разработку Delphi, находится в [Торонто](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%80%D0%BE%D0%BD%D1%82%D0%BE), тогда как сама разработка сконцентрирована главным образом в Канаде и Испании[[](https://ru.wikipedia.org/wiki/Delphi_(%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)#cite_note-3)

Среда предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows, Linux, Mac OS X, а также iOS и Android. Благодаря уникальной совокупности простоты языка и генерации машинного кода позволяет непосредственно, и, при желании, достаточно низкоуровнево взаимодействовать с операционной системой, а также с библиотеками, написанными на C/C++. Созданные программы независимы от стороннего ПО, как Microsoft .NET Framework или Java Virtual Machine. Выделение и освобождение памяти управляется в основном пользовательским кодом, что, с одной стороны, ужесточает требования к качеству кода, а с другой — делает возможным создание сложных приложений с высокими требованиями к отзывчивости (работа в реальном времени). До версии 10.4 в кросс-компиляторах для мобильных платформ предусмотрен автоматический подсчёт ссылок на объекты, облегчающий задачу управления их временем жизни.

**Структура среды программирования**

Внешний вид среды программирования Delphi отличается от многих других из тех, что можно увидеть в Windows. К примеру, Borland Pascal for Windows 7.0, Borland C++ 4.0, Word for Windows, Program Manager - это все MDI приложения и выглядят по-другому, чем Delphi. MDI (Multiple Document Interface) - определяет особый способ управления нескольких дочерних окон внутри одного большого окна.

Среда Delphi же следует другой спецификации, называемой Single Document Interface (SDI), и состоит из нескольких отдельно расположенных окон. Это было сделано из-за того, что SDI близок к той модели приложений, что используется в Windows 95.

**Главные составные части среды программирования**

Ниже перечислены основные составные части Delphi:

* Дизайнер Форм (Form Designer)
* Окно Редактора Исходного Текста (Editor Window)
* Палитра Компонент (Component Palette)
* Инспектор Объектов (Object Inspector)
* Справочник (On-line help)

Есть, конечно, и другие важные составляющие Delphi, вроде линейки инструментов, системного меню и многие другие, нужные для точной настройки программы и среды программирования.

Программисты на Delphi проводят большинство времени переключаясь между Дизайнером Форм и Окном Редактора Исходного Текста (которое для краткости называют Редактор). Дизайнер Форм показан на рисунке 1, окно Редактора - на рисунке 2.

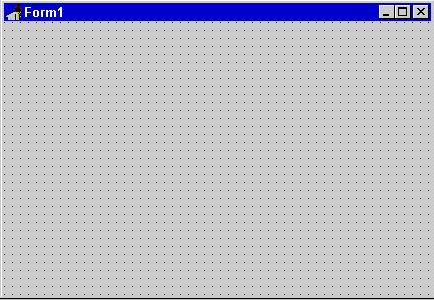


Рисунок 1: Дизайнер Форм - то место, где создается визуальный интерфейс программы.

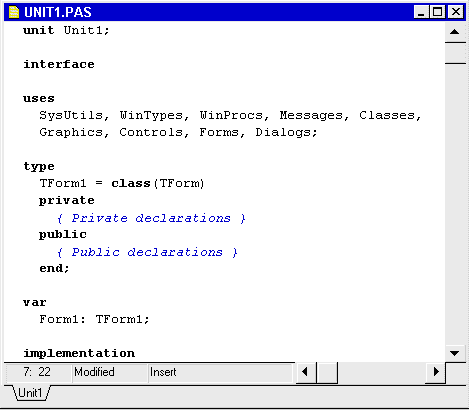


Рисунок 2: В окне Редактора создаетcя логика управления программой.

Дизайнер Форм в Delphi интуитивно понятен и прост в использовании. Дизайнер Форм первоначально состоит из одного пустого окна, которое заполняется всевозможными объектами, выбранными на Палитре Компонент.

Несмотря на всю важность Дизайнера Форм, местом, где программисты проводят основное время, является Редактор. Логика является движущей силой программы и Редактор - то место, где она "кодируется".

Палитра Компонент (рисунок 3) позволяет выбрать нужные объекты для размещения их на Дизайнере Форм. Для использования Палитры Компонент просто первый раз нужно щелкнуть мышкой на один из объектов, и потом второй раз - на Дизайнере Форм. Выбранный объект появится на проектируемом окне и им можно манипулировать с помощью мыши.

Палитра Компонент использует постраничную группировку объектов. Внизу Палитры находится набор закладок - Standard, Additional, Dialogs и т.д. Если Вы щелкнете мышью на одну из закладок, то Вы можете перейти на следующую страницу Палитры Компонент. Принцип разбиения на страницы широко используется в среде программирования Delphi и его легко можно использовать в своей программе. (На странице Additional есть компоненты для организации страниц с закладками сверху и снизу).

http://citforum.ru/programming/32less/les11_03.gif

Рисунок 3: Палитра Компонент - место, где выбираются объекты, которые будут помещены на форму.

После помещения компонента TEdit на форму; можно двигать его с места на место. Также можно использовать границу, прорисованную вокруг объекта для изменения его размеров. Большинством других компонент можно манипулировать тем же образом. Однако, невидимые во время выполнения программы компоненты (типа TMenu или TDataBase) не меняют своей формы.

Слева от Дизайнера Форм можно видеть Инспектор Объектов (рисунок 4). Информация в Инспекторе Объектов меняется в зависимости от объекта, выбранного на форме. Важно понять, что каждый компонент является настоящим объектом и можно менять его вид и поведение с помощью Инспектора Объектов.

Инспектор Объектов состоит из двух страниц, каждую из которых можно использовать для определения поведения данного компонента. Первая страница - это список свойств, вторая - список событий. Если нужно изменить что-нибудь, связанное с определенным компонентом, то обычно делается это в Инспекторе Объектов. К примеру, можно изменить имя и размер компонента TLabel изменяя свойства Caption, Left, Top, Height, и Width.

Можете использовать закладки внизу Инспектора Объектов для переключения между страницами свойств и событий.

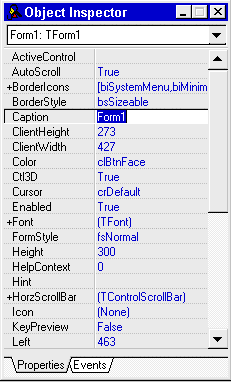


Рисунок 4: Инспектор Объектов позволяет определять свойства и поведение объектов, помещенных на форму.

Страница событий связана с Редактором; если дважды щелкнуть мышкой на правую сторону какого-нибудь пункта, то соответствующий данному событию код автоматически запишется в Редактор, сам Редактор немедленно получит фокус, и сразу же есть возможность добавить код обработчика данного события.

Последняя важная часть среды Delphi - Справочник (on-line help). Для доступа к этому инструменту нужно просто выбрать в системном меню пункт Help и затем Contents. На экране появится Справочник, показанный на рисунке 5.

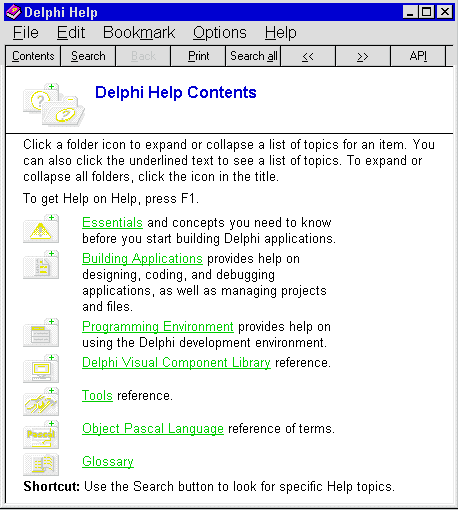


Рисунок 5: Справочник - быстрый поиск любой информации.

Справочник является контекстно-зависимым; при нажатии клавиши F1,мы получим подсказку, соответствующую текущей ситуации. Например, находясь в Инспекторе Объектов, если выбрать какое-нибудь свойство и нажать F1 - получим справку о назначении данного свойства. Если в любой момент работы в среде Delphi возникает неясность или затруднение – можно жать F1 и необходимая информация появится на экране.

**Дополнительные элементы**

В данном разделе внимание фокусируется на трех инструментах, которые можно воспринимать как вспомогательные для среды программирования:

Меню (Menu System)

Панель с кнопками для быстрого доступа (SpeedBar)

Редактор картинок (Image Editor)

Меню предоставляет быстрый и гибкий интерфейс к среде Delphi, потому что может управляться по набору "горячих клавиш". Это удобно еще и потому, что здесь используются слова или короткие фразы, более точные и понятные, нежели иконки или пиктограммы. Можно использовать меню для выполнения широкого круга задач; скорее всего, для наиболее общих задач вроде открытия и закрытия файлов, управления отладчиком или настройкой среды программирования.

SpeedBar находится непосредственно под меню, слева от Палитры Компонент (рисунок 6). SpeedBar выполняет много из того, что можно сделать через меню. Если задержать мышь над любой из иконок на SpeedBar, то появится подсказка, объясняющая назначение данной иконки.

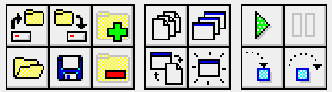


Рисунок 6: SpeedBar находится слева от Палитры Компонент.

**Инструментальные средства**

В дополнение к инструментам, обсуждавшимся выше, существуют пять средств, поставляемых вместе с Delphi. Эти инструментальные средства:

* Встроенный отладчик
* Внешний отладчик (поставляется отдельно)
* Компилятор командной строки
* WinSight
* WinSpector

Данные инструменты собраны в отдельную категорию не потому, что они менее важны, чем другие, но потому, что они играют достаточно абстрактную техническую роль в программировании.

Отладчик позволяет пройти пошагово по исходному тексту программы, выполняя по одной строке за раз, и открыть просмотровое окно (Watch), в котором будут отражаться текущие значения переменных программы.

Встроенный отладчик, который наиболее важен из пяти вышеперечисленных инструментов, работает в том же окне, что и Редактор. Внешний отладчик делает все, что делает встроенный и кое-что еще. Он более быстр и мощен, чем встроенный. Однако он не так удобен в использовании, главным образом из-за необходимости покидать среду Delphi.

**Стандартные компоненты**

На первой странице Палитры Компонент размещены 14 объектов (рисунок 8) определенно важных для использования. Мало кто обойдется длительное время без кнопок, списков, окон ввода и т.д. Все эти объекты такая же часть Windows, как мышь или окно.

Набор и порядок компонент на каждой странице являются конфигурируемыми. Так, Вы можете добавить к имеющимся компонентам новые, изменить их количество и порядок.

http://citforum.ru/programming/32less/les11_08.gif

Рисунок 8: Компоненты, расположенные на первой странице Палитры.

Стандартные компоненты Delphi перечислены ниже с некоторыми комментариями по их применению.

**TMainMenu** позволяет поместить главное меню в программу. При помещении TMainMenu на форму это выглядит, как просто иконка. Иконки данного типа называют "невидимыми компонентом", поскольку они невидимы во время выполнения программы. Создание меню включает три шага: (1) помещение TMainMenu на форму, (2) вызов Дизайнера Меню через свойство Items в Инспекторе Объектов, (3) определение пунктов меню в Дизайнере Меню.

**TPopupMenu** позволяет создавать всплывающие меню. Этот тип меню появляется по щелчку правой кнопки мыши.

**TLabel** служит для отображения текста на экране. Можно изменить шрифт и цвет метки, если дважды щелкнете на свойство Font в Инспекторе Объектов. Это легко сделать и во время выполнения программы, написав всего одну строчку кода.

**TEdit** - стандартный управляющий элемент Windows для ввода. Он может быть использован для отображения короткого фрагмента текста и позволяет пользователю вводить текст во время выполнения программы.

**TMemo** - иная форма TEdit. Подразумевает работу с большими текстами. TMemo может переносить слова, сохранять в Clipboard фрагменты текста и восстанавливать их, и другие основные функции редактора. TMemo имеет ограничения на объем текста в 32Кб, это составляет 10-20 страниц. (Есть VBX и "родные" компоненты Delphi, где этот предел снят).

**TButton** позволяет выполнить какие-либо действия при нажатии кнопки во время выполнения программы. В Delphi все делается очень просто. Поместив TButton на форму, по двойному щелчку можно создать заготовку обработчика события нажатия кнопки. Далее нужно заполнить заготовку кодом.

**TCheckBox** отображает строку текста с маленьким окошком рядом. В окошке можно поставить отметку, которая означает, что что-то выбрано. Например, если посмотреть окно диалога настроек компилятора (пункт меню Options | Project, страница Compiler), то можно увидеть, что оно состоит преимущественно из CheckBox'ов.

**TRadioButton** позволяет выбрать только одну опцию из нескольких. Если открыть диалог Options | Project и выбрать страницу Linker Options, то Вы можете видеть, что секции Map file и Link buffer file состоят из наборов RadioButton.

**TListBox** нужен для показа прокручиваемого списка. Классический пример ListBox'а в среде Windows - выбор файла из списка в пункте меню File | Open многих приложений. Названия файлов или директорий и находятся в ListBox'е.

**TComboBox** во многом напоминает ListBox, за исключением того, что позволяет водить информацию в маленьком поле ввода сверху ListBox. Есть несколько типов ComboBox, но наиболее популярен выпадающий вниз (drop-down combo box), который можно видеть внизу окна диалога выбора файла.

**TScrollbar** - полоса прокрутки, появляется автоматически в объектах редактирования, ListBox'ах при необходимости прокрутки текста для просмотра.

**TGroupBox** используется для визуальных целей и для указания Windows, каков порядок перемещения по компонентам на форме (при нажатии клавиши TAB).

**TPanel** - управляющий элемент, похожий на TGroupBox, используется в декоративных целях. Чтобы использовать TPanel, просто поместите его на форму и затем положите другие компоненты на него. Теперь при перемещении TPanel будут передвигаться и эти компоненты. TPanel используется также для создания линейки инструментов и окна статуса.

**TScrollBox** представляет место на форме, которое можно скроллировать в вертикальном и горизонтальном направлениях.

# 56 Охарактеризуйте сервер приложений.

**Сервер приложений** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *application server*) — это программная платформа, предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения. Сервер приложений действует как набор компонентов, доступных разработчику программного обеспечения через API ([интерфейс прикладного программирования](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), определённый самой платформой.

Для веб-приложений основная задача компонентов сервера — обеспечивать создание динамических страниц. Однако современные серверы приложений включают в себя и поддержку [кластеризации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80_(%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BE%D0%B2)), повышенную [отказоустойчивость](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D0%B0%D0%B7%D0%BE%D1%83%D1%81%D1%82%D0%BE%D0%B9%D1%87%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D1%8C), [балансировку нагрузки](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BD%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8), позволяя таким образом разработчикам сфокусироваться только на реализации [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0).

В случае [Java](https://ru.wikipedia.org/wiki/Java)-сервера приложений сервер приложений ведёт себя как расширенная [виртуальная машина](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) для запуска приложений, прозрачно управляя соединениями с базой данных, с одной стороны, и соединениями с веб-клиентом, с другой.

Сервер приложений (Application Sever) – это [сервер](https://itelon.ru/catalog/oborudovanie/servers/) промежуточного программного обеспечения (ПО, middleware). Это системное ПО, которое располагается между операционной системой (ОС) с одной стороны, внешними ресурсами, например, системой управления базами данных СУБД (DBMS, Database Management System) или Интернет-сервисами, с другой стороны, и приложениями пользователя.

Сервер приложений действует как хост для бизнес-логики пользователя, он также обеспечивает доступ к бизнес-приложениям и задаёт их параметры для пользователя. Сервер приложений должен устойчиво работать независимо от изменений трафика клиентских запросов, отказов оборудования и ПО, распределённого характера масштабных приложений, а также возможной разнородности форматов данных и ресурсов их обработки.

Внешние ресурсы, например, СУБД и Интернет-сервисы, предоставляют [веб-серверы](https://itelon.ru/solution/server_web/) (Web Server). Они отвечает на запросы пользователя по доставке контента.

Серверы приложений иногда путают с веб-серверами. У них есть общие функции, но есть и много различий. Понимание этих различий поможет правильно сконфигурировать [программное обеспечение](https://itelon.ru/catalog/oborudovanie/software/) и инфраструктуру оборудования для нужд предприятия.

## Различия между серверами приложений и веб-серверами

|  |  |  |
| --- | --- | --- |
| **Параметр сравнения** | **Веб-сервер** | **Сервер приложений** |
| **Основная цель** | Хостинг сайтов и ответы на простые веб-запросы | Хостинг приложений и обеспечение сложных взаимосвязей бизнес-логики |
| **Тип контента** | Доставка только статического контента HTML | Доставка как статического, так и динамического контента |
| **Протоколы** | Только HTTP/HTTPS | HTTP/HTTPS и другие протоколы |
| **Соединение с приложениями** | Нет | Да |
| **Подключения к базами данных** | К статическим базам данных | К базам данных приложений |
| **Типичные клиенты** | Веб-браузеры | Веб- и мобильные приложения, а также веб-браузеры |
| **Многопотоковая обработка** | Не поддерживается | Поддерживается параллельная обработка многих запросов |
| **Потребление ресурсов** | Трафик не потребляет много ресурсов | Процессы с интенсивным потреблением ресурсов |
| **Контейнеры** | Только веб-контейнеры | Веб-контейнеры (сервлеты, JSP, JSF, веб-сервисы), контейнеры клиентских приложений (DI, безопасность) |
| **Ёмкость** | Низкая | Высокая |
| **Результат запроса** | Гипертекстовый документ, отображающий информацию в браузере | Файлы, содержащие данные, по требованию клиента |

## Что такое сервер приложений?

Сервер приложений (Application Server, App-Server) – это программный комплекс, предназначенный для доставки контента и средств его представления для клиентских приложений. Клиентами могут быть веб-приложения, браузеры или мобильные приложения.

Серверы приложений предоставляют для клиентов бизнес-логику, то есть, преобразуют данные в динамический контент и обеспечивают функционал приложений. Примеры такого контента:

* Результаты транзакций;
* Поддержка принятия решений;
* Аналитика в реальном времени, и др.

Сервер приложений – это связующее звено между клиентом и программным кодом физического сервера. Типичные задачи сервера приложений:

* Управление транзакциями;
* Безопасность;
* Внедрение зависимости DI (Dependency injection);
* Одновременность исполнения процессов (Concurrency).

Серверы приложений также обрабатывают такие процессы, как кластеризация, исправление отказов и балансировка нагрузки (рисунок 9).

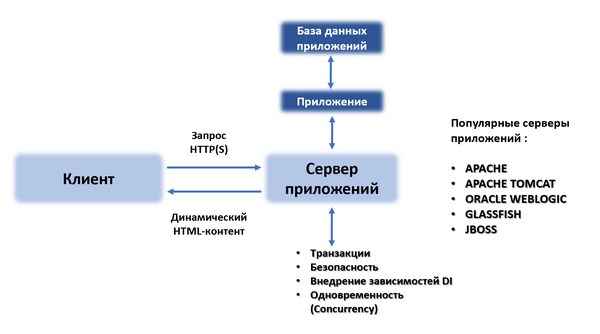


Рисунок 9. Сервер приложений.

## Что общего у веб-сервера и сервера приложений

Если в качестве основного приложения клиента выступает веб-браузер, то различия между двумя типами серверов размываются. Большинство веб-серверов имеют плагины на основе скриптов (ASP, JSP, JSF, PHP, Perl, и пр.), которые позволяют генерировать динамический контент.

Поскольку в сценариях применения у веб-серверов и серверов приложений много общего, то наиболее популярные серверы являются гибридами этих двух типов. Гибридное решение, совмещающее свойства обеих серверов, обеспечивает максимальную скорость и функциональность системы.

## Наиболее популярные серверы приложений

**Apache Tomcat**

Apache Tomcat – контейнер сервлетов с открытым исходным кодом на языке Java. Tomcat позволяет запускать веб-приложения и содержит ряд программ для автоматического конфигурирования и часто используется вместе с конфигурационным файлом Apache HTTPD (Apache Hypertext Transfer Protocol Server daemon). Tomcat может исполнять Java-сервлеты, доставлять клиентам страницы в кодах Java Server Page, и может обслуживать приложения Java EE (Java Enterprise Edition).

**Oracle WebLogic**

Сервер Oracle WebLogic – сервер для распределённых приложений с использованием стандартов Java EE. Он полностью интегрирован с продуктами и облачными сервисами Oracle.

**Glassfish**

Glassfish – сервер приложений с открытым кодом на Java EE, который поддерживает Java-сервлеты, а также спецификацию написания и поддержки серверных компонентов с бизнес-логикой EJB (Enterprise JavaBeans).

**JBoss**

JBoss – сервер приложений с открытым кодом для создания, развёртывания и хостинга приложений на языке Java. JBoss может работать на разных платформах и в любой операционной системе с поддержкой Java.

## Какой сервер приложений будет наиболее подходящим?

Знание различий между сервером приложений и веб-сервером помогает выбрать сервер для того или иного использования.

* Если нужно обслуживать только веб-страницы со статическим контентом, то лучше использовать веб-сервер;
* Если приложения требуют наличия JSP (JavaServer Pages) и сервлетов, лучше использовать простой сервер приложений, типа Jetty или Apache Tomcat;
* Если приложения содержат много сложных функций, таких как распределённые транзакции и мессенджеры, то лучше использовать полнофункциональные серверы приложений, такие как JBoss или Oracle WebLogic.

Другим подходом может быть добавление функционала в веб-сервер при помощи плагинов. В этом случает, веб-сервер может использовать технологию программирования на стороне сервера (server-side), такую как скрипты CGI, JSP, сервлеты, ASP (Active Server Pages) или JavaScript на стороне сервера.

## Использование обоих типов сервера в одной системе

Часто и веб-сервер, и сервер приложений, развёртывают в одной системе. Это даёт возможность предоставлять клиентам как статический, так и динамический контент. В этом случае, веб-сервер становится подсистемой сервера приложений и все их сервисы работают на одной и той же программно-аппаратной платформе.

Преимуществом такого подхода является более высокая производительность системы. В каждом типе сервера максимально используются их преимущества. Простые веб-запросы будут сразу же обрабатываться веб-сервером и при этом не будет снижаться производительность сервера приложений.

Например, на сайте Интернет-магазина должна предоставляться информация о ценах в реальном времени. Обычно на сайте также есть форма для приобретения товара. Когда пользователь посылает запрос, веб-страница магазина ищет актуальную цену и выдаёт результат в виде HTML-страницы. Эту функциональность можно обеспечить как при помощи сервера приложений, так и при помощи веб-сервера с соответствующими плагинами. Возможно несколько сценариев.

### Сценарий 1. Использование только веб-сервера с плагинами

Веб-сервер предоставляет функционал Интернет-магазина:

* Сервер получает запрос и передаёт его в соответствующую программу на стороне сервера;
* Эта программа ищет актуальные цены в базе данных или в обычном файле;
* Программа формулирует ответ в форме HTML;
* Веб-сервер посылает запрос обратно в веб-браузер клиента.

### Сценарий 2. Использование как веб-сервера, так и сервера приложений

Сервер приложений хранит бизнес-логику для поиска цены. Веб-сервер делегирует ему генерацию ответа, скрипт вызывает сервис поиска в сервере приложений, и затем формулирует ответ HTML.

Размещение логики поиска цены в сервере приложений позволяет использовать её различными частями приложения. В первом сценарии сервис поиска цены не может повторно использоваться, поскольку данные встроены в HTML-страницу (рисунок 10).

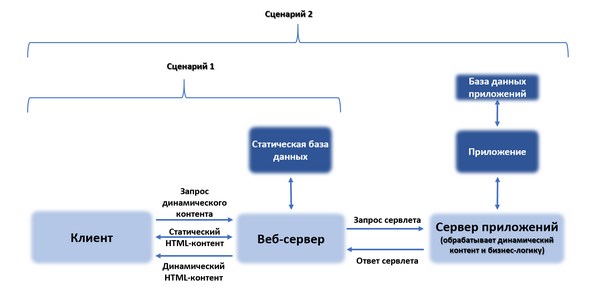


Рисунок 10. Использование как веб-сервера, так и сервера приложений.

## Заключение

Пересечение функций веб-сервера и сервера приложений означает, что каждый сценарий применения может иметь несколько решений. Можно применять веб-серверы и серверы приложений отдельно, а можно использовать их комбинацию.

Однако, не каждая конфигурация будет равноценной по параметрам работы и потреблению ресурсов, хотя и будет выполнять возложенные на неё функции. Знание различий между двумя типами серверов поможет сэкономить средства, облегчить масштабирование системы и повысить производительность.

74 Магазин CD»

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server по № варианта (задание 61-90), содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.

**Решение**

Создадим форму и разместим на ней компонент TabControl в котором разместим 2 вкладки Внешний вид формы представлен на рисунке 11. На каждой из них разместим соответствующие элементы управления: поля ввода данных и кнопки для добавления, редактирования и удаления записи, а на итоговой вкладке кнопки для запуска заданий.

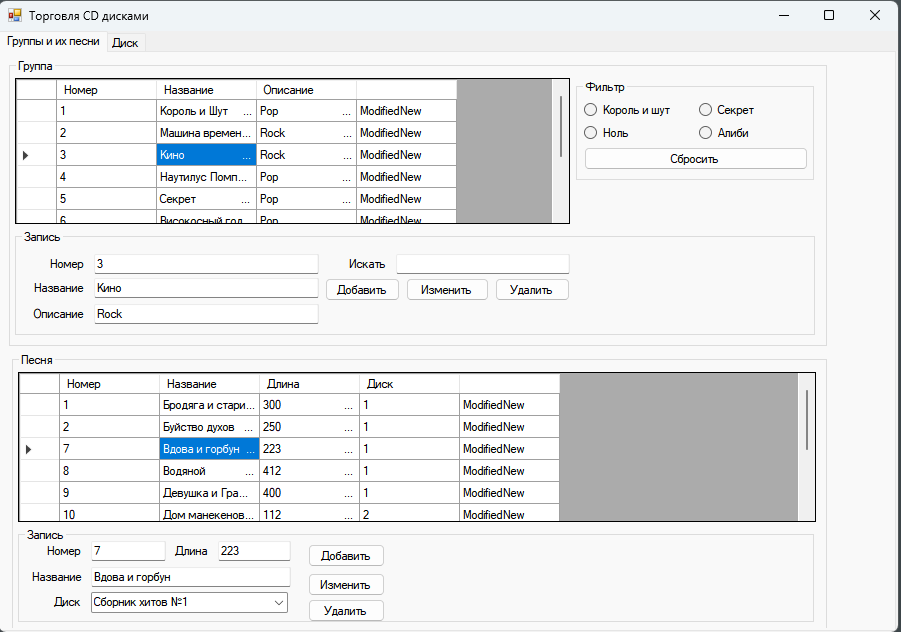


Рисунок 11 Внешний вид основной формы

Для примера добавления строки в таблицу внешней формой создадим вторую форму и разместим на ней элементы управления (рисунок 12)

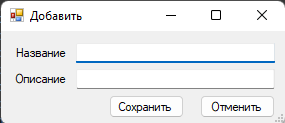


Рисунок 12 Внешний вид формы добавления

Заполним таблицы данными, у нас это будут таблицы: Диск, Песня и Группа. Внешний вид приложения представлен на рисунках 13, 14 и 15.

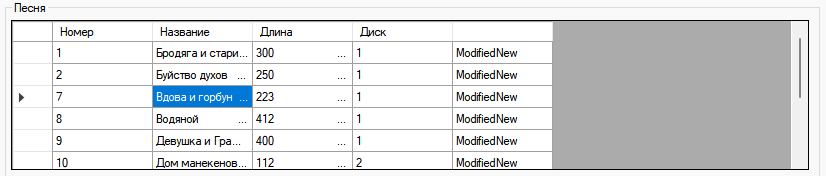


Рисунок 13 Приложение таблица Песня

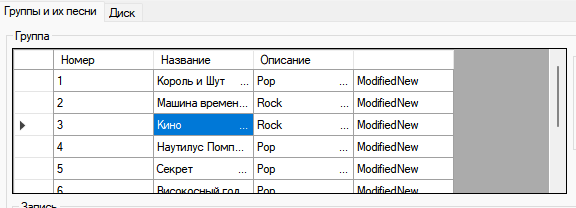


Рисунок 14. Приложение таблица Группа

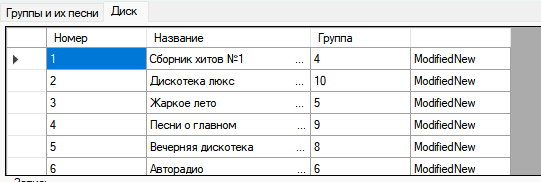


Рисунок 15. Приложение таблица Диск.

На рисунке 16 представлен внешний вид приложения с открытой вкладкой для выполнения заданий. Выполнено задание вывода данных из двух таблиц.

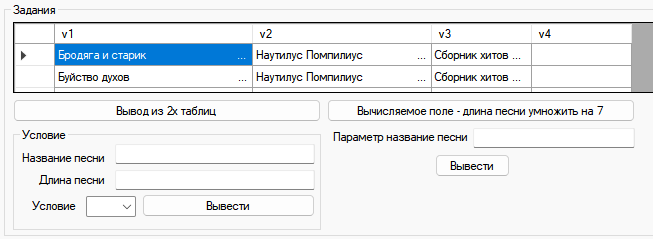


Рисунок 16 Приложение вкладка выполнения заданий

На рисунке 17 представлен результат выполнения действий над полями таблицы, длина песни умножается на 7 и выводится в третьем столбце.

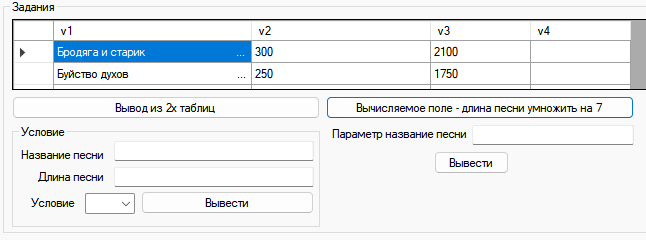


Рисунок 17 Результат выполнения задания вычисляемых полей

На рисунке 18 представлен вид приложения с выполненным заданием выбора песни по названию, либо по длине.

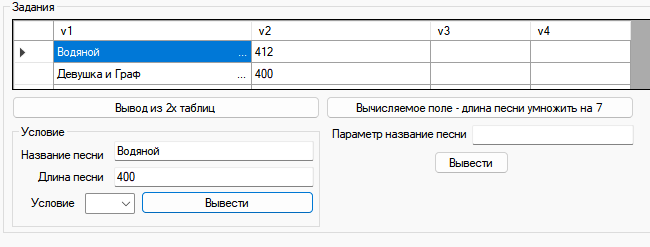


Рисунок 18 Результат выполнения задания Условие

На рисунке 19 представлен вывод задания с параметром, в качестве параметра название песни

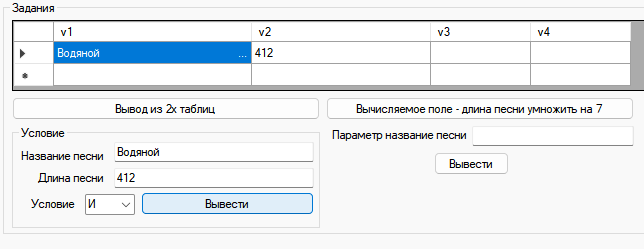
****

Рисунок 19 Результат выполнения задания Параметр

На рисунке 20 представлен результат выполнения задания Фильтр

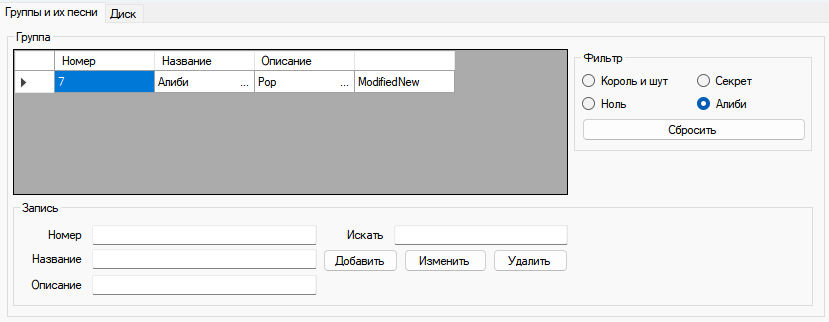
****

Рисунок 20результат выполнения задания Фильтр

На рисунке 21 представлен результат выполнения задания Поиск

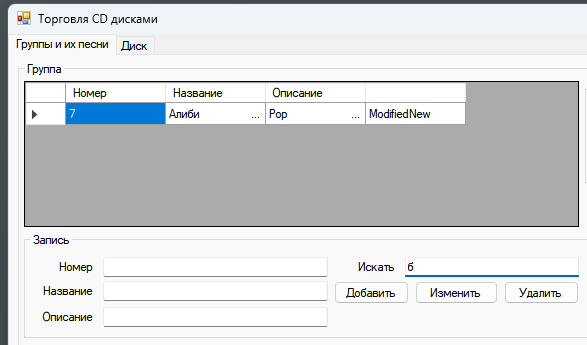


Рисунок 21результат выполнения задания Поиск

**Код программы, Главная Форма Приложения**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Disk

{

enum rowState

{

Existed,

New,

Modified,

ModifiedNew,

Deleted

}

public partial class Form1 : Form

{

DataBase database = new DataBase();

int selectedRow;

public Form1()

{

InitializeComponent();

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Disk]");

}

private void Form1\_Load(object sender, EventArgs e)

{

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

//refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Disk]");

}

private void CreateColumns()

{

dataGridView1.Columns.Add("ID", "Номер");

dataGridView1.Columns.Add("Name", "Название");

dataGridView1.Columns.Add("Descr", "Описание");

dataGridView1.Columns.Add("IsNew", String.Empty);

dataGridView2.Columns.Add("ID", "Номер");

dataGridView2.Columns.Add("Name", "Название");

dataGridView2.Columns.Add("Len", "Длина");

dataGridView2.Columns.Add("IDDisk", "Диск");

dataGridView2.Columns.Add("IsNew", String.Empty);

dataGridView3.Columns.Add("ID", "Номер");

dataGridView3.Columns.Add("Name", "Название");

dataGridView3.Columns.Add("IDGroup", "Группа");

dataGridView3.Columns.Add("IsNew", String.Empty);

dataGridView4.Columns.Add("Col1", "v1");

dataGridView4.Columns.Add("Col2", "v2");

dataGridView4.Columns.Add("Col3", "v3");

dataGridView4.Columns.Add("Col4", "v4");

}

private void ReadSinglRow(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), rowState.ModifiedNew);

}

private void ReadSinglRow2(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetInt32(3), rowState.ModifiedNew);

}

private void ReadSinglRow3(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetInt32(2), rowState.ModifiedNew);

}

private void ReadSinglRow4(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetString(0), record.GetString(1), record.GetString(2), record.GetString(3));

}

private void refreshDG(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Group]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dgw, reader);

}

reader.Close();

}

private void refreshDG2(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Song]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow2(dgw, reader);

}

reader.Close();

}

private void refreshDG3(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Disk]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow3(dgw, reader);

}

reader.Close();

}

private void refreshDG4(DataGridView dgw, string s)

{

dgw.Rows.Clear();

string querystr = s;// $"Select \* from [Disk]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dgw, reader);

}

reader.Close();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Form211 f\_add = new Form211();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where concat(Name,Descr) like '%" + textBox4.Text + "%'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton1\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Король и шут'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

refreshDG(dataGridView1);

}

private void button3\_Click\_1(object sender, EventArgs e)

{

Form211 f\_add = new Form211();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void dataGridView1\_CellClick\_1(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void radioButton2\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Секрет'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton4\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Алиби'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton3\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Ноль'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"delete from [Group] where ID={textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"update [Group] set Name='{textBox2.Text}' , Descr = '{textBox3.Text}' where ID = {textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "databaseDataSet1.Group". При необходимости она может быть перемещена или удалена.

this.groupTableAdapter.Fill(this.databaseDataSet1.Group);

// TODO: данная строка кода позволяет загрузить данные в таблицу "databaseDataSet1.Disk". При необходимости она может быть перемещена или удалена.

this.diskTableAdapter.Fill(this.databaseDataSet1.Disk);

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

textBox5.Text = row.Cells[0].Value.ToString();

textBox6.Text = row.Cells[1].Value.ToString();

textBox7.Text = row.Cells[2].Value.ToString();

comboBox1.SelectedIndex = Int32.Parse(row.Cells[3].Value.ToString())-1;

}

}

private void button5\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Song] (Name,Len, IdDisk) values ('{textBox6.Text}' , '{textBox7.Text}',{comboBox1.SelectedIndex+1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button7\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Song] where ID={textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

textBox8.Text = row.Cells[0].Value.ToString();

textBox9.Text = row.Cells[1].Value.ToString();

comboBox2.SelectedIndex = Int32.Parse(row.Cells[2].Value.ToString())-1;

}

}

private void button8\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Disk] (Name,IdGroup) values ('{textBox9.Text}' ,{comboBox2.SelectedIndex + 1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button10\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Disk] where ID={textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button6\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Song] set Name='{textBox6.Text}' , Len = '{textBox7.Text}', IdDisk = '{comboBox1.SelectedIndex+1}' where ID = {textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button9\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Disk] set Name='{textBox9.Text}' , IdGroup = '{comboBox2.SelectedIndex+1}' where ID = {textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button11\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select s.name as col1, g.name as col2, d.name as col3,'' as col4\r\nfrom [Disk] d, [Song] s, [Group] g\r\nwhere d.idgroup=g.id and s.iddisk=d.id";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button12\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, CAST(len\*7 AS varchar(5)) as col3, '' as col4 from [Song] s";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button13\_Click(object sender, EventArgs e)

{// условие

dataGridView4.Rows.Clear();

string s = "OR";

if (comboBox3.SelectedIndex == 0) {

s = "AND"; }

string n = textBox10.Text;

string l = textBox11.Text;

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, '' as col3, '' as col4 from [Song] s where Name='{n}' {s} Len='{l}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button14\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string n = textBox12.Text;

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, '' as col3, '' as col4 from [Song] s where Name='{n}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

}

class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=Database;Integrated Security=True");

public void openConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()

{

return sqlConnection;

}

}

}

**Список использованных источников**

1. Дейт, К.Дж. Введение в системы баз данных/ К.Дж.Дейт. – М.: Вильямс, 2018
2. Дюбуа, Поль. MySQI / Поль Дюбуа. – 3-е изд. – М.: Издательский дом "Вильямс", 2007. - 816 с.: ил.
3. Колисниченко, Д.Н. Профессиональное программирование на PНP / Д.Н. Колисниченко. – Санкт-Петербург: БХВ-Петербург, 2007. –416 с.
4. Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас. – Минск:РИПО, 2016
5. Роб, П. Системы баз данных: проектирование, реализация и управление / П.Роб, К.Коронел. – 5-е изд., перераб. и доп.: пер. с англ. – Санкт-Петербург: БХВ-Петербург, 2004. - 1040 с.: ил
6. Фаронов, В.В. Программирование баз данных в Delphi / В.В.Фаронов. – 2-е изд. – СанктПетербург.: Питер, 2004. – 459 с.
7. Хернандес, М.Дж. SQL-запросы для простых смертных: практическое руководство по манипулированию данными в SQL / М.Дж.Хернандес, Дж.Л.Вьескас. – Москва: Лори, 2000. - 473 с.