**ПРАВИТЕЛЬСТВО САНКТ-ПЕТЕРБУРГА**

**КОМИТЕТ ПО НАУКЕ И ВЫСШЕЙ ШКОЛЕ**

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Санкт-Петербургский технический колледж управления и коммерции»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование курса:** | | | | *МДК.01.01. Разработка программных модулей* | | | | | | | |
| **Тема:** | *«Разработка инф. Системы для мини-вики по Naruto”* | | | | | | | | | |
|  | *для “NarutoViki”* | | | | | | | | | |
| **Специальность:** | | *09.02.07 Информационные системы* | | | | | | | | |
|  | | *и программирование* | | | | | | | | |
|  | | | (шифр и наименование специальности) | | | | | | | |
|  | | |  | | | | | | | |
| **Студент** | *группы 9ПО-43* | | | |
|  |  | | *Забавин Д.С.* | |
|  | ***(подпись)*** | | | ***(И.О. Фамилия)*** |
| **Оценка** |  | | | |
|  | *(проставляется цифрами и в скобках прописью)* | | | |
| **Преподаватель** |  | *А. В. Смирнова* | | |
|  | ***(подпись)*** | ***(И.О. Фамилия)*** | | |

Санкт-Петербург

2023 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc102579488)

[1.ПОСТАНОВКА ЗАДАЧИ 5](#_Toc102579489)

[2. СРЕДСТВА РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ 7](#_Toc102579490)

[3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ 8](#_Toc102579491)

[3.1 Инфологическое проектирование базы данных 9](#_Toc102579492)

[3.1.1 Определение сущностей 10](#_Toc102579493)

[3.2 Логическое (даталогическое) проектирование 11](#_Toc102579494)

[3.2.1 Преобразование ER-диаграммы в схему базы данных 12](#_Toc102579495)

[3.3 Физическое проектирование 13](#_Toc102579496)

[3.4. Создание таблиц базы данных 18](#_Toc102579497)

[3.5. Ввод тестовых данных 22](#_Toc102579498)

[3.6. Создание просмотров 25](#_Toc102579499)

[3.7. Создание триггеров 28](#_Toc102579500)

[3.8. Создание хранимых процедур 32](#_Toc102579501)

[4. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 34](#_Toc102579502)

[4.1. Разработка рабочего модуля ИС 34](#_Toc102579503)

[4.2. Демонстрация работы приложения 35](#_Toc102579504)

[ЗАКЛЮЧЕНИЕ 39](#_Toc102579505)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 40](#_Toc102579506)

**ВВЕДЕНИЕ**

Эффективная работа с информацией требует систематизированного подхода к её хранению. В связи с этим было решено создать базу данных, предназначенную для хранения ключевой информации о персонажах, техниках и мире Наруто. В бумажной форме подобная информация становится менее доступной и ненадежной, поэтому создание централизованной базы данных кажется более логичным и эффективным шагом.

Данный курсовой проект ориентирован на разработку информационной системы "NarutoViki", которая предоставляет удобный доступ к информации о персонажах, их техниках, связанных мирах и сюжетах вселенной Наруто. Подобное приложение позволяет пользователям быстро получить доступ к различным аспектам этой фантастической вселенной, предоставляя данные о количестве персонажей, их особенностях, а также видео материалах, связанных с этим миром.

Целью этого курсового проекта является автоматизация хранения и обработки информации в информационной системе "NarutoViki". Проект включает в себя создание базы данных и разработку пользовательского интерфейса, обеспечивающего удобный доступ к данным.

Для успешного выполнения поставленной цели требуется решить следующие задачи:

1. Провести анализ предметной области мира Наруто.
2. Разработать концептуальные и логические модели базы данных для "NarutoViki".
3. Создать серверную часть информационной системы для обеспечения доступа к базе данных.
4. Разработать клиентское приложение, обеспечивающее удобный интерфейс для взаимодействия пользователей с данными о мире Наруто.

Этот проект направлен на создание удобной информационной платформы "NarutoViki", позволяющей пользователям получать доступ к разнообразным аспектам вселенной Наруто через удобный и структурированный интерфейс.

**ПОСТАНОВКА ЗАДАЧИ**

Проект "NarutoViki" представляет собой информационную систему, посвященную миру "Наруто". Целью данного курсового проекта является создание приложения мини-вики о мире "Наруто" на языке C# с использованием среды разработки Microsoft Visual Studio и СУБД MS SQL Server. Система должна обладать следующими функциональными возможностями:

1. **Авторизация и регистрация пользователей:**
   * Создание новых учетных записей пользователей.
   * Вход в систему для авторизованных пользователей.
2. **Информация о персонажах:**
   * Просмотр базовой информации о персонажах мира "Наруто" (имя, клан, ранг и другие характеристики).
   * Поиск информации о конкретных персонажах.
   * Отображение видео с персонажами.
3. **Информация о техниках:**
   * Просмотр информации о различных техниках из мира "Наруто".
   * Фильтрация и поиск техник по типу, подтипу, рангу и другим характеристикам.
4. **Информация о хвостатых:**
   * Отображение базовой информации о хвостатых существах (имя, количество хвостов и т.д.).

**Функциональности приложения:**

1. **Окна просмотра информации:**
   * Представление информации в виде таблицы с понятными заголовками.
   * Возможность поиска информации о персонажах и техниках.
   * Возможность сортировки записей по различным полям.
2. **Окна ввода данных:**
   * Поля для ввода данных (при добавлении новой информации).
   * Пояснения к каждому полю для уточнения назначения.
   * Кнопка "Сохранить" для сохранения введенных данных.

Данное приложение будет создано с использованием средств языка C# и разработано в Microsoft Visual Studio с возможностью подключения к СУБД MS SQL Server для хранения данных о персонажах, техниках и хвостатых существах.

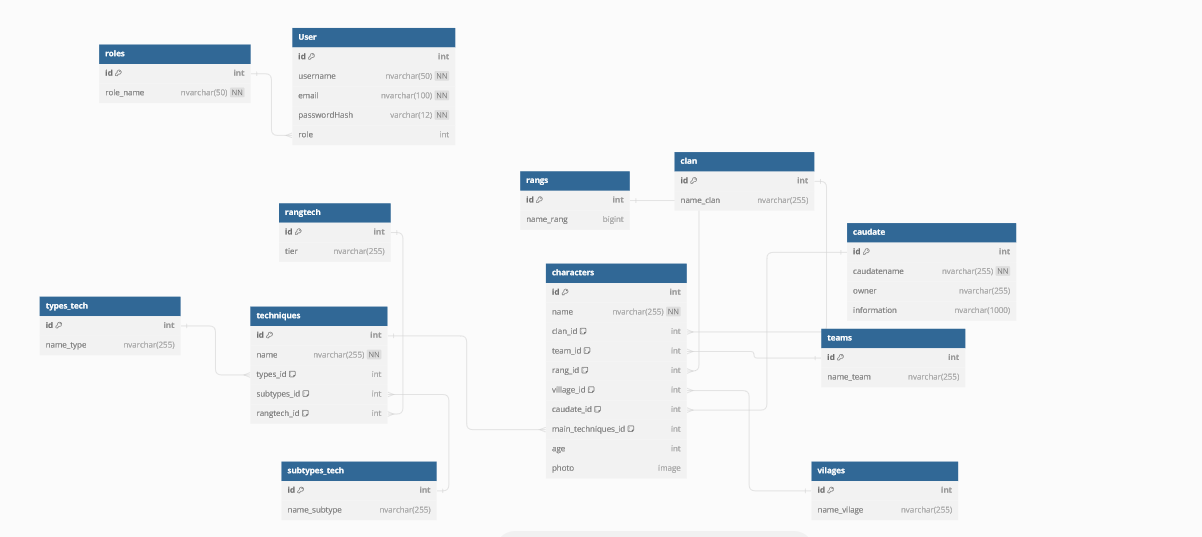


Рисунок 1 – Физическая диаграмма пользования

## 2. СРЕДСТВА РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Правильно спроектированная БД должна удовлетворять следующим требованиям:

1. Минимальная избыточность;
2. Непротиворечивость;
3. Целостность данных;
4. Независимость данных;
5. Возможность ведения (добавления и удаления) и актуализации (корректировки, модификации) данных;
6. Безопасность и секретность;
7. Высокая производительность.

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой MS SQL Server. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, используемые вашими приложениями, а также создавать запросы и скрипты.

Создание многопользовательской БД и получение одновременного доступа нескольких пользователей к общей базе данных возможно как в рамках модели сервера баз данных (предпочтительно), так и модели удаленного доступа к данным

3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ

Проектирование БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить следующие этапы проектирования;

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Проектирование инфологической модели предметной области - частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ЕR-модели.

3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели данных.

4. Физическое проектирование БД, то есть выбор эффективного размещения БД па внешних носителях для обеспечения наиболее эффективной работы системы.

Оценка проекта БД:

1. сокращение избыточности данных,
2. уменьшение затрат на многократное обновление полей,
3. устранение возможных противоречий информации из-за хранения в различных местах и получение «чистого» проекта.

Этапы проектирования БД:

1. Концептуальное проектирование - сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:

* обследование предметной области, изучение ее информационной структуры
* выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами
* моделирование и интеграция всех представлений

По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели "сущность-связь".

2. Логическое проектирование - преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

3. Физическое проектирование - определение особенностей хранения данных, методов доступа и т.д.

## 3.1 Инфологическое проектирование базы данных

Инфологический аспект имеется в виду при рассмотрении вопросов, связанных со смысловым содержанием данных, с их семантикой, независимо от способа представления в памяти компьютерной системы.

Концептуальное (инфологическое) проектирование – построение сематической модели предметной области, то есть информационной модели наиболее высоко уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Термины «сематическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контекст равноправно могут использоваться слова «модель базы данных» и «модель предметной области», поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

* Описание информационных объектов или понятий предметной области и связей между ними. Описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.
* Описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

### 3.1.1 Определение сущностей

Этапом инфологического проектирования является определение информации о предметной области, которая будет представлена в базе данных. Для приложения "NarutoViki" были выделены следующие сущности:

1. **Роль пользователя (Roles):** Определяет роли, которые пользователь может иметь в системе, например, администратор или обычный пользователь.
2. **Пользователь (User):** Содержит информацию о зарегистрированных пользователях, включая логин, адрес электронной почты и зашифрованный пароль.
3. **Технология рангов (RangTech):** Хранит информацию о различных уровнях или категориях рангов, используемых в мире Наруто.
4. **Ранги (Rangs):** Содержит данные о конкретных рангах или статусах персонажей или команд в мире Наруто.
5. **Хвостатые (Caudate):** Информация о существах с хвостами в мире Наруто, их названия, владельцах и других деталях.
6. **Команды (Teams):** Данные о командах или группировках персонажей.
7. **Деревни (Villages):** Содержит информацию о различных деревнях в мире Наруто.
8. **Клан (Clan):** Информация о кланах или семьях персонажей.
9. **Подтипы техник (Subtypes\_Tech):** Хранит подкатегории или разновидности техник в мире Наруто.
10. **Типы техник (Types\_Tech):** Содержит типы или категории техник, используемых в мире Наруто.
11. **Техники (Techniques):** Информация о различных техниках, их названиях, типах, подтипах и связанных с ними рангах.
12. **Персонажи (Characters):** Данные о персонажах, их именах, принадлежности к клану или команде, ранге, деревне, информация о хвостатых, основные техники, возраст и фотографии.

3.1.2. Связи между сущностями

В реляционной БД отношение между объектами (сущностями) определяет связи между таблицами.

Всего существует 3 типа связей:

* Один к одному. Связь один к одному образуется, когда ключевой столбец (идентификатор) присутствует в другой таблице, в которой тоже является ключом либо свойствами столбца задана его уникальность (одно и тоже значение не может повторяться в разных строках).
* Один ко многим. В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.
* Многие ко многим. Если нескольким записям соответствует несколько записей из другой таблицы, то связь называется «многие ко многим».

В результате концептуального проектирования, исходя из особенностей предметной области и выявленных в ней сущностей была сформирована концептуальная модель данных, представленная на рисунке 2.

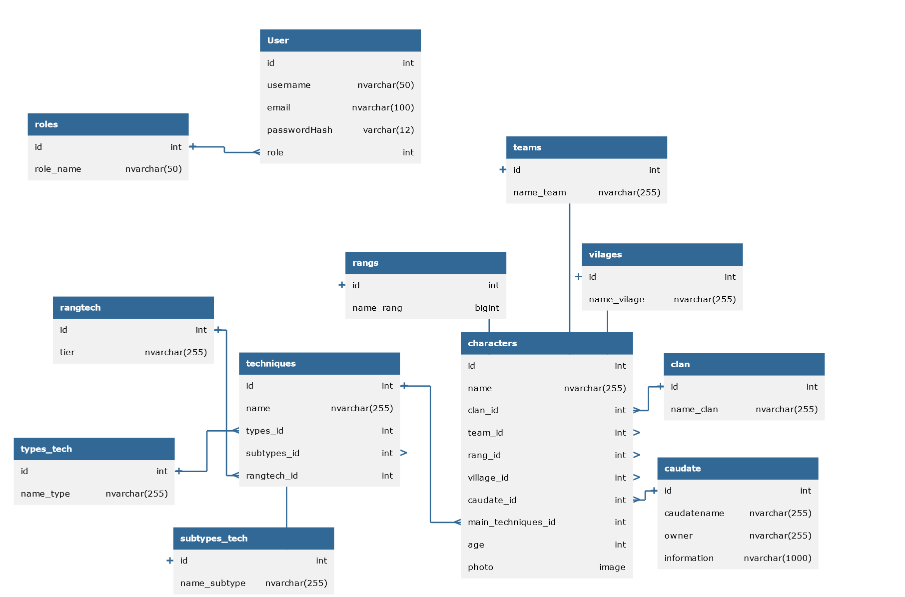


Рисунок 2 – Концептуальная модель данных

## 3.2 Логическое (даталогическое) проектирование

Логическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик (например, длины поля). Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. В данном проекте будет использоваться реляционная модель данных, так как именно она поддерживается СУБД MS SQL Server.

## 3.2.1 Преобразование ER-диаграммы в схему базы данных

Преобразование ER–диаграммы в схему реляционной БД выполняется путем преобразования сущностей в отношения по определенным правилам. При этом, данные правила учитывают не только степень связи, но степень участия сущностей в формируемой связи. Исходя из этого получаем три правила для связи 1:1, два для связи 1:М, одно для связи М:М. Применяя данные правила последовательно к каждой паре сущностей, мы получим схему БД (ее логическую модель).

Связь типа 1:М (один-ко-многим) между отношениями реализуются через внешний ключ. Ключ вводиться для того отношения, к которому осуществляется множественная связь. Внешнему ключу должен соответствовать первичный или уникальный ключ основного (родительского) отношения.

Применив к диаграмме на рисунке 2 правила преобразования концептуальной модели данных в реляционную, нами была получена логическая модель, которая представлена на рисунке 3. На ней представлены отношения, связи между ними (с указанием степени связи), первичные ключи сформированных отношений.

Вероятно, при изучении логической модели могут возникнуть сомнения в организации связи между Пользователями и Администратором. В данной базе данных выбрано такое решение исходя из того, что , к системе смогут подключаться только зарегистрированные клиенты.

## 3.3 Физическое проектирование

На этапе физического проектирования для каждого отношения выбирается тип данных из тех, что предлагает конкретная СУБД.

Результат физического проектирования представлен на рисунке 4.

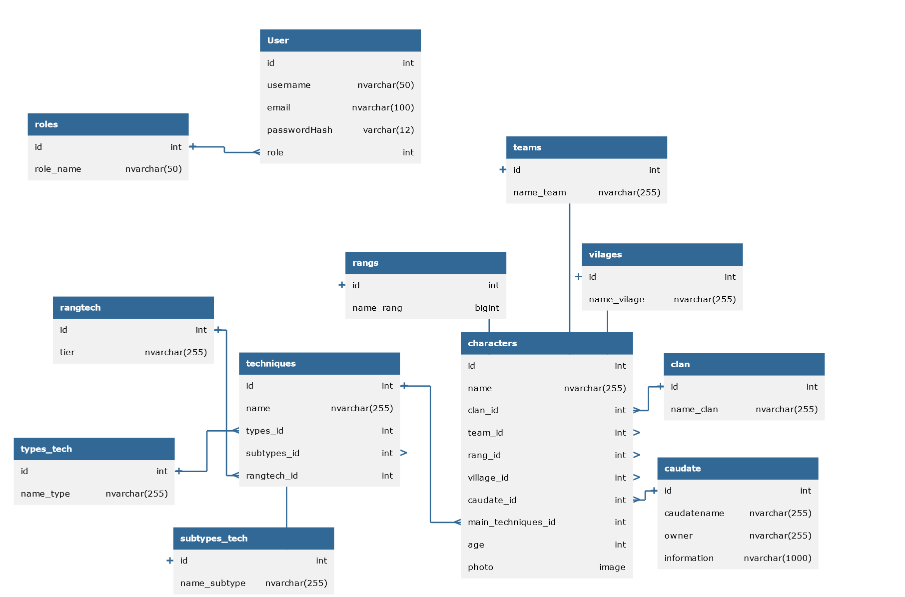


Рисунок 4 – Физическая модель базы данных

Ниже представлена структура каждой из сформированных таблиц с указанием их полей, их типов данных и обязательности заполнения.

Структура таблицы «Роли пользователей» представлена на рисунке 5.

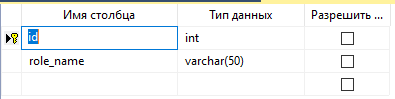


Рисунок 5 – Структура таблицы «Роли пользователей»

**Рисунок 7 – Структура таблицы «Пользователь»**



Таблица "Пользователь" хранит информацию о зарегистрированных пользователях, включая их логины, адреса электронной почты и зашифрованные пароли.

**Рисунок 8 – Структура таблицы «Технология рангов»**

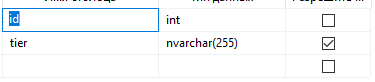


Таблица "Технология рангов" содержит данные о различных уровнях или категориях рангов, используемых в мире Наруто.

**Рисунок 9 – Структура таблицы «Ранги»**

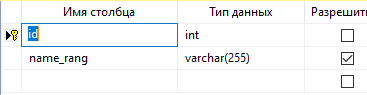


Таблица "Ранги" представляет информацию о конкретных рангах или статусах персонажей или команд в мире Наруто.

**Рисунок 10 – Структура таблицы «Хвостатые»**

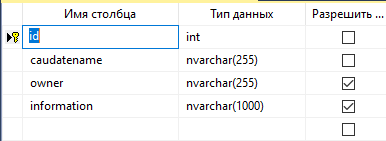


Таблица "Хвостатые" содержит данные о существах с хвостами в мире Наруто, их названиях, владельцах и других деталях.

**Рисунок 11 – Структура таблицы «Команды»**

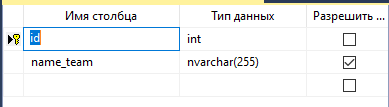


Таблица "Команды" содержит информацию о различных командах или группировках персонажей.

**Рисунок 12 – Структура таблицы «Деревни»**

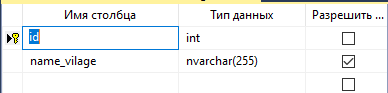


Таблица "Деревни" содержит данные о различных деревнях в мире Наруто.

**Рисунок 13 – Структура таблицы «Клан»**

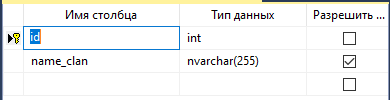


Таблица "Клан" содержит информацию о кланах или семьях персонажей.

**Рисунок 14 – Структура таблицы «Подтипы техник»**

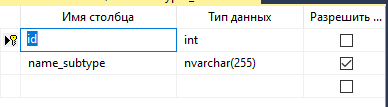


Таблица "Подтипы техник" хранит подкатегории или разновидности техник в мире Наруто.

**Рисунок 15 – Структура таблицы «Типы техник»**

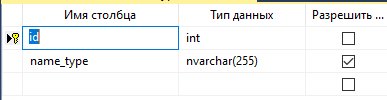


Таблица "Типы техник" содержит типы или категории техник, используемых в мире Наруто.

**Рисунок 16 – Структура таблицы «Техники»**

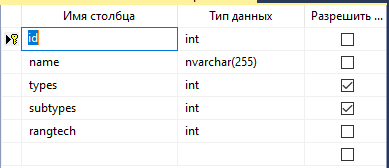


Таблица "Техники" представляет информацию о различных техниках, их названиях, типах, подтипах и связанных с ними рангах.

**Рисунок 16 – Структура таблицы «Ранги техник»**

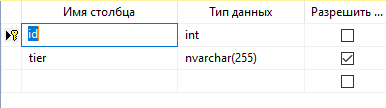


Таблица "Ранги техник" представляет информацию о сложности в изучении какой либо из техник.

**Рисунок 18 – Структура таблицы «Персонажи»**

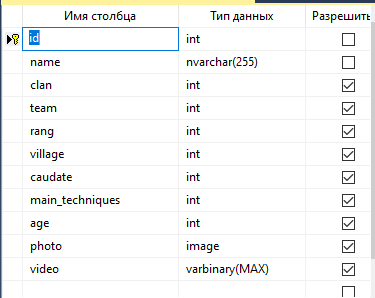


Таблица "Персонажи" содержит данные о персонажах, их именах, принадлежности к клану или команде, ранге, деревне, информации о хвостатых, основных техниках, возрасте и фотографиях.

3.4. Создание таблиц базы данных

Для создания определенных ранее таблиц использовался оператор SQL create table.

**Создание таблицы Роли**

CREATE TABLE Roles (

id INT PRIMARY KEY,

role\_name NVARCHAR(50) NOT NULL

);

**Создание таблицы Ранговая Технология**

CREATE TABLE RangTech (

id INT IDENTITY PRIMARY KEY,

tier NVARCHAR(255) NULL

);

**Создание таблицы Ранги**

CREATE TABLE Rangs (

id INT IDENTITY PRIMARY KEY,

name\_rang BIGINT NULL

);

**Создание таблицы Хвостатые**

CREATE TABLE Caudate (

id INT IDENTITY PRIMARY KEY,

caudatename NVARCHAR(255) NOT NULL,

owner NVARCHAR(255),

information NVARCHAR(1000)

);

**Создание таблицы Команды**

CREATE TABLE Teams (

id INT IDENTITY PRIMARY KEY,

name\_team NVARCHAR(255) NULL

);

**Создание таблицы Деревни**

CREATE TABLE Vilages (

id INT IDENTITY PRIMARY KEY,

name\_vilage NVARCHAR(255) NULL

);

**Создание таблицы Клан**

CREATE TABLE Clan (

id INT IDENTITY PRIMARY KEY,

name\_clan NVARCHAR(255) NULL

);

**Создание таблицы Подтипы Техник**

CREATE TABLE SubTypes\_Tech (

id INT IDENTITY PRIMARY KEY,

name\_subtype NVARCHAR(255) NULL

);

**Создание таблицы Типы Техник**

CREATE TABLE Types\_Tech (

id INT IDENTITY PRIMARY KEY,

name\_type NVARCHAR(255) NULL

);

**Создание таблицы Техники**

CREATE TABLE Techniques (

id INT IDENTITY PRIMARY KEY,

name NVARCHAR(255) NOT NULL,

types INT NULL DEFAULT 1,

subtypes INT NULL DEFAULT 1,

rangtech INT NOT NULL DEFAULT 1,

FOREIGN KEY (types) REFERENCES Types\_Tech(id),

FOREIGN KEY (subtypes) REFERENCES SubTypes\_Tech(id),

FOREIGN KEY (rangtech) REFERENCES RangTech(id)

);

**Создание таблицы Персонажи**

CREATE TABLE Characters (

id INT IDENTITY PRIMARY KEY,

name NVARCHAR(255) NOT NULL,

clan INT NULL DEFAULT 1,

team INT NULL DEFAULT 1,

rang INT NULL DEFAULT 1,

village INT NULL DEFAULT 1,

caudate INT NULL DEFAULT 1,

main\_techniques INT NULL DEFAULT 1,

age INT NULL,

photo IMAGE NULL,

FOREIGN KEY (main\_techniques) REFERENCES Techniques(id),

FOREIGN KEY (rang) REFERENCES Rangs(id),

FOREIGN KEY (team) REFERENCES Teams(id),

FOREIGN KEY (caudate) REFERENCES Caudate(id),

FOREIGN KEY (clan) REFERENCES Clan(id),

FOREIGN KEY (village) REFERENCES Vilages(id)

);

В результате выполнения всех этих запросов были созданы таблицы базы данных. Их список представлен на рисунке 19.

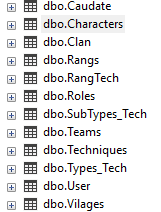


Рисунок 19 – Список созданных в базе данных таблиц

## 3.5. Ввод тестовых данных

После создания таблиц в них были внесены тестовые данные, позволяющие в дальнейшем оценить работу приложения. Внесенные данные представлены ниже (рисунки 20- 32 ).

Рисунок 20 – Данные в таблице «roles»

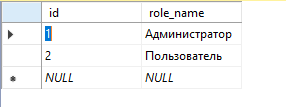


Рисунок 20

Рисунок 21 – Данные в таблице «User»

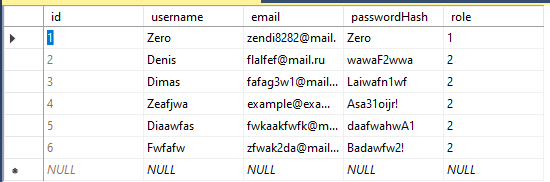


Рисунок 21

Рисунок 22 – Данные в таблице «rangtech»

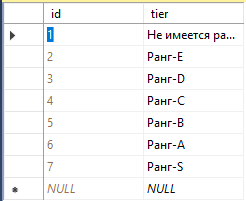


Рисунок 22

Рисунок 23 – Данные в таблице «rangs»

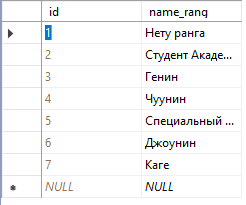


Рисунок 23

Рисунок 24 – Данные в таблице «caudate»

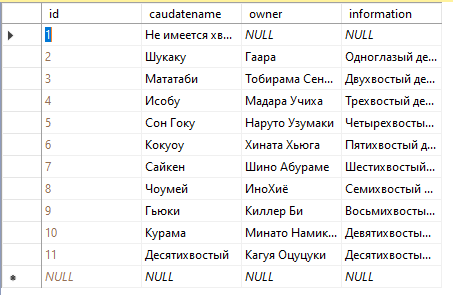


Рисунок 24

Рисунок 25 – Данные в таблице «teams»



Рисунок 25

Рисунок 26 – Данные в таблице «vilages»

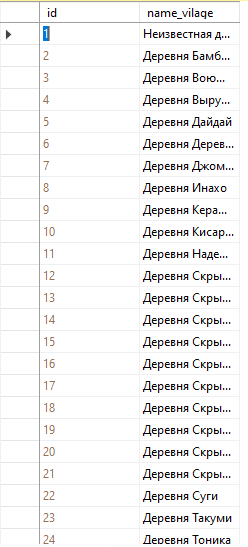


Рисунок 26

Рисунок 27 – Данные в таблице «clan»

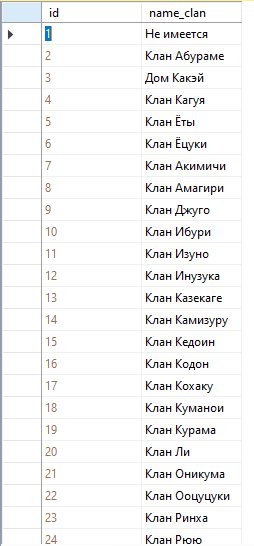


Рисунок 27

Рисунок 28 – Данные в таблице «subtypes\_tech»

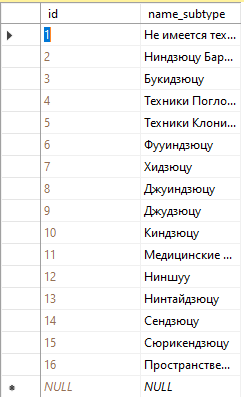


Рисунок 28

Рисунок 29 – Данные в таблице «types\_tech»

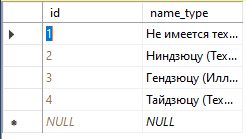


Рисунок 29

Рисунок 30 – Данные в таблице «techniques»

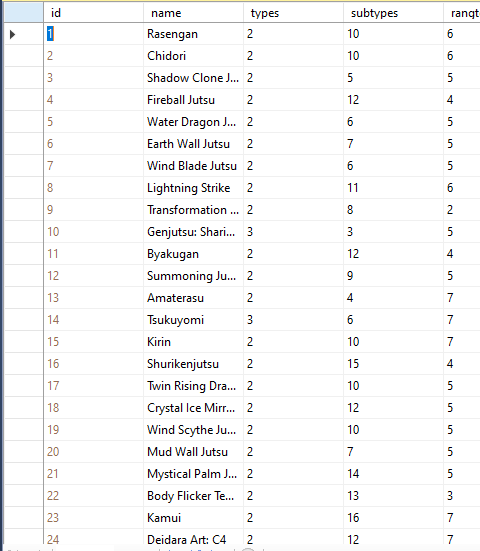


Рисунок 30

Рисунок 31 – Данные в таблице «characters»

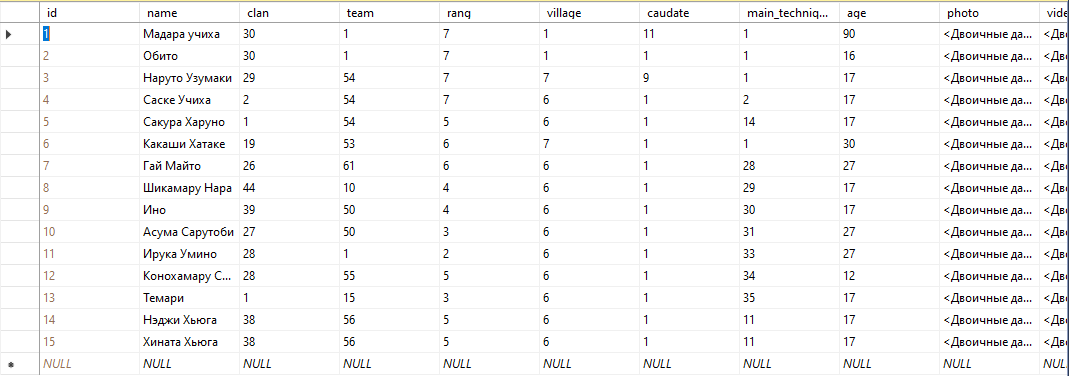


Рисунок 31

Просмотр (или представление) в базе данных представляет собой виртуальную таблицу, созданную на основе запроса SELECT, который выбирает определенные данные из одной или нескольких таблиц. Он не содержит реальных данных, а представляет собой "взгляд" на данные из таблицы или нескольких таблиц, отображая только необходимую информацию.

Преимущества использования просмотров включают:

1. **Ускорение выполнения запросов:** Просмотр создается на основе готового запроса, что может ускорить выполнение сложных запросов.
2. **Защита данных:** Просмотр может ограничивать доступ к определенным полям таблицы, скрывая конфиденциальную информацию.
3. **Форматирование данных для удобства пользователя:** Просмотр может отображать данные в удобном формате, соответствующем потребностям конкретных пользователей.

Для создания просмотра используется оператор SQL CREATE VIEW, который выглядит следующим образом:

CREATE VIEW имя\_просмотра[(столбец\_1,…, столбец\_N)]

AS

SELECT оператор\_SELECT;

Этот запрос создает "виртуальную таблицу" (просмотр), основанную на запросе SELECT.

Пример создания просмотра, который выводит данные из таблицы Techniques и RangTech:

CREATE VIEW TechniqueRanks AS

SELECT t.name AS TechniqueName, rt.tier AS RankName

FROM Techniques t

JOIN RangTech rt ON t.rangtech = rt.id;

Этот пример создает просмотр под названием TechniqueRanks , который соединяет данные о ролях и командах из таблиц Techniques и RangTech.

Результат вызова данного представления представлен на рисунке 32.

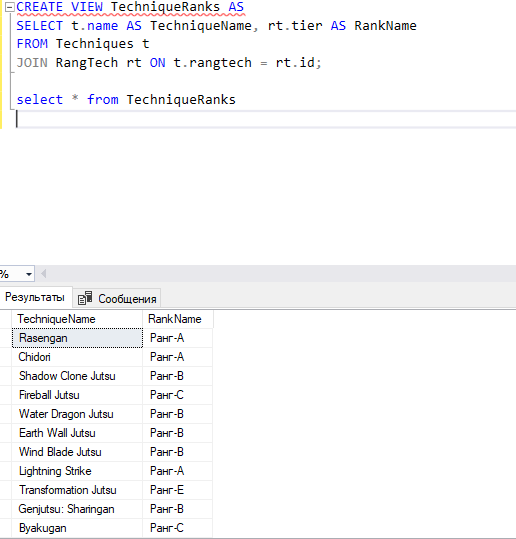


Рисунок 32 – Результат вызова представления TechniqueRanks

Префикс v (View) в имени представления позволяет программистам легче ориентироваться во множестве имен объектов. В ходе разработки было создано представление, которое содержит данные о персонажах и связанных с ними сущностях из других таблиц.

CREATE VIEW [dbo].[vCharacterDetails] AS

SELECT

c.id AS CharacterID,

c.name AS CharacterName,

c.age AS CharacterAge,

t.name\_team AS TeamName,

r.name\_rang AS RangName,

v.name\_vilage AS VillageName,

cd.caudatename AS CaudateName,

ct.name\_clan AS ClanName,

tech.name AS MainTechnique

FROM Characters c

LEFT JOIN Teams t ON c.team = t.id

LEFT JOIN Rangs r ON c.rang = r.id

LEFT JOIN Vilages v ON c.village = v.id

LEFT JOIN Caudate cd ON c.caudate = cd.id

LEFT JOIN Clan ct ON c.clan = ct.id

LEFT JOIN Techniques tech ON c.main\_techniques = tech.id;

Этот запрос создаст представление vCharacterDetails, содержащее данные о персонажах и связанных с ними данными из других таблиц по внешним ключам.

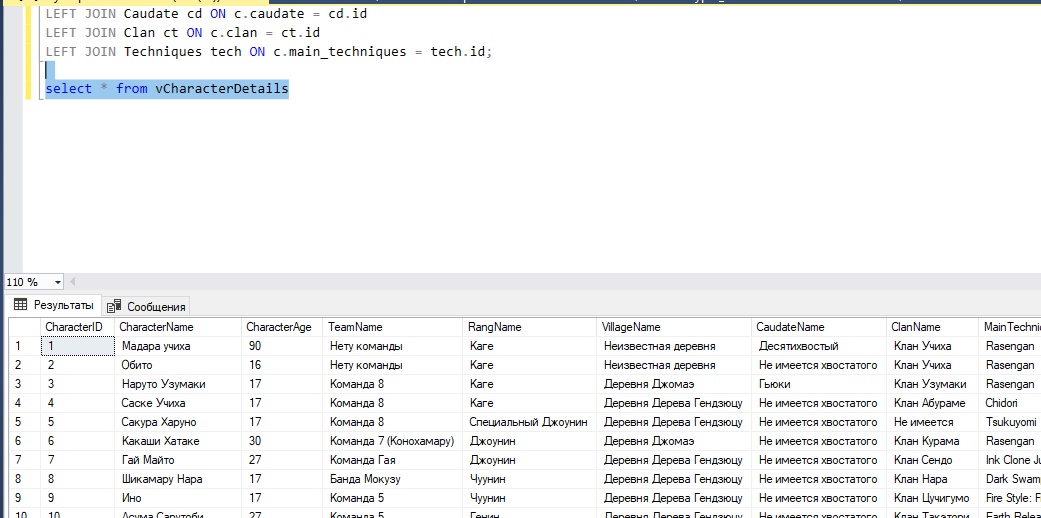


Рисунок 33 – Результат вызова представления vCharacterDetails

**3.7. Создание триггеров**

Триггеры представляют собой разновидность хранимых процедур, автоматически запускаемых при возникновении определенных событий (добавление, удаление, изменение) в таблицах базы данных. Они отличаются от хранимых процедур тем, что не вызываются пользователем, а активируются автоматически в ответ на определенные действия с данными.

Триггеры выполняются либо после события (AFTER), либо вместо него (INSTEAD OF), в зависимости от типа действия и используемой СУБД. Они используются для реализации ограничений, которые нельзя или сложно реализовать на уровне структуры таблицы. Эти ограничения могут быть связаны с поддержанием целостности данных или бизнес-правилами, касающимися данных из различных таблиц.

В ходе разработки базы данных возникла потребность в создании триггера, который ограничит изменения данных в таблице Characters, не допуская отрицательные значения возраста персонажа. Вот пример триггера, который реализует это ограничение:

CREATE TRIGGER trgPreventNegativeAge

ON Characters

AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT id

FROM inserted

WHERE age <= 0

)

BEGIN

RAISERROR('Возраст персонажа не может быть отрицательным или равным нулю.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

Таким образом созданный триггер обеспечивает предотвращение обновления данных в таблице Characters с недопустимыми значениями возраста персонажа.

## 3.8. Создание хранимых процедур

если вы хотите создать процедуру для обновления имени персонажа в таблице Characters, вот пример того, как это можно сделать:

CREATE PROCEDURE UpdateCharacterName

@CharacterID INT,

@NewName NVARCHAR(255)

AS

BEGIN

UPDATE Characters

SET name = @NewName

WHERE id = @CharacterID;

END;

Это пример простой хранимой процедуры, которая принимает идентификатор персонажа и новое имя в качестве параметров и обновляет имя персонажа в таблице Characters.

Раздел 4. Разработка клиентского приложения

4.1. Разработка модуля ИС NarutoViki При создании приложения NarutoViki были выделены различные модули, ответственные за функциональные возможности приложения, такие как модули форм, а также модуль, отвечающий за взаимодействие с базой данных. В последний были интегрированы функции, обеспечивающие доступ к базе данных, выполнение запросов. Это обеспечило абстракцию визуального интерфейса приложения от логики работы с базой данных, что позволило вносить изменения в базу данных независимо от пользовательского интерфейса. Такой подход к разработке обеспечивает гибкость системы и позволяет в дальнейшем расширять ее функциональность.

Ниже представлен фрагмент кода из модуля-класса, отвечающего за взаимодействие с базой данных в приложении NarutoViki:

Подключение к БД -

internal class SqlConnectionManager

{

// Строка подключения к вашей базе данных

public const string ConnectionString = @"Data Source=LAPTOP-GI7CLN64\SQLSTANDART;Initial Catalog=Kursnaruto;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

public SqlConnection GetSqlConnection()

{

SqlConnection connection = new SqlConnection(ConnectionString);

try

{

connection.Open();

Console.WriteLine("Соединение успешно установлено");

return connection;

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при установлении соединения: " + ex.Message);

return null;

}

}

// Метод для закрытия подключения

public void CloseSqlConnection(SqlConnection connection)

{

if (connection != null && connection.State == System.Data.ConnectionState.Open)

{

connection.Close();

Console.WriteLine("Соединение успешно закрыто");

}

}

}

Метод авторизации -

public partial class Login : Page

{

private SqlConnectionManager sqlConnectionManager = new SqlConnectionManager();

private MainWindow mainWindow;

private TechniquesPage techniquesPageInstance;

public int roleId;

private Login login; // Объявление переменной login

public Login(MainWindow mainWindow, TechniquesPage techniquesPage)

{

InitializeComponent();

this.mainWindow = mainWindow;

this.techniquesPageInstance = techniquesPage;

}

private void ManageMenuBasedOnRole(int roleId)

{

if (roleId == 1)

{

// Показать меню

techniquesPageInstance.ShowAdminMenu();

}

else if (roleId == 2)

{

// Скрыть меню

techniquesPageInstance.HideAdminMenu();

}

}

private void Login\_Click(object sender, RoutedEventArgs e)

{

string username = txtUsername.Text;

string password = txtPassword.Password;

if (AuthenticateUser(username, password))

{

MessageBox.Show("Успешный вход!");

mainWindow.SetIsUserRegistered(true);

MainGrid.Visibility = Visibility.Collapsed;

}

else

{

MessageBox.Show("Ошибка входа. Проверьте логин и пароль.");

}

}

private void Register\_Click(object sender, RoutedEventArgs e)

{

if (NavigationService != null)

{

NavigationService.Navigate(new Registr());

}

}

private bool AuthenticateUser(string username, string password)

{

SqlConnection connection = null;

try

{

connection = sqlConnectionManager.GetSqlConnection();

if (connection != null)

{

string query = "SELECT id, role FROM [dbo].[User] WHERE username = @Username AND passwordHash = @Password";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

roleId = reader.GetInt32(1); // Сохраняем значение roleId

mainWindow.SetIsUserRegistered(true);

SetUserRole(roleId); // Используем roleId для установки роли пользователя

return true;

}

}

}

return false;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка входа: " + ex.Message);

return false;

}

finally

{

sqlConnectionManager.CloseSqlConnection(connection);

}

}

public void SetUserRole(int roleId)

{

if (techniquesPageInstance != null && login != null)

{

if (roleId == 1)

{

// Показать меню

techniquesPageInstance.ShowAdminMenu();

}

else if (roleId == 2)

{

// Скрыть меню

techniquesPageInstance.HideAdminMenu();

}

}

}

}

}

Метод регистрации:

public partial class Registr : Page

{

private Login loginPage;

private SqlConnectionManager sqlConnectionManager; // Объявление объекта SqlConnectionManager

private Frame mainFrame;

public Registr()

{

InitializeComponent();

sqlConnectionManager = new SqlConnectionManager(); // Инициализация объекта SqlConnectionManager

}

private bool IsEmailValid(string email)

{

// Регулярное выражение для проверки формата email

string emailPattern = @"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$";

return Regex.IsMatch(email, emailPattern);

}

private void btnAdd\_Click(object sender, RoutedEventArgs e)

{

string username = tbtName.Text;

string email = tbtDr.Text;

string password = tbtTel.Text;

if (password.Length < 6 || password.Length > 12)

{

MessageBox.Show("Пароль должен содержать от 6 до 12 символов.");

return;

}

// Проверка формата email

if (!IsEmailValid(email))

{

MessageBox.Show("Неправильный формат email.");

return;

}

// Проверка пароля на соответствие требованиям

if (!CheckPasswordRequirements(password))

{

MessageBox.Show("Пароль должен содержать от 6 до 12 символов, включая минимум одну заглавную букву и одну цифру.");

return;

}

TechniquesPage techniquesPage = new TechniquesPage(mainFrame, loginPage);

// Вызов метода регистрации пользователя

bool registrationResult = AddUserToDatabase(username, email, password);

if (registrationResult)

{

MessageBox.Show("Регистрация успешна!");

// Создание объекта Login с передачей MainWindow и TechniquesPage в качестве аргументов

Login loginPage = new Login((MainWindow)Application.Current.MainWindow, techniquesPage);

NavigationService?.Navigate(loginPage);

}

else

{

MessageBox.Show("Ошибка при регистрации. Пожалуйста, проверьте данные.");

}

}

private bool CheckPasswordRequirements(string password)

{

// Проверка пароля на длину, наличие заглавной буквы и цифры

if (password.Length < 6 || password.Length > 12)

{

return false;

}

// Используем регулярные выражения для проверки наличия заглавной буквы и цифры

if (!Regex.IsMatch(password, @"^(?=.\*[A-Z])(?=.\*\d).+$"))

{

return false;

}

return true;

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

NavigationService?.GoBack();

}

private bool AddUserToDatabase(string username, string email, string password)

{

try

{

using (SqlConnection connection = sqlConnectionManager.GetSqlConnection())

{

if (connection != null)

{

string query = "INSERT INTO [User] (username, email, passwordHash, role) VALUES (@Username, @Email, @Password, @Role)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Email", email);

command.Parameters.AddWithValue("@Password", password);

command.Parameters.AddWithValue("@Role", 2); // Устанавливаем роль пользователя

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

return false;

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при регистрации пользователя: " + ex.Message);

return false;

}

}

}

}

Метод вывода видео:  
public partial class VideoPlayerWindow : Window

{

private readonly byte[] videoBytes;

public VideoPlayerWindow(byte[] video)

{

InitializeComponent();

videoBytes = video;

Loaded += VideoPlayerWindow\_Loaded;

}

private void PlayVideoWithKMPlayer(string videoPath)

{

try

{

Process.Start(@"C:\KMPlayer\KMPlayer.exe", videoPath);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при запуске KMPlayer: {ex.Message}");

}

}

private void VideoPlayerWindow\_Loaded(object sender, RoutedEventArgs e)

{

try

{

if (videoBytes != null)

{

string tempVideoPath = System.IO.Path.Combine(System.IO.Path.GetTempPath(), "tempVideo.mp4");

System.IO.File.WriteAllBytes(tempVideoPath, videoBytes);

PlayVideoWithKMPlayer(tempVideoPath);

}

else

{

MessageBox.Show("Видео для этого персонажа не найдено.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при воспроизведении видео: {ex.Message}");

}

}

private void MediaElement\_MediaOpened(object sender, RoutedEventArgs e)

{

if (mediaElement.NaturalDuration.HasTimeSpan)

{

mediaElement.Play();

}

}

private void MediaElement\_MediaFailed(object sender, ExceptionRoutedEventArgs e)

{

MessageBox.Show($"Ошибка при воспроизведении видео: {e.ErrorException.Message}");

}

}

}

Страница вывода техник:  
public partial class TechniquesPage : Page

{

private Login login;

string connectionString = SqlConnectionManager.ConnectionString;

SqlConnectionManager sqlConnectionManager = new SqlConnectionManager();

private Frame MainFrame { get; set; }

public ObservableCollection<TechniquesList> TechniquesList { get; set; }

public TechniquesPage(Frame mainFrame, Login login)

{

InitializeComponent();

MainFrame = mainFrame;

TechniquesList = new ObservableCollection<TechniquesList>();

this.login = login;

if (this.login != null)

{

this.login.SetUserRole(this.login.roleId);

}

LoadData();

}

public void ShowAdminMenu()

{

if (AdminMenu != null)

{

AdminMenu.Visibility = Visibility.Visible;

}

}

// Метод для скрытия меню для других ролей, кроме администратора

public void HideAdminMenu()

{

if (AdminMenu != null)

{

AdminMenu.Visibility = Visibility.Collapsed;

}

}

private void ComboBoxFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ApplyFilters();

}

private void TextBoxFilter\_TextChanged(object sender, TextChangedEventArgs e)

{

ApplyFilters();

}

// Метод для проверки роли пользователя и отображения меню

private void ApplyFilters()

{

var filteredList = TechniquesList.ToList();

// Фильтрация по типу техники

string selectedType = (comboBoxTypeFilter.SelectedItem as ComboBoxItem)?.Content.ToString();

if (!string.IsNullOrEmpty(selectedType) && selectedType != "Не имеется техники")

{

filteredList = filteredList.Where(item => item.TypeName == selectedType).ToList();

}

// Фильтрация по подтипу техники

string selectedSubtype = (comboBoxSubtypeFilter.SelectedItem as ComboBoxItem)?.Content.ToString();

if (!string.IsNullOrEmpty(selectedSubtype) && selectedSubtype != "Не имеется техники")

{

filteredList = filteredList.Where(item => item.SubtypeName == selectedSubtype).ToList();

}

// Фильтрация по рангу техники

string selectedRang = (comboBoxRangFilter.SelectedItem as ComboBoxItem)?.Content.ToString();

if (!string.IsNullOrEmpty(selectedRang) && selectedRang != "Не имеется ранга")

{

filteredList = filteredList.Where(item => item.RangTechTier == selectedRang).ToList();

}

// Поиск по текстовому полю

if (!string.IsNullOrEmpty(textBoxSearch.Text))

{

string searchQuery = textBoxSearch.Text.ToLower();

filteredList = filteredList.Where(item =>

item.TechniqueName.ToLower().Contains(searchQuery) ||

item.TypeName.ToLower().Contains(searchQuery) ||

item.SubtypeName.ToLower().Contains(searchQuery) ||

item.RangTechTier.ToLower().Contains(searchQuery)).ToList();

}

techniquesDataGrid.ItemsSource = filteredList;

}

public void Add\_Click(object sender, RoutedEventArgs e)

{

if (MainFrame != null)

{

MainFrame.Navigate(new AddTechniquePage(MainFrame, this));

}

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

if (techniquesDataGrid.SelectedItem != null)

{

// Получение выбранного элемента

TechniquesList selectedTechnique = (TechniquesList)techniquesDataGrid.SelectedItem;

// Удаление элемента из коллекции и базы данных

if (DeleteTechnique(selectedTechnique.TechniqueID))

{

MessageBox.Show("Техника успешно удалена!");

// Перезагрузка данных после удаления

LoadData();

}

else

{

MessageBox.Show("Ошибка при удалении техники.");

}

}

else

{

MessageBox.Show("Выберите технику для удаления.");

}

}

private bool DeleteTechnique(int techniqueID)

{

try

{

using (SqlConnection connection = sqlConnectionManager.GetSqlConnection())

{

if (connection != null)

{

string query = @"DELETE FROM Techniques WHERE id = @TechniqueID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@TechniqueID", techniqueID);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

return false;

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении техники: " + ex.Message);

return false;

}

}

private void Update\_Click(object sender, RoutedEventArgs e)

{

// Получение выбранного объекта TechniquesList из DataGrid

TechniquesList selectedTechnique = techniquesDataGrid.SelectedItem as TechniquesList;

if (selectedTechnique != null)

{

// Выполните необходимые изменения в объекте TechniquesList (например, изменение его свойств)

// Обновление источника данных DataGrid

techniquesDataGrid.ItemsSource = null;

techniquesDataGrid.ItemsSource = TechniquesList;

}

}

public void SetUserRole(int roleId)

{

if (roleId == 1)

{

// Показать меню

ShowAdminMenu();

}

else if (roleId == 2)

{

// Скрыть меню

HideAdminMenu();

}

}

public void LoadData()

{

string connectionString = SqlConnectionManager.ConnectionString;

string query = @"SELECT

Techniques.id AS TechniqueID,

Techniques.name AS TechniqueName,

RangTech.tier AS RangTechTier,

SubTypes\_Tech.name\_subtype AS SubtypeName,

Types\_Tech.name\_type AS TypeName

FROM

[dbo].[Techniques] AS Techniques

INNER JOIN

[dbo].[RangTech] AS RangTech ON Techniques.rangtech = RangTech.id

INNER JOIN

[dbo].[SubTypes\_Tech] AS SubTypes\_Tech ON Techniques.subtypes = SubTypes\_Tech.id

INNER JOIN

[dbo].[Types\_Tech] AS Types\_Tech ON Techniques.types = Types\_Tech.id; ";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

TechniquesList = new ObservableCollection<TechniquesList>();

foreach (DataRow row in dataTable.Rows)

{

TechniquesList technique = new TechniquesList(

Convert.ToInt32(row["TechniqueID"]),

row["TechniqueName"].ToString(),

row["RangTechTier"].ToString(),

row["SubtypeName"].ToString(),

row["TypeName"].ToString()

);

TechniquesList.Add(technique);

}

techniquesDataGrid.ItemsSource = TechniquesList;

}

}

}

}

Главное окно:  
 public partial class MainWindow : Window

{

private TechniquesPage techniquesPageInstance;

private Login loginPage;

private TechniquesPage techniquesPage;

private bool IsUserRegistered = false;

public MainWindow()

{

InitializeComponent();

techniquesPage = new TechniquesPage(MainFrame, loginPage);

loginPage = new Login(this, techniquesPage);

WindowState = WindowState.Maximized;

HideButtons(); // Скрыть кнопки при инициализации MainWindow

if (MainFrame != null)

{

MainFrame.Navigate(loginPage);

}

}

public void SetTechniquesPageInstance(TechniquesPage techniquesPage)

{

techniquesPageInstance = techniquesPage;

}

public void SetIsUserRegistered(bool value)

{

IsUserRegistered = value;

CheckUserRegistration(); // Вызов метода для проверки статуса регистрации пользователя

}

public bool GetIsUserRegistered()

{

return IsUserRegistered;

}

private void CheckUserRegistration()

{

HideButtons(); // Скрываем кнопки по умолчанию

if (IsUserRegistered)

{

ShowButtons();

}

}

public void ShowButtons()

{

// Показать кнопки, установив их видимость в true

CharactersButton.Visibility = Visibility.Visible;

TechniquesButton.Visibility = Visibility.Visible;

TailedBeastsButton.Visibility = Visibility.Visible;

}

public void HideButtons()

{

// Скрыть кнопки, установив их видимость в false

CharactersButton.Visibility = Visibility.Collapsed;

TechniquesButton.Visibility = Visibility.Collapsed;

TailedBeastsButton.Visibility = Visibility.Collapsed;

}

private void MainFrame\_Navigated(object sender, NavigationEventArgs e)

{

}

private void CharactersButton\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new Characters());

}

private void TechniquesButton\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new TechniquesPage(MainFrame, loginPage));

}

private void TailedBeastsButton\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new CaudatePage());

}

}

}

Страница вывода персонажей:  
 public partial class Characters : Page

{

private ObservableCollection<clases.Character.CharacterList> charactersList;

private readonly SqlConnectionManager connectionManager = new SqlConnectionManager();

private SqlConnection connection; // Объявление переменной connection

public Characters()

{

InitializeComponent();

charactersList = new ObservableCollection<clases.Character.CharacterList>();

LoadCharactersFromDatabase();

charactersListBox.ItemsSource = charactersList;

}

private void GetInfo\_Click(object sender, RoutedEventArgs e)

{

if (charactersListBox.SelectedItem != null)

{

clases.Character.CharacterList selectedCharacter = (clases.Character.CharacterList)charactersListBox.SelectedItem;

string connectionString = SqlConnectionManager.ConnectionString;

string query = @"

SELECT Characters.name AS CharacterName,

Rangs.name\_rang AS Rang,

Clan.name\_clan AS Clan,

Caudate.caudatename AS Caudate,

Teams.name\_team AS Teams,

RangTech.tier AS RangTech,

Techniques.name AS Techniques,

Types\_Tech.name\_type AS TypeTech,

SubTypes\_Tech.name\_subtype AS SubTypesTech,

Vilages.name\_vilage AS Vilages,

Characters.age AS Age

FROM Characters

LEFT JOIN Rangs ON Characters.rang = Rangs.id

LEFT JOIN Clan ON Characters.clan = Clan.id

LEFT JOIN Caudate ON Characters.caudate = Caudate.id

LEFT JOIN Teams ON Characters.team = Teams.id

LEFT JOIN RangTech ON Rangs.id = RangTech.id

LEFT JOIN Techniques ON Characters.main\_techniques = Techniques.id

LEFT JOIN Types\_Tech ON Techniques.types = Types\_Tech.id

LEFT JOIN SubTypes\_Tech ON Techniques.subtypes = SubTypes\_Tech.id

LEFT JOIN Vilages ON Characters.village = Vilages.id

WHERE Characters.id = @CharacterId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@CharacterId", selectedCharacter.Id); // Передача ID выбранного персонажа

connection.Open();

SqlDataReader reader = command.ExecuteReader();

// Переменные для хранения данных

string characterName = null, rang = null, clan = null, caudate = null, teams = null,

rangTech = null, techniques = null, typeTech = null, subTypesTech = null, vilages = null;

int age = 0;

// Если есть результаты запроса

if (reader.Read())

{

characterName = reader["CharacterName"].ToString();

rang = reader["Rang"].ToString();

clan = reader["Clan"].ToString();

caudate = reader["Caudate"].ToString();

teams = reader["Teams"].ToString();

rangTech = reader["RangTech"].ToString();

techniques = reader["Techniques"].ToString();

typeTech = reader["TypeTech"].ToString();

subTypesTech = reader["SubTypesTech"].ToString();

vilages = reader["Vilages"].ToString();

age = Convert.ToInt32(reader["Age"]);

}

reader.Close();

// Создаем объект для хранения информации о персонаже

clases.CharacterInfo characterInfo = new clases.CharacterInfo(

characterName, rang, clan, caudate, teams, rangTech, techniques, typeTech, subTypesTech, vilages, age.ToString()

);

// Передаем выбранного персонажа в окно CharacterInfoWindow

CharacterInfoWindow characterInfoWindow = new CharacterInfoWindow(characterInfo);

characterInfoWindow.ShowDialog();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса: " + ex.Message);

}

}

}

private void GetVideo\_Click(object sender, RoutedEventArgs e)

{

if (charactersListBox.SelectedItem != null)

{

clases.Character.CharacterList selectedCharacter = (clases.Character.CharacterList)charactersListBox.SelectedItem;

GetVideo(selectedCharacter);

}

}

private void GetVideo(clases.Character.CharacterList selectedCharacter)

{

if (selectedCharacter.Video != null)

{

// Создание окна проигрывателя видео и передача пути к видео

VideoPlayerWindow videoPlayerWindow = new VideoPlayerWindow(selectedCharacter.Video);

videoPlayerWindow.Show();

}

else

{

MessageBox.Show("Видео для этого персонажа не найдено.");

}

}

private void CharacterSelected(object sender, SelectionChangedEventArgs e)

{

if (charactersListBox.SelectedItem != null)

{

clases.Character.CharacterList selectedCharacter = (clases.Character.CharacterList)charactersListBox.SelectedItem;

GetInfo(selectedCharacter.Id);

}

}

private void GetInfo(int characterId)

{

if (charactersListBox.SelectedItem != null)

{

string connectionString = SqlConnectionManager.ConnectionString;

string query = @"

SELECT Characters.name AS CharacterName,

Rangs.name\_rang AS Rang,

Clan.name\_clan AS Clan,

Caudate.caudatename AS Caudate,

Teams.name\_team AS Teams,

RangTech.tier AS RangTech,

Techniques.name AS Techniques,

Types\_Tech.name\_type AS TypeTech,

SubTypes\_Tech.name\_subtype AS SubTypesTech,

Vilages.name\_vilage AS Vilages,

Characters.age AS Age

FROM Characters

LEFT JOIN Rangs ON Characters.rang = Rangs.id

LEFT JOIN Clan ON Characters.clan = Clan.id

LEFT JOIN Caudate ON Characters.caudate = Caudate.id

LEFT JOIN Teams ON Characters.team = Teams.id

LEFT JOIN RangTech ON Rangs.id = RangTech.id

LEFT JOIN Techniques ON Characters.main\_techniques = Techniques.id

LEFT JOIN Types\_Tech ON Techniques.types = Types\_Tech.id

LEFT JOIN SubTypes\_Tech ON Techniques.subtypes = SubTypes\_Tech.id

LEFT JOIN Vilages ON Characters.village = Vilages.id

WHERE Characters.id = @CharacterId"; // Фильтрация по ID персонажа

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@CharacterId", characterId); // Передача ID выбранного персонажа

connection.Open();

SqlDataReader reader = command.ExecuteReader();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса: " + ex.Message);

}

}

}

private void LoadCharactersFromDatabase()

{

try

{

using (connection = connectionManager.GetSqlConnection())

{

if (connection != null)

{

string query = "SELECT Id, Name, Clan, Team, Rang, Village, Caudate, Main\_techniques, Age, Photo, Video FROM Characters";

SqlCommand command = new SqlCommand(query, connection);

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

int id = reader.GetInt32(0);

string name = reader.GetString(1);

int? clanId = reader.IsDBNull(2) ? null : (int?)reader.GetInt32(2);

int? teamId = reader.IsDBNull(3) ? null : (int?)reader.GetInt32(3);

int? rangId = reader.IsDBNull(4) ? null : (int?)reader.GetInt32(4);

int? villageId = reader.IsDBNull(5) ? null : (int?)reader.GetInt32(5);

int? caudateId = reader.IsDBNull(6) ? null : (int?)reader.GetInt32(6);

int? mainTechniqueId = reader.IsDBNull(7) ? null : (int?)reader.GetInt32(7);

int? age = reader.IsDBNull(8) ? null : (int?)reader.GetInt32(8);

byte[] photo = reader.IsDBNull(9) ? null : (byte[])reader["Photo"];

byte[] video = reader.IsDBNull(10) ? null : (byte[])reader["Video"];

var character = new clases.Character.CharacterList(

id, name, clanId, teamId, rangId, villageId, caudateId, mainTechniqueId, age, photo, video

);

charactersList.Add(character);

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке персонажей из базы данных: {ex.Message}");

}

}

private void OnExit(object sender, EventArgs e)

{

connectionManager.CloseSqlConnection(connection);

}

}

}

Окно вывода информации о персонаже:  
public partial class CharacterInfoWindow : Window

{

public clases.CharacterInfo CharacterInfo { get; set; }

public CharacterInfoWindow(clases.CharacterInfo characterInfo)

{

InitializeComponent();

CharacterInfo = characterInfo;

DataContext = CharacterInfo;

}

public clases.Character SelectedCharacter { get; set; }

}

}

Страница вывода информации о хвостатых:  
public partial class CaudatePage : Page

{

private List<Hvost> caudateList = new List<Hvost>();

public CaudatePage()

{

InitializeComponent();

LoadCaudateData();

}

private void LoadCaudateData()

{

string connectionString = SqlConnectionManager.ConnectionString;

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT id, caudatename, owner, information FROM Caudate";

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

Hvost caudate = new Hvost(

reader.GetInt32(0), // ID

reader.GetString(1), // CaudateName

reader.IsDBNull(2) ? string.Empty : reader.GetString(2), // Description из БД

reader.IsDBNull(3) ? string.Empty : reader.GetString(3) // Abilities из БД

);

caudateList.Add(caudate);

}

}

reader.Close();

}

Caudate.ItemsSource = caudateList;

}

private void Caudate\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Caudate.SelectedItem != null)

{

Hvost selectedHvost = (Hvost)Caudate.SelectedItem;

// При выборе строки, выводите информацию о хвостатом персонаже в MessageBox

string message = $"ID: {selectedHvost.ID}\nИмя: {selectedHvost.CaudateName}\n" +

$"Описание: {selectedHvost.Description}\nСпособности: {selectedHvost.Abilities}";

MessageBox.Show(message, "Информация о хвостатом персонаже");

}

}

}

}

Страница добавления техник:  
public partial class AddTechniquePage : Page

{

private TechniquesPage techniquesPage;

private Frame MainFrame { get; set; }

public ObservableCollection<TechniquesList> TechniquesList { get; set; }

private SqlConnectionManager sqlConnectionManager;

public AddTechniquePage(Frame mainFrame, TechniquesPage techniquesPage)

{

InitializeComponent();

sqlConnectionManager = new SqlConnectionManager();

this.techniquesPage = techniquesPage;

}

private void btnAdd\_Click(object sender, RoutedEventArgs e)

{

try

{

using (SqlConnection connection = sqlConnectionManager.GetSqlConnection())

{

string techniqueName = TechniqueNameTextBox.Text;

string rangTechTier = (RangTechTierComboBox.SelectedItem as ComboBoxItem)?.Content.ToString();

string subTypeName = (SubtypeNameComboBox.SelectedItem as ComboBoxItem)?.Content.ToString();

string typeName = (TypeNameComboBox.SelectedItem as ComboBoxItem)?.Content.ToString();

// Добавление данных в базу данных

if (AddTechniqueToDatabase(techniqueName, rangTechTier, subTypeName, typeName))

{

MessageBox.Show("Техника успешно добавлена в базу данных!");

}

else

{

MessageBox.Show("Ошибка при добавлении техники в базу данных.");

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

techniquesPage.LoadData();

NavigationService?.GoBack();

}

private bool AddTechniqueToDatabase(string techniqueName, string rangTechTier, string subTypeName, string typeName)

{

try

{

using (SqlConnection connection = sqlConnectionManager.GetSqlConnection())

{

if (connection != null)

{

string query = @"

INSERT INTO Techniques ([name], [types], [subtypes], [rangtech])

SELECT

@Name, Types\_Tech.id, SubTypes\_Tech.id, RangTech.id

FROM

[dbo].[Types\_Tech] AS Types\_Tech

INNER JOIN

[dbo].[SubTypes\_Tech] AS SubTypes\_Tech ON SubTypes\_Tech.name\_subtype = @SubType

INNER JOIN

[dbo].[RangTech] AS RangTech ON RangTech.tier = @RangTech

WHERE Types\_Tech.name\_type = @Type; -- Фильтрация по выбранному типу техники

";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Name", techniqueName);

command.Parameters.AddWithValue("@RangTech", rangTechTier);

command.Parameters.AddWithValue("@SubType", subTypeName);

command.Parameters.AddWithValue("@Type", typeName);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

return false;

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении техники: " + ex.Message);

return false;

}

}

}

}

Лист персонажей:

using System.Windows.Media.Imaging; // Для BitmapImage

namespace NarutoKurs.clases

{

public class Character

{

public class CharacterList

{

public int Id { get; set; }

public string Name { get; set; }

public int? ClanId { get; set; }

public int? TeamId { get; set; }

public int? RangId { get; set; }

public int? VillageId { get; set; }

public int? CaudateId { get; set; }

public int? MainTechniqueId { get; set; }

public int? Age { get; set; }

public BitmapImage Photo { get; set; }

public byte[] Video { get; set; }

public CharacterList(int id, string name, int? clanId, int? teamId, int? rangId, int? villageId, int? caudateId, int? mainTechniqueId, int? age, byte[] photo, byte[] video)

{

Id = id;

Name = name;

ClanId = clanId;

TeamId = teamId;

RangId = rangId;

VillageId = villageId;

CaudateId = caudateId;

MainTechniqueId = mainTechniqueId;

Age = age;

Photo = ByteArrayToBitmapImage(photo);

Video = video;

}

// Метод конвертации массива байтов в BitmapImage

private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)

{

if (byteArray == null || byteArray.Length == 0) return null;

BitmapImage bitmapImage = new BitmapImage();

using (System.IO.MemoryStream memoryStream = new System.IO.MemoryStream(byteArray))

{

memoryStream.Position = 0;

bitmapImage.BeginInit();

bitmapImage.CacheOption = BitmapCacheOption.OnLoad;

bitmapImage.StreamSource = memoryStream;

bitmapImage.EndInit();

}

return bitmapImage;

}

}

}

}

Лист информации о персонажах:

namespace NarutoKurs.clases

{

public class CharacterInfo

{

public string CharacterName { get; set; }

public string Rang { get; set; }

public string Clan { get; set; }

public string Caudate { get; set; }

public string Teams { get; set; }

public string RangTech { get; set; }

public string Techniques { get; set; }

public string TypeTech { get; set; }

public string SubTypesTech { get; set; }

public string Vilages { get; set; }

public string Age { get; set; }

public CharacterInfo(string characterName, string rang, string clan, string caudate, string teams, string rangtech, string techniques, string typetech, string subTypesTech, string vilages, string age)

{

CharacterName = characterName;

Rang = rang;

Clan = clan;

Caudate = caudate;

Teams = teams;

RangTech = rangtech;

Techniques = techniques;

TypeTech = typetech;

SubTypesTech = subTypesTech;

Vilages = vilages;

Age = age;

}

}

}

Метод для изменения внешнего вида таблицы в окне Techniques  
namespace NarutoKurs.clases

{

public class RankToColorConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value != null && value is string rank)

{

switch (rank)

{

case "Ранг-E":

return new SolidColorBrush(Colors.Gray);

case "Ранг-D":

return new SolidColorBrush(Colors.Brown);

case "Ранг-C":

return new SolidColorBrush(Colors.Blue);

case "Ранг-B":

return new SolidColorBrush(Colors.Red);

case "Ранг-A":

return new SolidColorBrush(Colors.Gold);

case "Ранг-S":

// Для Ранг-S используем градиент

LinearGradientBrush gradientBrush = new LinearGradientBrush();

gradientBrush.GradientStops.Add(new GradientStop(Colors.Red, 0));

gradientBrush.GradientStops.Add(new GradientStop(Colors.Orange, 0.2));

gradientBrush.GradientStops.Add(new GradientStop(Colors.Yellow, 0.4));

gradientBrush.GradientStops.Add(new GradientStop(Colors.Green, 0.6));

gradientBrush.GradientStops.Add(new GradientStop(Colors.Blue, 0.8));

gradientBrush.GradientStops.Add(new GradientStop(Colors.Violet, 1));

return gradientBrush;

default:

return new SolidColorBrush(Colors.Transparent);

}

}

return new SolidColorBrush(Colors.Transparent);

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

Лист Техник:  
namespace NarutoKurs.clases

{

public class TechniquesList

{

public int TechniqueID { get; set; }

public string TechniqueName { get; set; }

public string RangTechTier { get; set; }

public string SubtypeName { get; set; }

public string TypeName { get; set; }

// Конструктор для инициализации объектов класса TechniquesList

public TechniquesList(int techniqueID, string techniqueName, string rangTechTier, string subTypeName, string typeName)

{

TechniqueID = techniqueID;

TechniqueName = techniqueName;

RangTechTier = rangTechTier;

SubtypeName = subTypeName;

TypeName = typeName;

}

}

}

Во всем проекте сохранен единый стиль наименований, отражающий функциональное назначение методов, переменных, компонентов.

Проект включает в себя комментарии как на уровне классов, так и методов. Внутри методов комментарии поясняют назначение существенных для понимания фрагментов кода.

## 4.2. Демонстрация работы приложения

После запуска приложения пользователю отображается окно авторизации, представленное на рисунке 34.

Так скажем меню нашего приложения

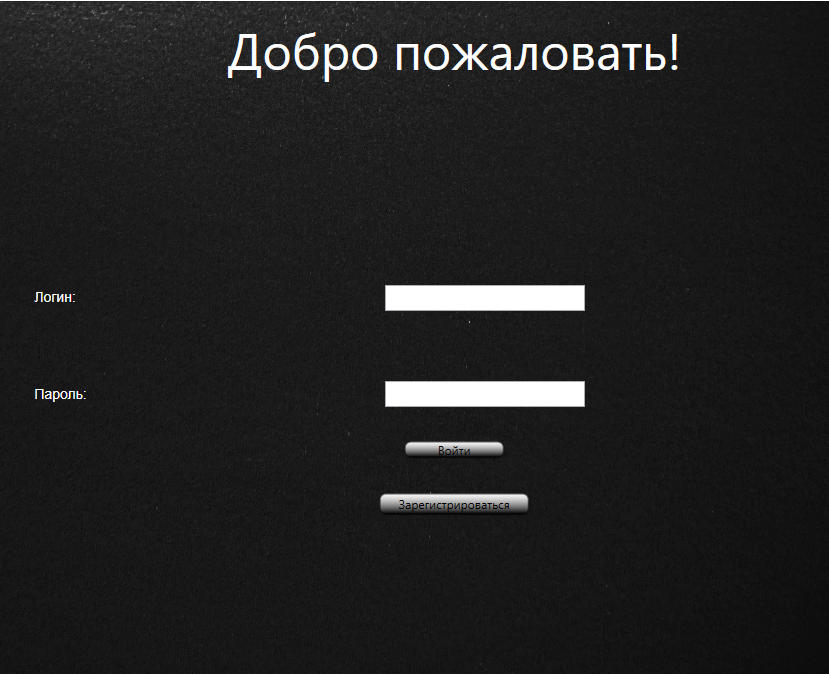


Рисунок 34.

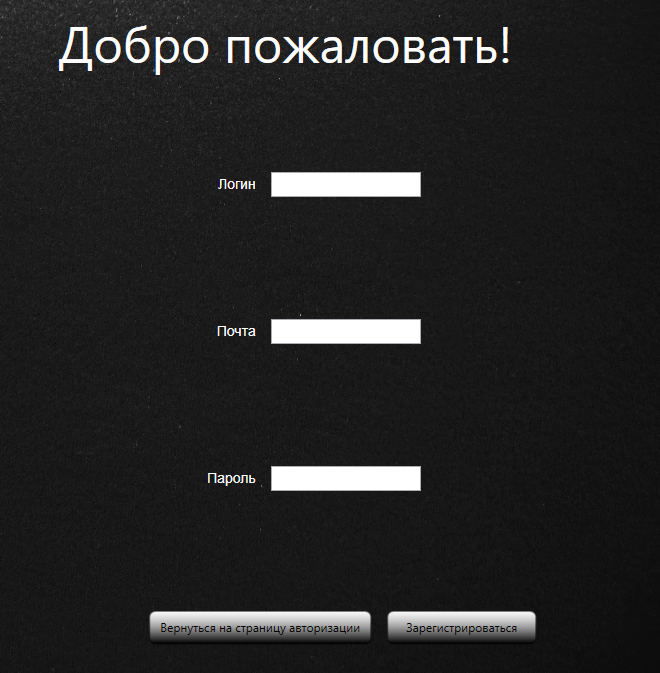
Страница регистрации:  


Рисунок 35.

После успешной авторизации, в левой части окна вам откроются кнопки навигации, с помощью которых вы сможете перейти на одну из страниц.

(Техники, Персонажи, Хвостатые)

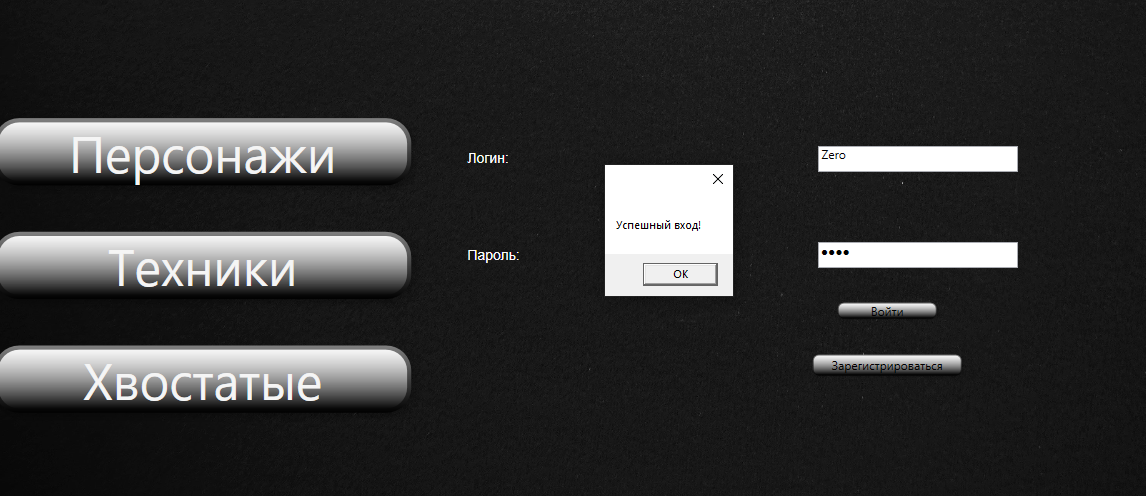


Рисунок 36.

На рисунке 36. Вы можете увидеть окно об успешной авторизации.

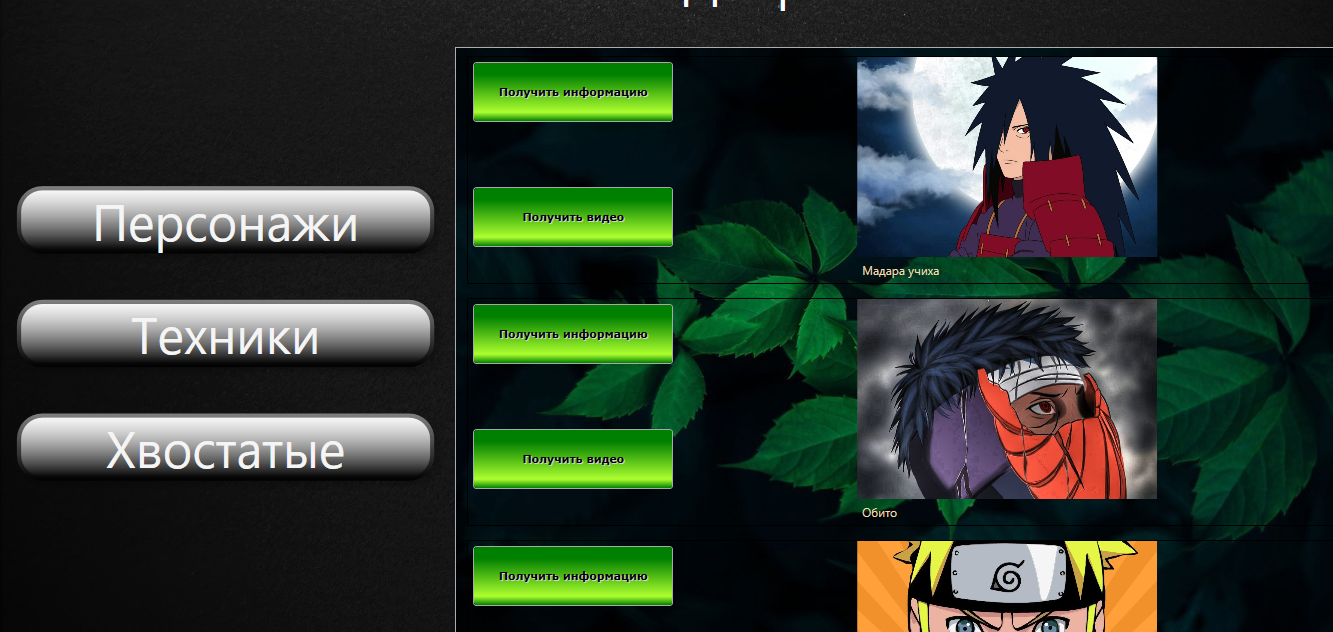


Рисунок 37.

На рисунке 37. Вы можете увидеть страницу с персонажами.

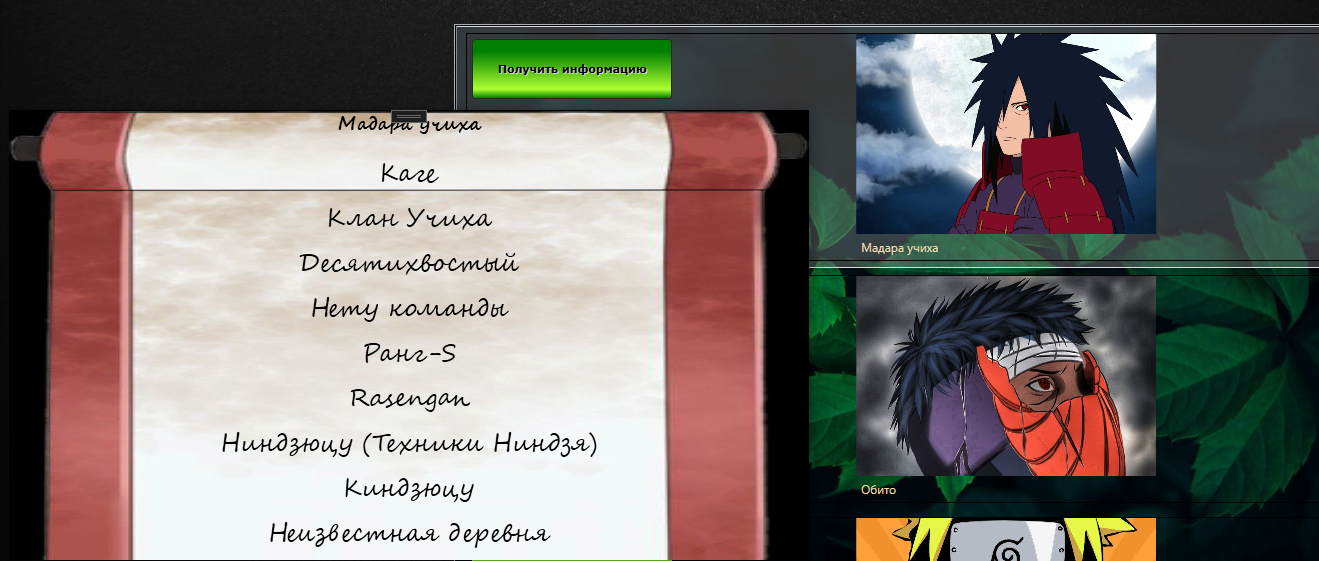


Рисунок 38.

Выберите персонажа и нажмите на кнопку получить информацию, после чего вы сможете получить информацию о персонаже как на рисунке 38.

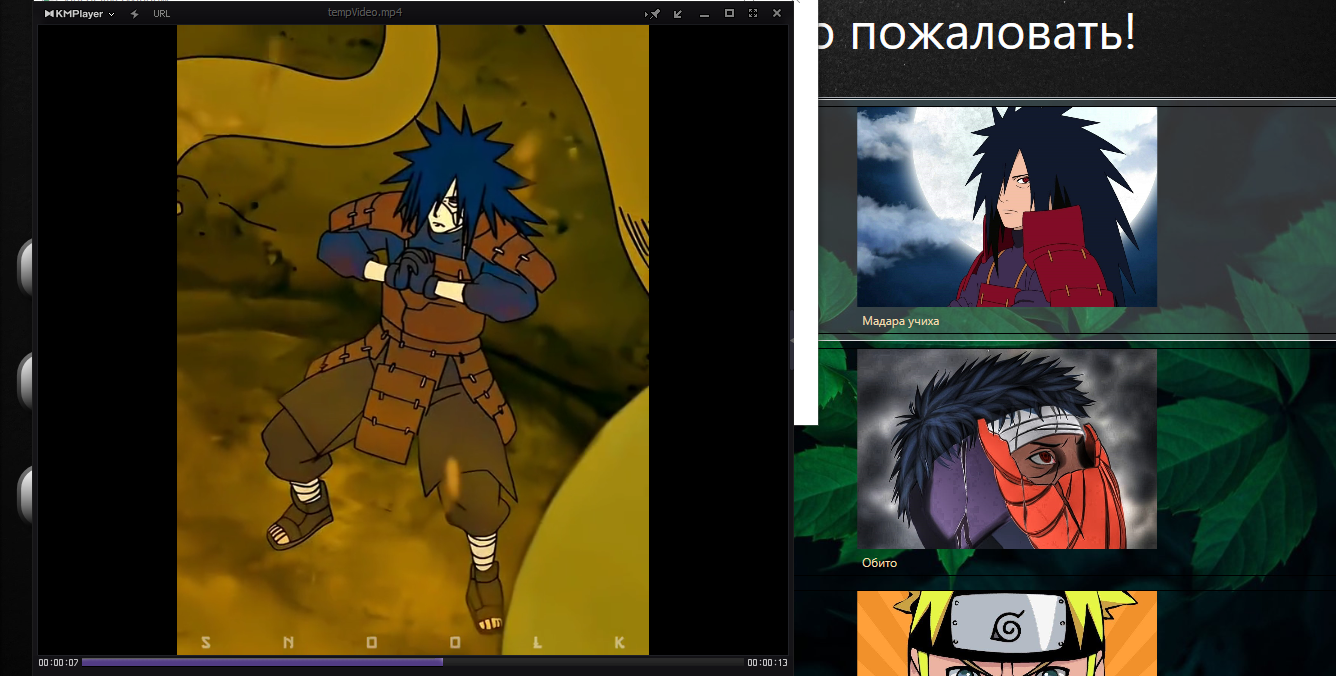


Рисунок 39.



Рисунок 40.

Что бы получить видео о персонаже вам необходимо воспользоваться стороннем приложением KMP Player.

Что бы получить видео о персонаже нажмите на кнопку получить видео, после чего вам откроется окно с видео о персонаже как на картинке 39. и 40.

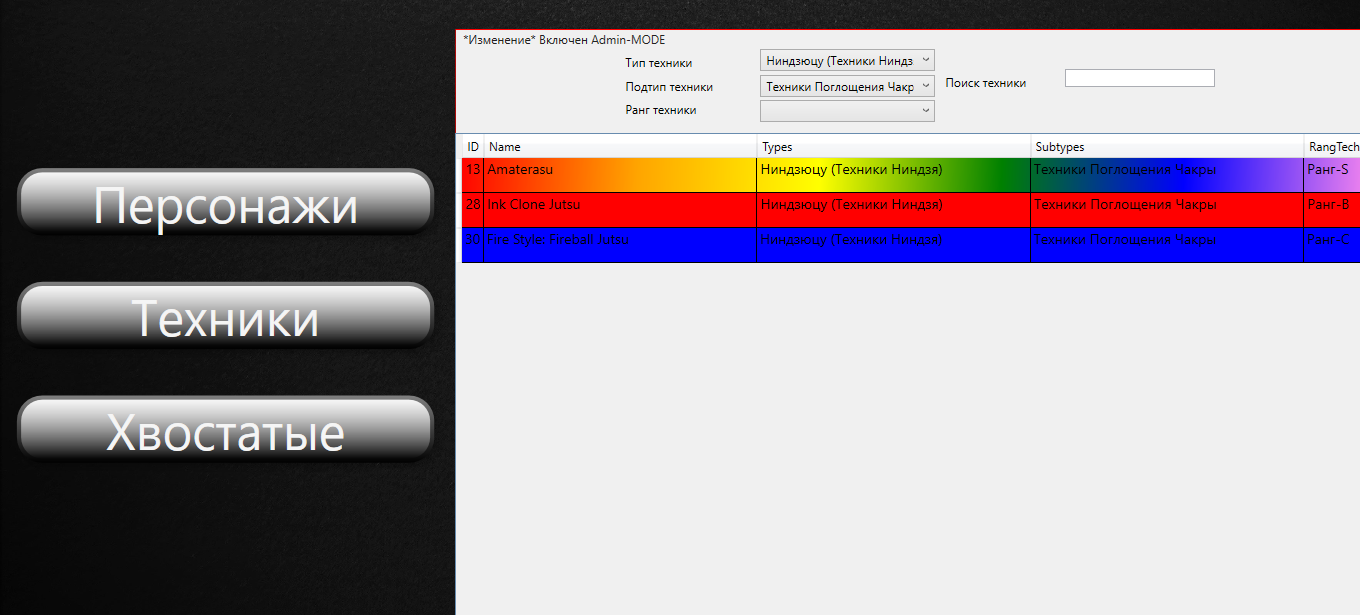


Рисунок 41.

Для того что бы увидеть информацию о техниках вам необходимо нажать на кнопку техники слева в меню навигации. Результат можете увидеть на рисунке 41.

Вы так же можете пользоваться фильтрами и поиском техники для нахождения определённой техники.

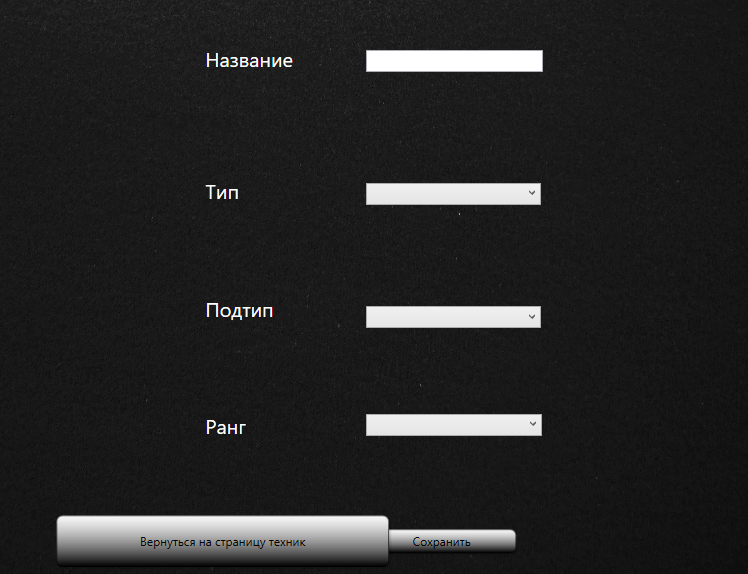


Рисунок 42.

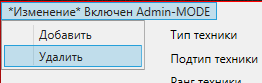


Рисунок 43.

Администратор обладает возможность пользоваться меню добавления с удобным и быстрым добавлением данных с помощью комбобоксов, а так же обладает возможностью удаления данных из таблицы пример на рисунке 42 и 43.

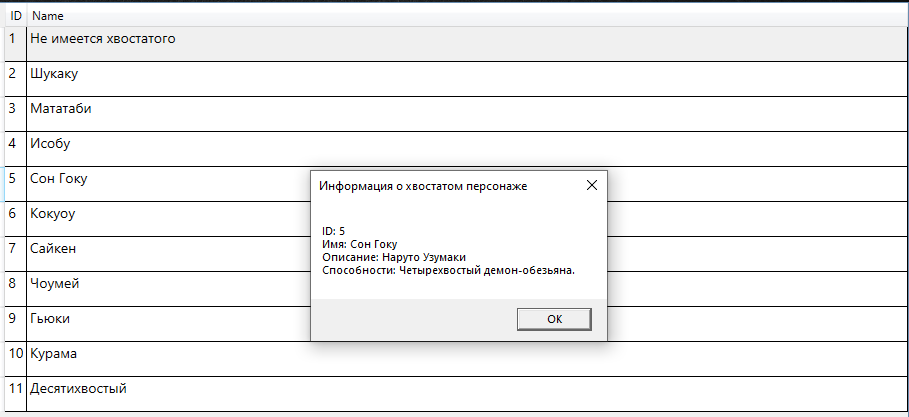


Рисунок 44.

На странице хвостатых вы можете получить информацию об одном из десятихвостых. Для того что бы получить информацию об одном из зверей, вам необходимо навести курсор на строчку со зверем и произвести нажатие ПКМ.

ЗАКЛЮЧЕНИЕ

Курсовой проект по разработке мини-вики для мира Наруто, «NarutoViki», нацелен на автоматизацию сбора, хранения и обработки информации о персонажах, техниках, хвостатых созданиях и других аспектах этой вселенной. Реализация проекта позволила успешно достичь поставленных целей и решить следующие задачи:

1. Анализ предметной области и определение требований пользователя.
2. Создание концептуальной и логической моделей базы данных для эффективного хранения информации.
3. Разработка серверной части информационной системы, обеспечивающей взаимодействие с базой данных.
4. Реализация клиентского приложения, предоставляющего доступ к информации о персонажах, тефхниках и других элементах мира Наруто.

При проектировании информационной подсистемы учтены базовые потребности пользователей, однако в связи с учебным характером проекта не все данные и функциональные возможности мира Наруто были включены. Была обеспечена нормализация базы данных для поддержания целостности и соответствия информации.

Разработанная информационная система предоставляет удобный доступ к информации о персонажах, техниках и других аспектах мира Наруто, а также может быть расширена для включения дополнительных функций в будущем. Курсовой проект стал основой для создания функционально насыщенной и информативной системы, предназначенной для поклонников вселенной Наруто.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. <https://jut.su/naruuto/>
2. <https://naruto.wiki/%D0%94%D0%B7%D1%8E%D1%86%D1%83>
3. <https://naruto.fandom.com/wiki/Narutopedia>
4. <https://naruto.wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0>
5. <https://www.kmplayer.com/home>