|  |  |  |
| --- | --- | --- |
| Департамент образования Ярославской области  Государственное профессиональное образовательное автономное учреждение  Ярославской области  «ЯРОСЛАВСКИЙ ПРОМЫШЛЕННО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ  им. Н.П. ПАСТУХОВА» | | |
| **Отчет По учебной практике**  **по профессиональному модулю ПМ.07**  **Соадминистрирование и автоматизация баз данных и серверов** | | |
| ОУП ПМ.05.09.02.07.18ИП2.19 | | |
|  | | |
|  | | |
|  | Студент  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.О. Чеготова  «\_\_\_\_» декабрь 2021 г. |
|  | Преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.В. Маянцева  «\_\_\_\_» декабрь 2021 г. |
| 2021 | | |

**Содержание**

[Практическая работа №1 3](#_Toc89324634)

[Практическая работа №2 10](#_Toc89324635)

[Практическая работа №3 12](#_Toc89324636)

[Практическая работа №4 17](#_Toc89324637)

# **Практическая работа №1**

**Тема**

Создание базы данных.

**Цель работы**

По разработанной на УП.05 ERD-диаграмме реализовать базу данных, создать скрипт и разместить его в репозитории.

**Теоретическая часть**

ERD - модель данных, позволяющая описывать концептуальные схемы предметной области. ERD-модель используется при высокоуровневом проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

**Задание 1.**

По разработанной на УП.05 ERD-диаграмме с помощью СУБД MS SQL реализуйте базe данных. Заполните базу данных первичными данными.

Тема: АИС «Планировщик задач». Этот программный продукт направлен на учет и выполнение поставленных руководителем задач.

[**Ненормализованная форма или нулевая нормальная форма (UNF)**](https://info-comp.ru/zero-normal-form)

Перед тем как переходить к процессу нормализации и приведению базы данных к определённой нормальной форме, необходимо привести базу данных к табличному виду, но так, чтобы он отвечал базовым принципам реляционной теории.

По реляционной теории строки в таблицах не должны быть пронумерованы, т.е. порядок строк не имеет значения, так же как не имеет значения порядок столбцов. Т.е. например, если мы поменяем порядок столбцов, или порядок строк, ничего измениться не должно, это не должно ни на что повлиять. Таким образом, по реляционной теории мы не можем обратиться к определённой строке или столбцу по ее номеру.

**Первая нормальная форма (1NF) базы данных**

После того как таблицы приведены к правильному табличному виду, мы можем начинать процесс нормализации.

Требование первой нормальной формы (1NF) заключается в том, чтобы таблицы соответствовали реляционной модели данных и соблюдали определённые реляционные принципы.

Таким образом, чтобы база данных находилась в 1 нормальной форме, необходимо чтобы ее таблицы соблюдали следующие реляционные принципы:

* В таблице не должно быть дублирующих строк.
* В каждой ячейке таблицы хранится атомарное значение (одно не составное значение).
* В столбце хранятся данные одного типа.
* Отсутствуют массивы и списки в любом виде.

**Вторая нормальная форма (2NF) базы данных**

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

* Таблица должна находиться в первой нормальной форме
* Таблица должна иметь первичный ключ
* Все неключевые столбцы таблицы должны зависеть от полного ключа (*в случае если он составной*)

Если ключ составной, т.е. состоит из нескольких столбцов, то все остальные неключевые столбцы должны зависеть от всего ключа, т.е. от всех столбцов в этом ключе. Если какой-то атрибут (столбец) зависит только от одного столбца в ключе, значит, база данных не находится во второй нормальной форме.

**Третья нормальная форма (3NF) базы данных**

Требование третьей нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость.

**Транзитивная зависимость** – это когда неключевые столбцы зависят от значений других неключевых столбцов.

Иными словами, неключевые столбцы не должны пытаться играть роль ключа в таблице, т.е. они действительно должны быть неключевыми столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Данная база данных представлена уже в третьей нормальной форме (рис.1).

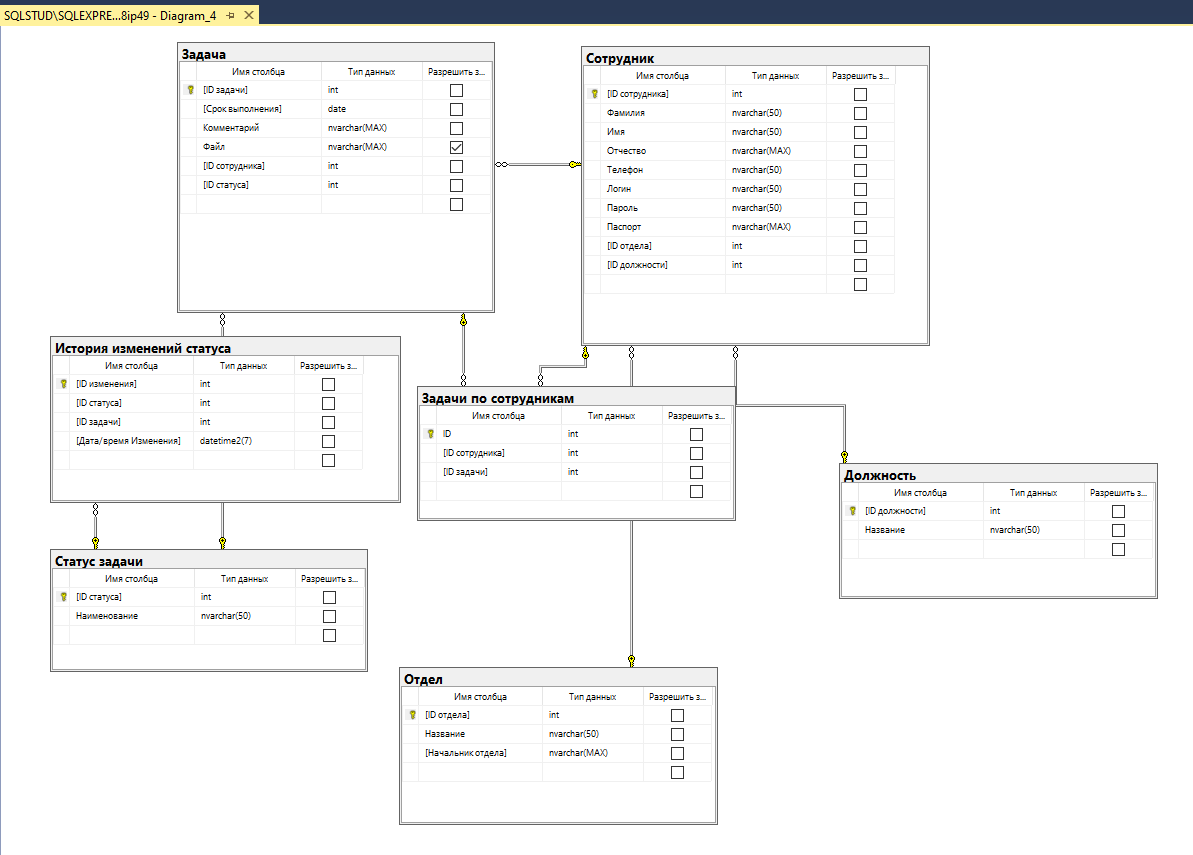


Рисунок 1. Схема баз данных в среде MS SQL

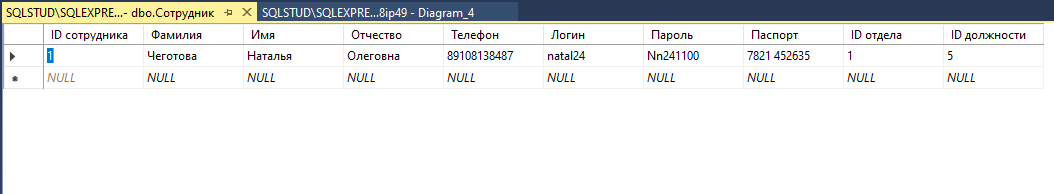


Рисунок 2. Первичное заполнение информацией таблицы «Сотрудник»

**Задание 2.**

**Создание скрипта базы данных с помощью команды «Создать скрипт»**

Для создания первого скрипта необходимо развернуть узел «Базы данных», далее открыть окно редактора запросов. Данный параметр создает скрипт только параметров конфигурации, с помощью него изменить название всей базы данных.

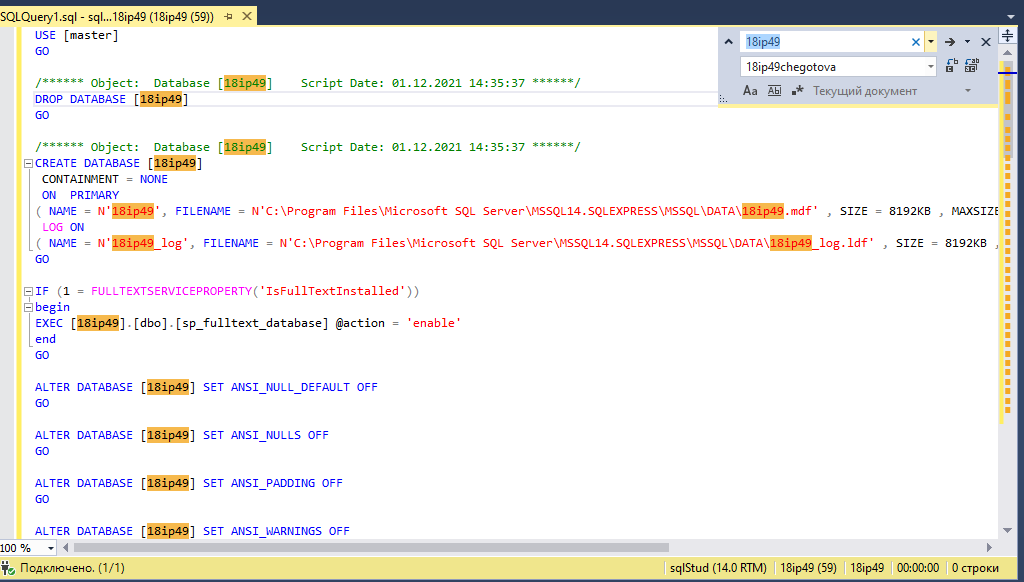


Рисунок 3. Изменения названия базы данных и ее элементов

Пример срабатывания скрипта:

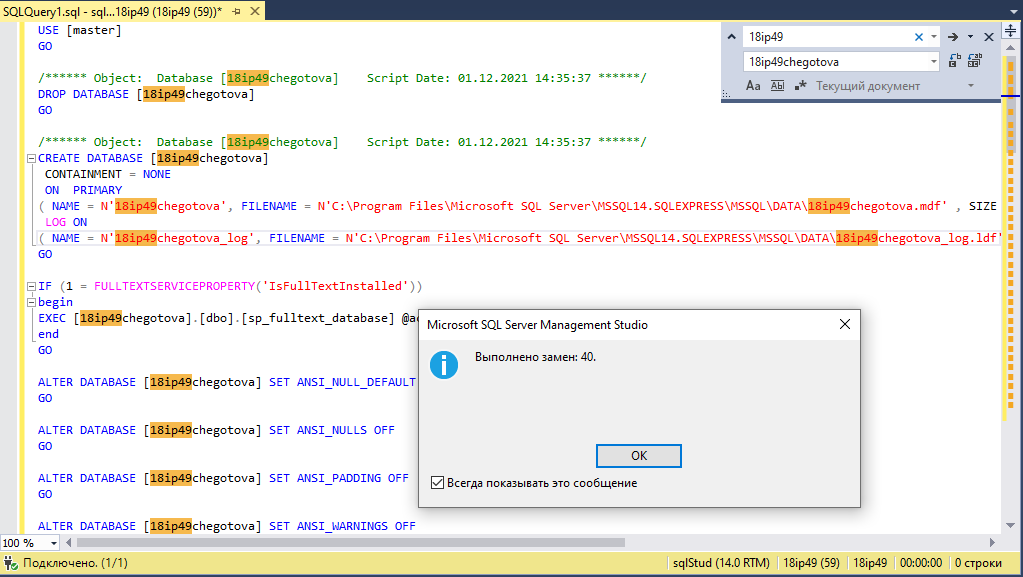


Рисунок 4. Успешное изменение и сохранение базы данных

**Создание скрипта базы данных с помощью команды «Создать скрипты»**

Создание скрипта с базой данных и сохранение его. Данный скрипт будет сохранять базу данных. Это удобно для восстановления базы данных.

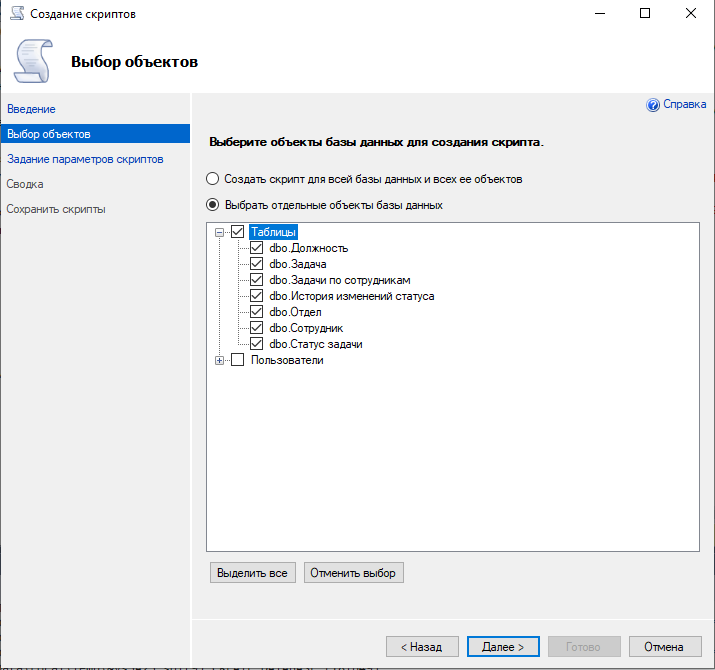


Рисунок 5. Выбор объектов для сохранения

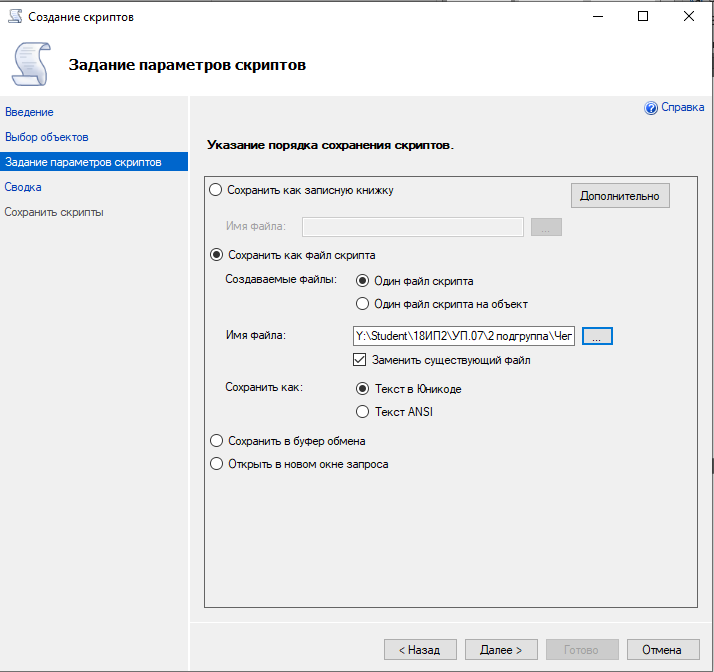


Рисунок 6. Задание параметров скрипта

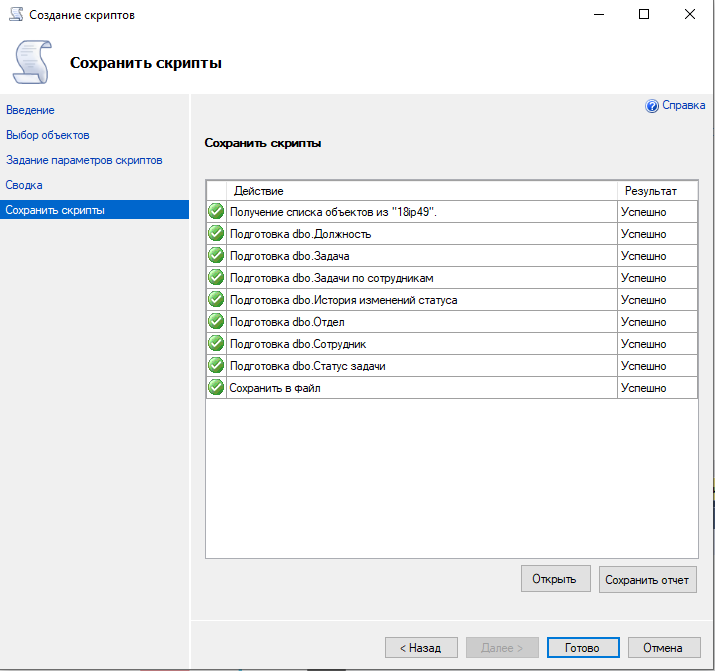


Рисунок 7. Сохраненный скрипт базы данных

После процесса создания скрипта в указанной папке появляется файл со скриптом. Далее этот скрипт можно импортировать/экспортировать.

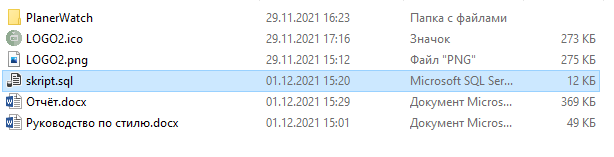


Рисунок 8. Файл со скриптом

**Вывод**

Благодаря данному практическому заданию, были вспомнены материалы по созданию базы данных в MS SQL. Так же были изучены методы создания скриптов для облегчения работы со схемой базы данных.

**Контрольные вопросы**

1. База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
2. Проектирование базы данных осуществляется в три этапа:
3. концептуальное проектирование
4. логическое проектирование
5. физическое проектирование
6. Таблицы являются объектами, которые содержат все данные в базах данных. В таблицах данные логически организованы в виде строк и столбцов по аналогии с электронной таблицей. Каждая строка представляет собой уникальную запись, а каждый столбец — поле записи.
7. Первичный ключ — в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа. Если в отношении имеется единственный потенциальный ключ, он является и первичным ключом.
8. Внешний ключ — понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных.
9. Всего существует 3 типа связей:
10. Один к одному;
11. Один ко многим;
12. Многие ко многим.
13. Целостность информации — термин в информатике (криптографии, теории телекоммуникаций, теории информационной безопасности), означающий, что данные не были изменены при выполнении какой-либо операции над ними, будь то передача, хранение или отображение.
14. Нормализация заключается в основном в разработке схемы базы данных таким образом, чтобы избежать дублирования и избыточности данных. Если какая-то часть данных дублируется в нескольких местах базы данных, существует риск того, что она будет обновлена в одном месте, но не в другом, что приведет к повреждению данных.

# **Практическая работа №2**

**Тема**

Размещение файлов в репозитории.

**Цель работы**

Разместить скрипт с базой данных, руководство по стилю, логотип и иконку в удаленный репозиторий проекта.

**Задание 1**

Создайте руководство по стилю для своего будущего проекта.

**Руководство по стилю**

**Использование логотипа (см. рисунок 9)**

Все экранные формы пользовательского интерфейса должны иметь заголовок с логотипом (в ресурсах). Не искажайте логотип (не изменяйте изображение, его пропорции, цвет).

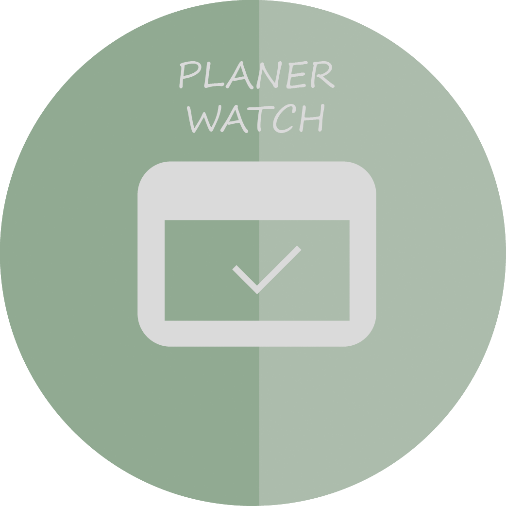
Также для приложений должна быть установлена иконка. 

Рисунок 9. Логотип

**Шрифт**

Используйте шрифт Times new roman.

**Цветовая схема**

В качестве основного фона используйте, цвет RGB (255,255,255), в качестве дополнительного: RGB (216,218,218).

Для актирования внимания пользователей на целевое действие интерфейса используйте цвет RGB (159,181,153)

|  |  |  |
| --- | --- | --- |
| Основной фон | Дополнительный фон | Акцентирование внимания |
| RGB  (255,255,255) | RGB  (216,218,218) | RGB  (159,181,153) |
|  |  |  |

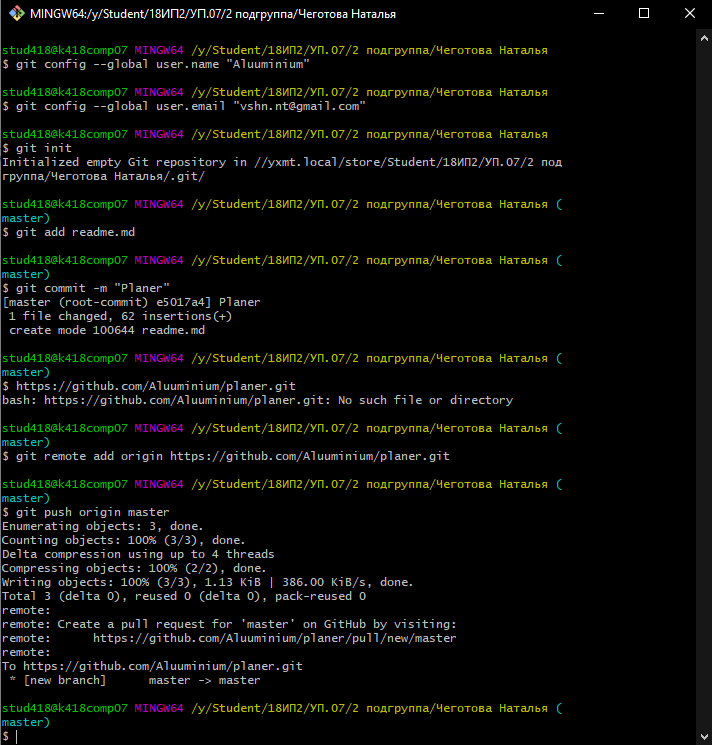
**Задание 2**

**Теоретическая часть:**

Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

**Практическая часть:**

GitHub — один из сервисов для использования системы контроля версий Git.

Импортируем файлы через GitHub. 

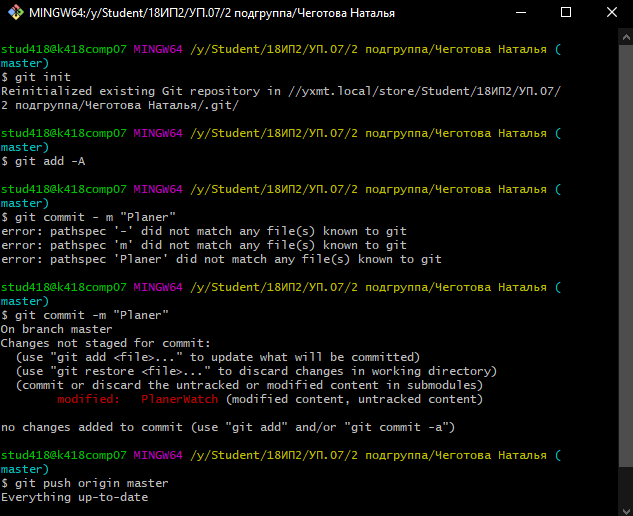


Рисунок 10. Пошаговое импортирование файлов в GitHub

Далее файлы (логотип и руководство по стилю) отобразились в самом репозитории.

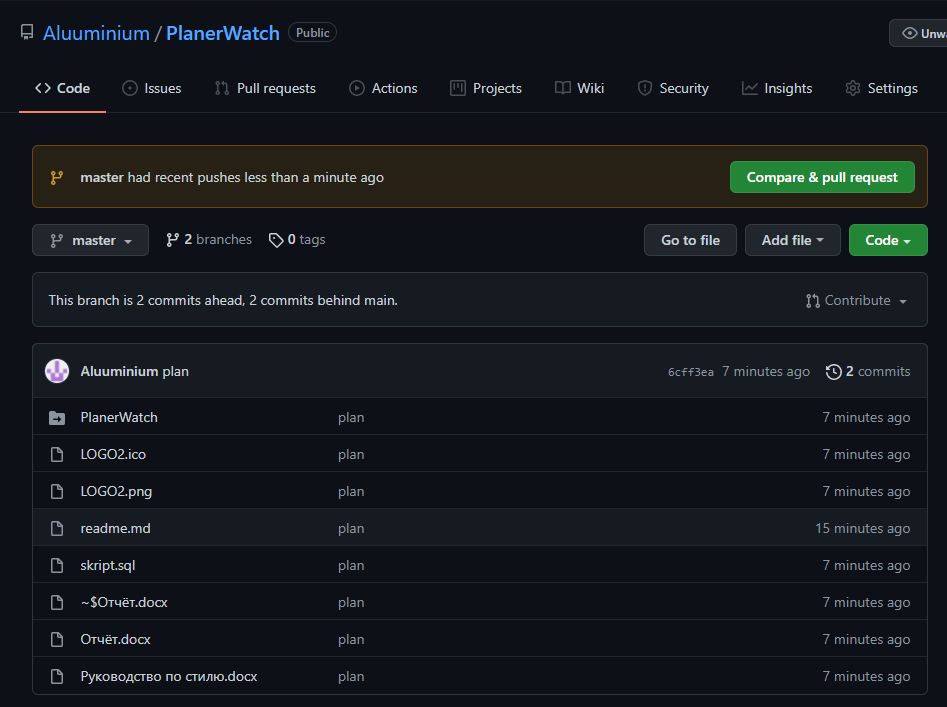


Рисунок 11. Итог импортирования в GitHub

**Вывод**

В результате выполнения данного практического задания, для приложения были подобранны цвета для реализации программного продукта, а так же были подобраны иконка и логотип, которые так же важны для программы.

**Контрольные вопросы**

1. Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
2. Программное обеспечение контроля версий отслеживает все вносимые в код изменения в специальной базе данных. При обнаружении ошибки разработчики могут вернуться назад и выполнить сравнение с более ранними версиями кода для исправления ошибок, сводя к минимуму проблемы для всех участников команды.
3. Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.
4. Установи GIT.
5. Настрой Git, так Git связывает Ваш аккаунт и Ваши изменения ...
6. Настрой Git для Windows.
7. Создай локальный GIT репозиторий для своего продукта
8. Подключи стандартные каталоги
9. Создай файл в локальном репозитории
10. Свяжи новый файл с удаленным репозиторием
11. Отправь код на удаленный репозиторий
12. Установить программу git на вашей системе.
13. Настроить программу и проверить её работоспособность локально
14. Зарегистрировать ваш аккаунт на GitHub
15. Создать локальный репозиторий.
16. Написать файл README.MD (при необходимости).
17. В случае, если вы начинаете проект, создать удаленный репозиторий.
18. Фиксировать изменения локально
19. Отправлять изменения на GitHub.
20. Зарегистрировать аккаунты разработчиков вашего проекта (при необходимости) и выдать им ссылку на проект.
21. git checkout имя\_файла - отменить изменения в файле
22. git reset --hard - отменить все локальные изменения
23. git stash - положить изменения в тайник
24. git stash pop - вытащить изменения из тайника (работает как стек)
25. git log -n 5 - вывести название последних пяти коммитов
26. Ряд сервисов предоставляют хостинг для git-репозиториев, среди наиболее известных — GitHub, Codebase, SourceForge, SourceHut, Gitea, Bitbucket, GitLab.
27. Пример: для добавления файлов используется команда:

git add readme

После выполнения команды, файл readme будет добавлен в систему управления версий (конечно, если он уже был, то этого в проекте). При добавлении файла генерируется хеш значение, которое выглядит примерно так:

9f2422325cef705b7682418d05a538d891bad5c8

Добавленные файлы хранятся в папке. git/objects/xx/yyyyyyyy, при этом первые 2 цифры хеша ипользуются для указания директории, а остальное хеш значение является именем файла.

# **Практическая работа №3**

**Тема**

Создание проекта в Visual Studio. Настройка проекта. Создание словаря стилей. Создание Git репозитория с помощью Visual Studio.

**Цель работы**

Создать и настроить проект в Visual Studio, создать репозиторий, используя средства Visual Studio.

**Теоретическая часть**

Словарь ресурсов представляет собой просто XAML-документ, который всего лишь хранит необходимые ресурсы.

Чтобы использовать словарь ресурсов, его необходимо объединить с коллекцией ресурсов. Это можно делать в каком-то конкретном окне.

Для начала внутри каталога мы создаем 3 папки и загружаем и их картинки для будущей нашей работы.

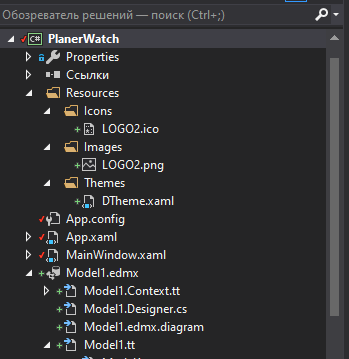


Рисунок 12. Обозреватель решений

После создания словаря, мы задает стиль кнопке и DataGrid. Этот файл будет хранится в каталоге «*Themes»*. Среда разработки создаст файл с разметкой, изменим еще пару свойств у кнопки. Зададим определенные значения для ширины и высоты элемента, а также сделаем бесцветными границы кнопки. Еще, для примера, определим стиль для элемента DataGrid.

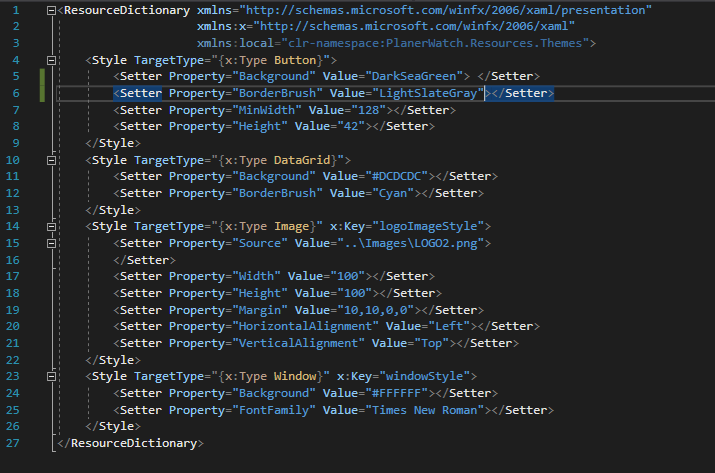


Рисунок 13. Полный код разметки стиля иконки

Как видно на скриншоте, для атрибута Source прописываем относительный путь для справочника ресурсов. Далее добавляем в разметку словарь ресурсов.

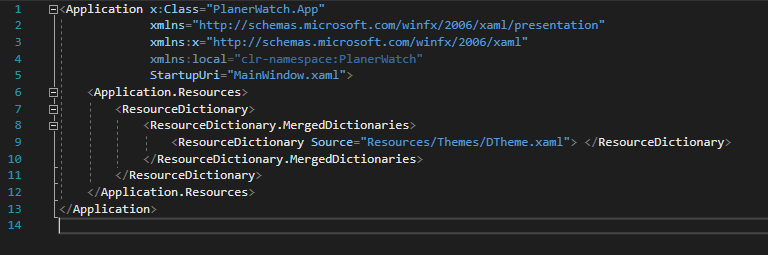


Рисунок 14. Добавление в разметку словаря ресурсов.

К кнопке автоматически применились описанные в справочнике стили. Добавим DataGrid на форму.



Рисунок 15. Работа стилей

Далее пропишем код к логотипу приложения для этого нам нужно прописать код на основной форме. Код будет выглядеть так (Рисунок 13).

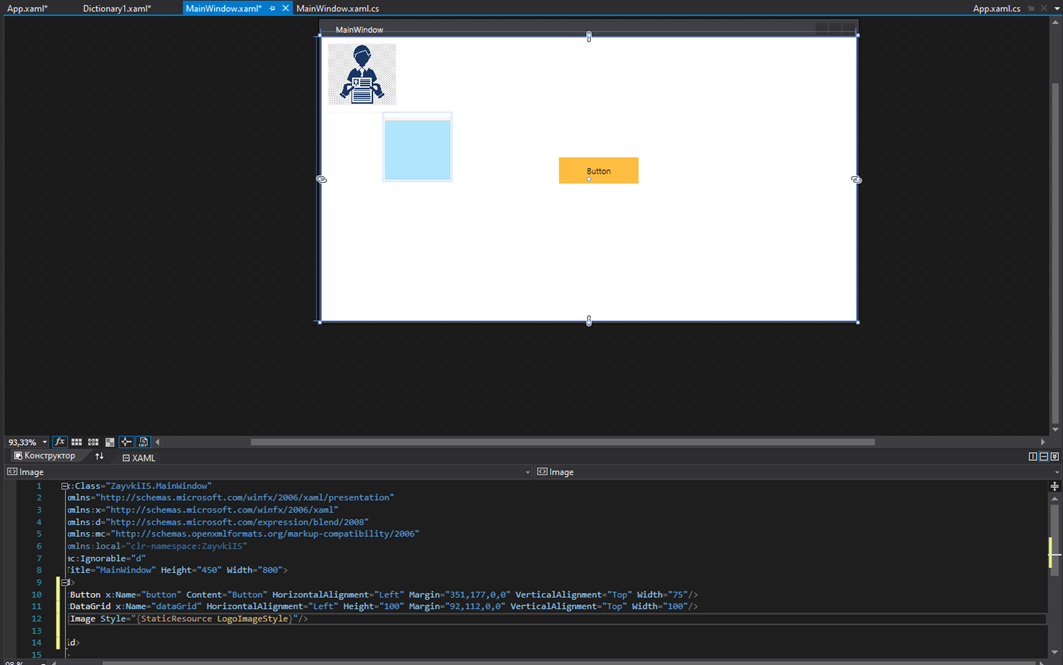


Рисунок 16. Отображение логотипа

Далее задаем параметры для главного окна.

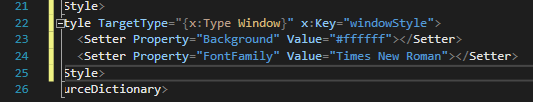


Рисунок 17. Код стиля для окна

После написания данного кода видим, что шрифт на кнопке изменился, а значит код работает.

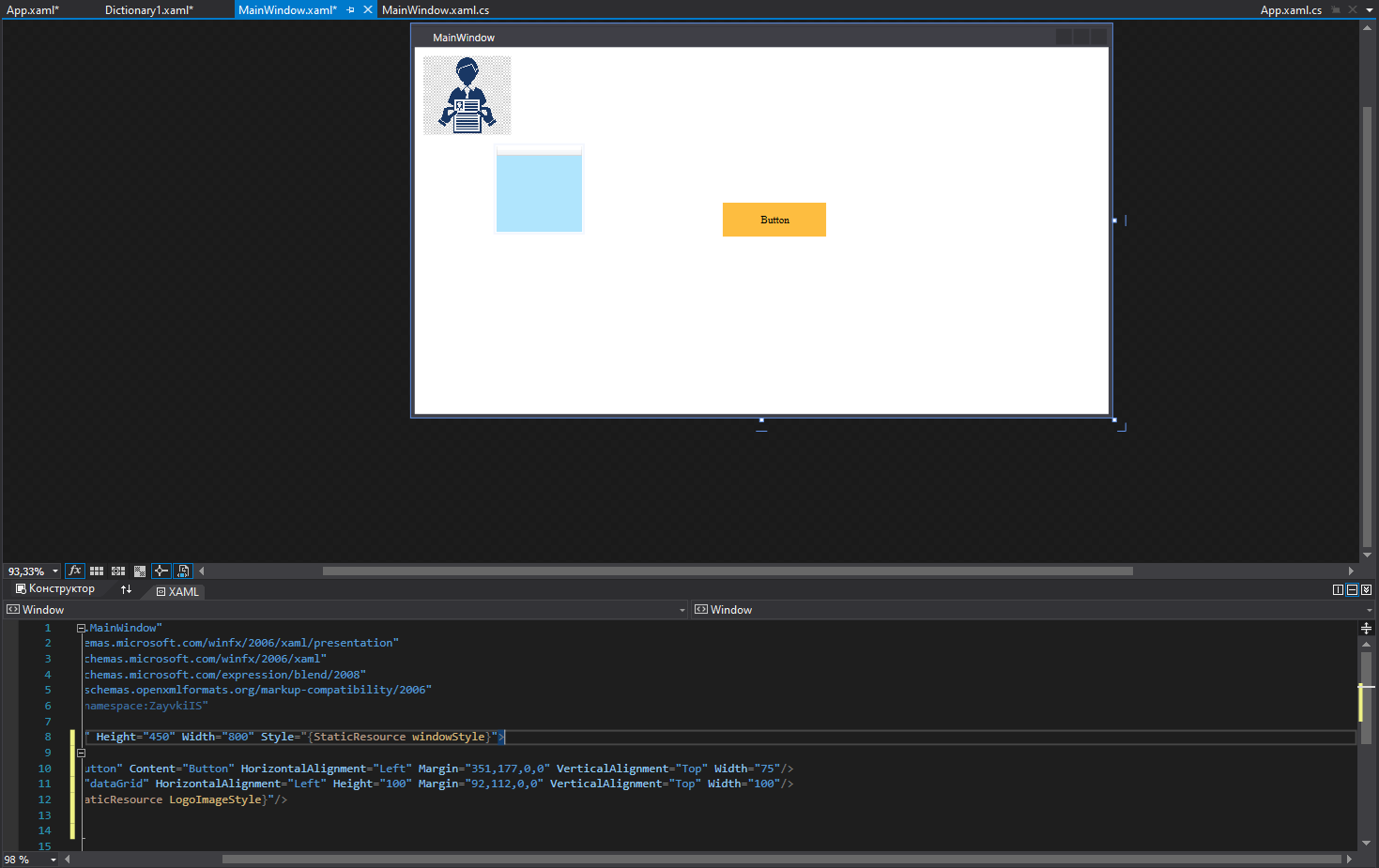


Рисунок 18. Измененный стиль окна

**Задание 2**

**Создание Git репозитория в Visual Studio.**

Далее нам нужно перекинуть наш проект в Visual studio.

Рисунок

**Вывод:**

Благодаря данному практическому заданию, были приобретены знания в создании словаря ресурсов, а также сохранении проекта в Git.

**Контрольные вопросы**

1. Чтобы разделить ресурсы между множеством проектов можно создать словарь ресурсов. Словарь ресурсов представляет собой просто XAML-документ, который всего лишь хранит необходимые ресурсы.
2. Вы можете определить стиль непосредственно на самом элементе. Использование тега Resources внутри элемента позволяет охватить дочерние элементы этого элемента и сам элемент. Следующим шагом будет определение стилей контейнера с помощью его "ресурсов" (Window — это наш главный контейнер на странице).
3. Как у всякого ресурса, у стиля устанавливается ключ с помощью атрибута x:Key , а также свойство TargetType - оно указывает на тип.
4. <Resources> <Style x:Key="baseStyle" TargetType="FrameworkElement">

# **Практическая работа №4**

**Тема**

Подключение базы данных. Авторизация в системе.

**Цель работы**

По заданной схеме данных разработать базу данных, создать программное приложение с возможностью авторизации в системе.

**Практическая работа**