Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

Пояснительная записка

Листов: 27

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc9503368)

[1. ОБЩАЯ ЧАСТЬ 5](#_Toc9503369)

[1.1. Цель разработки 5](#_Toc9503370)

[1.2. Средства разработки 5](#_Toc9503371)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 6](#_Toc9503372)

[2.1. Постановка задачи 6](#_Toc9503373)

[2.1.1. Входные и выходные данные 6](#_Toc9503374)

[2.1.2. Требования к проекту 6](#_Toc9503375)

[2.2. Внешняя спецификация 8](#_Toc9503376)

[2.2.1. Описание задачи 8](#_Toc9503377)

[2.2.2. Описание входных и выходных данных 8](#_Toc9503378)

[2.2.3. Метод 8](#_Toc9503379)

[2.2.4. Тесты 10](#_Toc9503380)

[2.2.5. Контроль целостности данных 10](#_Toc9503381)

[2.3. Проектирование 11](#_Toc9503382)

[2.3.1. Диаграмма прецедентов 11](#_Toc9503383)

[2.3.2. Диаграмма классов 12](#_Toc9503384)

[2.3.3. Функциональная схема 13](#_Toc9503385)

[2.3.4. Структурная схема 14](#_Toc9503386)

[2.3.5. Схема данных 15](#_Toc9503387)

[2.3.6. Схема интерфейса 17](#_Toc9503388)

[2.3.7. Схема и подход к тестированию 17](#_Toc9503389)

[2.3.8. Результат работы программы 18](#_Toc9503390)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 19](#_Toc9503391)

[3.1. Инструментальные средства 19](#_Toc9503392)

[3.2. Отладка программы 20](#_Toc9503393)

[3.3. Защитное программирование 21](#_Toc9503394)

[3.4. Характеристика программы 21](#_Toc9503395)

[ЗАКЛЮЧЕНИЕ 24](#_Toc9503396)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 26](#_Toc9503397)

ПРИЛОЖЕНИЕ А. Эскизный проект

ПРИЛОЖЕНИЕ Б. Сценарий тестов и результаты тестовых испытаний

ПРИЛОЖЕНИЕ В. Исходный текст программы

ПРИЛОЖЕНИЕ Г. Руководство пользователя

ПРИЛОЖЕНИЕ Д. Результаты проверки на уникальность

ВВЕДЕНИЕ

Базы данных неотъемлемая часть IT и уже с трудом можно представить хранение каких-либо данных иными и менее надежными способами. Ведь использование БД встречается буквально на каждом шагу, будь это поиск информации в интернете или же ввод своих личных данных, а именно логина и пароля, для входа на различные сервисы. Поэтому во всем мире базы данных имеет и представляет особую популярность, из-за чего их созданию и проектированию уделяется особое внимание.

Проектирование баз данных - процесс создания схемы базы данных и определения необходимых ограничений целостности, которое заключается в многоступенчатом описании будущей БД с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация ее структуры. Его основными задачами являются:

1. Обеспечение хранения в БД всей необходимой информации;
2. Обеспечение возможности получения данных по всем необходимым запросам;
3. Сокращение избыточности и дублирования данных;
4. Обеспечение целостности базы данных.

На данный момент распространённым CASE-средством для проектирования и документирования баз данных является AllFusion ERwin Data Modeler, который обладает большим набором возможностей, однако возникла необходимость создания более гибкой системы генерации скриптов и документации для баз данных, в связи с чем появилась идея разработки программного обеспечения «ScriptGen» с возможностью написания шаблона генерации при помощи простых команд, а также с возможностью расширять функционал при помощи системы плагинов.

1. ОБЩАЯ ЧАСТЬ

1.1. Цель разработки

Цель разработки программного обеспечения - упростить процесс создания базы данных и приложения для работы с ней.

1.2. Средства разработки

Разработка производилось на ноутбуке Dell Inspiron 5379, характеристики которого представлены в таблице 1.

Таблица 1 - Аппаратные средства

|  |  |
| --- | --- |
| Название компонента | Характеристики |
| Процессор | Intel Core i5 7200U |
| Графический чип | Intel HD Graphics 620 |
| Оперативная память | DDR-4 8GB |
| Жёсткий диск | Объём 1 Тб |

Во время разработки использовались программные средства, представленные в таблице 2.

Таблица 2 - Программные средства

|  |  |
| --- | --- |
| Название | Краткое описание |
| Visual Studio 2017 | Среда разработки |
| Microsoft Word | Текстовый процессор |
| Microsoft Visio | Векторный графический редактор |

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Постановка задачи

Разработать программное обеспечения для проектирования баз данных по средствам графического интерфейса и дальнейшего создания скрипта базы данных и дополнительного скрипта для работы с базой данных по шаблону.

2.1.1. Входные и выходные данные

Входные данные: графическая модель, созданная в приложении, настройки приложения и шаблоны для генерации скриптов, файлы плагинов.

Выходные данные: файл графической модели, скрипт базы данных и дополнительный скрипт.

2.1.2. Требования к проекту

Программное обеспечение должно обеспечивать следующие функциональные возможности:

1. Интерактивное создание модели базы данных;

Создание сущностей, связей между ними, атрибутов и их параметров (тип данных, является ли поле уникальным и может ли поле содержать null) при помощи графического интерфейса. Также названия сущностей и атрибутов должны быть представлены в трёх видах (физическое, логическое, программное).

1. Сохранение и загрузка созданной модели;

Сохранение созданных моделей в файл (xml) и последующая загрузка модели из данного файла.

1. Проверка модели на наличие ошибок;

Осуществление проверки модели на уникальность названий атрибутов и сущностей и наличие запрещённых слов и символов в названиях атрибутов и сущностей.

1. Генерация скриптов на основе модели по шаблону;

Создание скрипта базы данных и любых дополнительных скриптов по шаблону, указанному в настройках приложения.

1. Добавление новых шаблонов;

Добавление пользовательских шаблонов в программное обеспечение.

1. Добавление настроек приложения, которые включают типы данных, которые подсказывает приложение и список слов, запрещённых в названии сущностей и атрибутов.
2. Генерация случайных значений по указанному шаблону;

Генерация случайных данных с использованием шаблона генерации.

1. Добавление плагинов.

Возможность написания и внедрения плагинов в программное обеспечение.

Формат данных, используемых в приложении описан в пункте 2.2.2. «Описание входных и выходных данных».

Программное обеспечение должно соответствовать макету, представленному на рисунке 1-2. Более подробное описание интерфейса приведено в ПРИЛОЖЕНИИ А. «Эскизный проект».

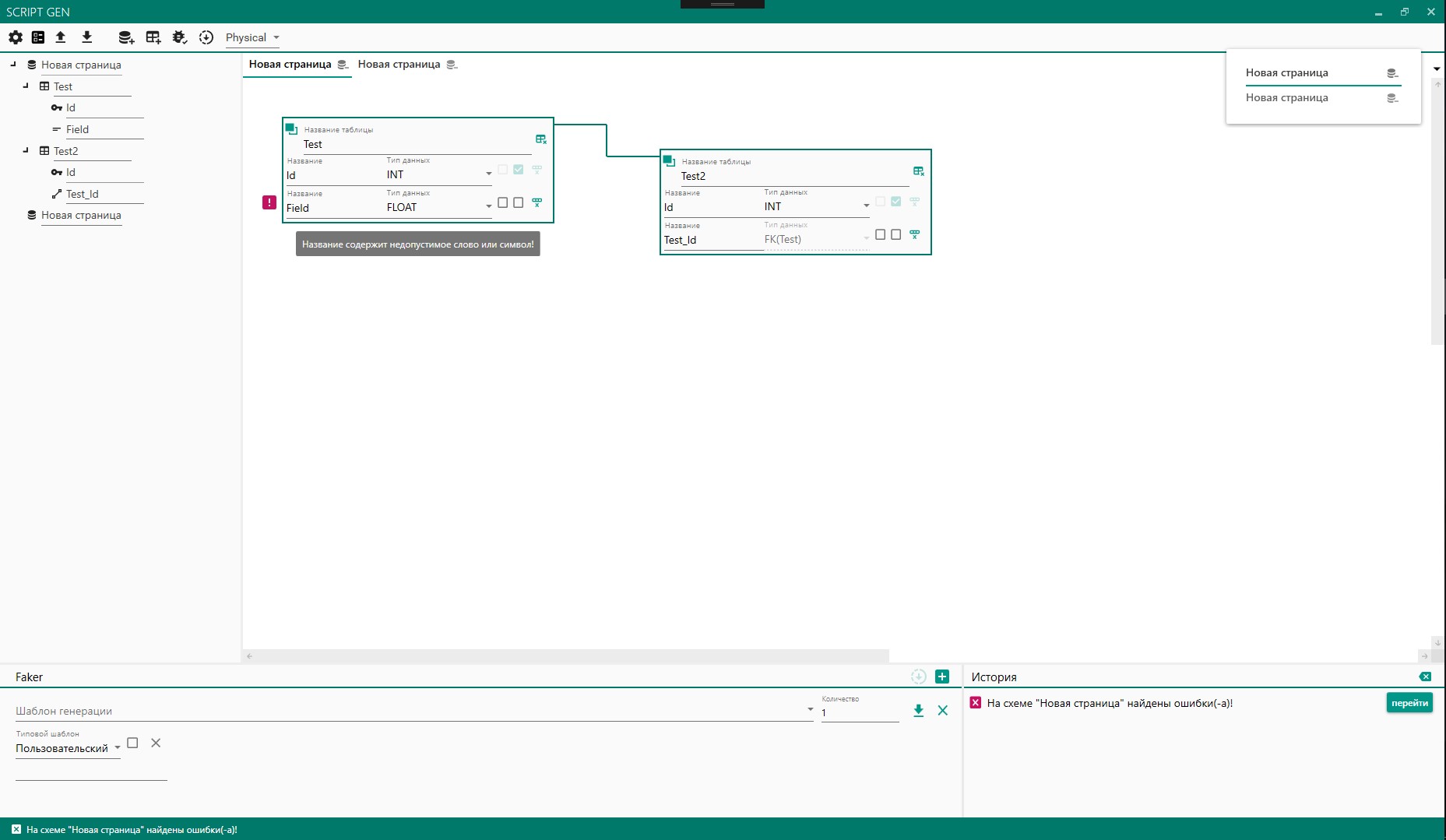


Рисунок 1 - Главное окно приложения

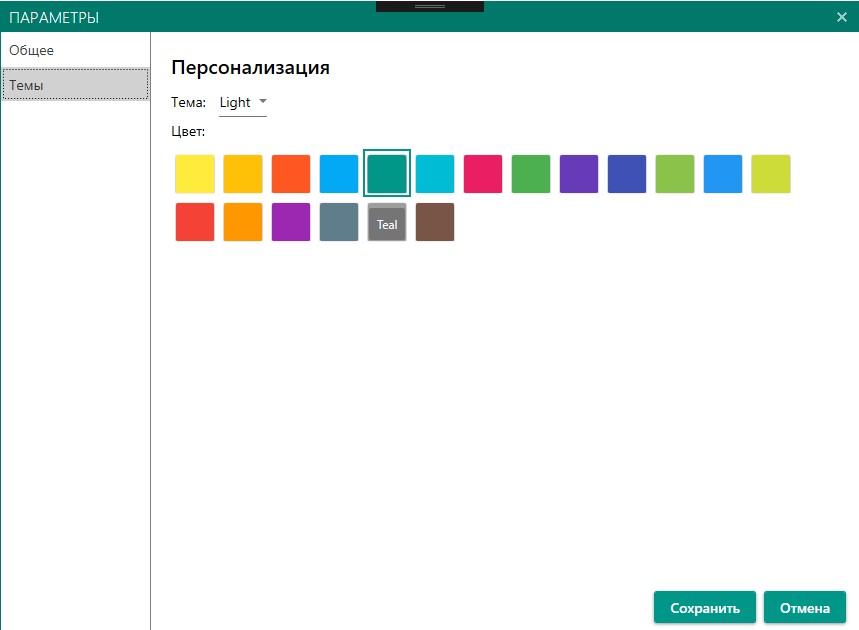
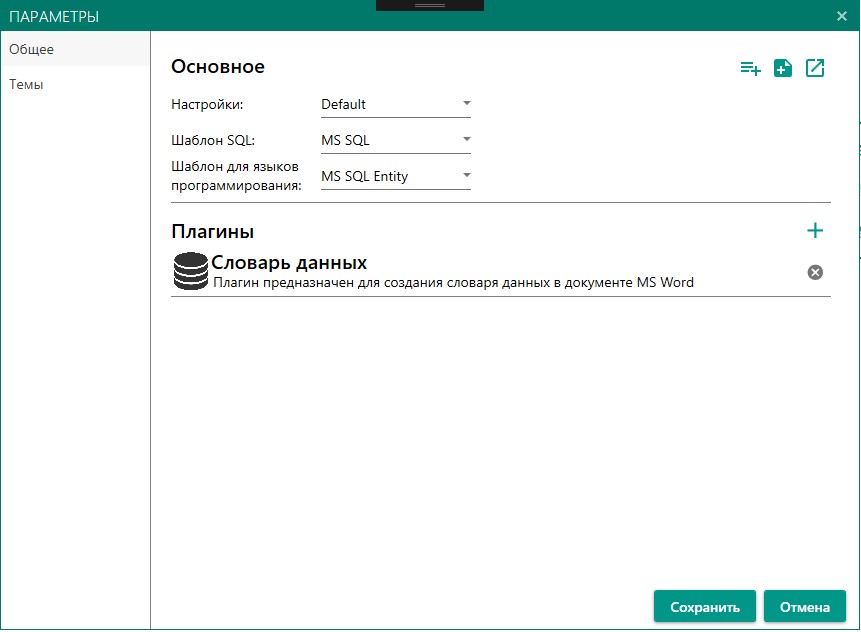


Рисунок 2 - Настройки приложения

2.2. Внешняя спецификация

2.2.1. Описание задачи

Необходимо предусмотреть сбор данных из интерактивной схемы в модель для дальнейшей обработки. Также необходима возможность воссоздания интерактивной схемы из сохранённой модели.

Программа должна предоставлять пользователю возможности, представленные в пункте 2.3.1. «Диаграмма прецедентов».

2.2.2. Описание входных и выходных данных

Программа получает на вход данные введённые пользователем при помощи графического интерфейса, а также файлы в формате «xml».

Программа создаёт на выходе файл «sql» и «txt» в которых находятся скрипты, полученные в результате обработки модели, а также файл модели при сохранении.

Структура указанных файлов приведена в пункте 2.3.5. «Схема данных»

2.2.3. Метод

При разработки программного обеспечения «ScriptGen» был использован шаблон проектирования архитектуры приложения MVVM.

Шаблон MVVM делится на три части:

1. Модель (англ. Model) представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.
2. Представление (англ. View) — графический интерфейс. Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
3. Модель Представления (англ. ViewModel) — с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащие связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Также разработка велась с использованием принципов объектно-ориентированного программирования.

Объектно-ориентированное программирование - парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Выделяют следующие принципы ООП:

* + - 1. Абстракция данных — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения не значимые. Соответственно, абстракция — это набор всех таких характеристик.

1. Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.
2. Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом
3. Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

2.2.4. Тесты

В таблице 3 приведено тестирование генерации скриптов, по созданной модели. В таблице приведены возможные варианты заполнения модели и реакция программы при генерации скриптов.

Таблица 3 - Тесты

|  |  |  |
| --- | --- | --- |
| Входные данные | Комментарий | Результат |
|  | Генерация скриптов при не заполненной модели. | Выведено сообщение об ошибках модели: Не указанно название таблицы, поля, не указан тип данных поля. |
|  | Генерация скриптов при использовании некорректное слово. | Выведено сообщение об ошибках модели: Недопустимое слово или символ в названии таблицы. |
|  | Генерация скриптов при модели с одинаковыми названиями. | Выведено сообщение об ошибках модели: Названия не могут совпадать. |
|  | Генерация скриптов при корректной модели. | Выведено сообщение о сохранении скриптов. |
| Отсутствует модель. | Генерация скриптов при отсутствии модели. | Кнопка для генерации скриптов заблокирована, запуск генерации невозможен. |

2.2.5. Контроль целостности данных

В таблице 4 приведены аномалии и реакция программы на них.

Таблица 4 - Контроль целостности данных

| Ситуация | Аномалия | Результат | Комментарий |
| --- | --- | --- | --- |
| Генерация скрипта по шаблону. | Команды шаблона содержат ошибку в синтаксисе. | Блок шаблона, содержащий ошибку, не выполнится. | При генерации, шаблон разбивается на независимые блоки и интерпретируется в результат, если команды содержат синтаксическую ошибку то блок кода вставится в результат без изменений. |
| Установка плагина. | Конфигурация плагина содержит ошибку. | Выведется сообщение о невозможности установить плагин. | Установка плагина, происходит путём локального копирования его файлов, основываясь на файле конфигурации, после чего исполняемый файл подгружается в оперативную память программы. Если во время этого процесса происходит ошибка, скопированные файлы удаляются и выводится сообщение об ошибке. |

2.3. Проектирование

2.3.1. Диаграмма прецедентов

Пользователю приложения «ScriptGen» доступны следующие возможности, представленные на рисунке 3, а их описание приведено в таблице 5.



Рисунок 3 - Диаграмма прецедентов

Таблица 5 - Описание диаграммы прецедентов

| Действие | Описание |
| --- | --- |
| Сохранение и загрузка модели базы данных | Сохранение интерактивной модели в файл и её последующее восстановление. |
| Генерация скриптов | Создание скриптов по шаблону, используя данные из модели. |
| Создание модели базы данных | Создание сущностей, атрибутов, и их параметров (тип данных, является ли поле уникальным и может ли поле содержать null). |
| Добавление шаблонов | Загрузка пользовательских шаблонов. |
| Генерация случайных данных | Создание случайных данных по шаблону. |
| Добавление настроек | Добавление пользовательских настроек приложения. |

2.3.2. Диаграмма классов

Структура классов программного обеспечения представлена на рисунке 4

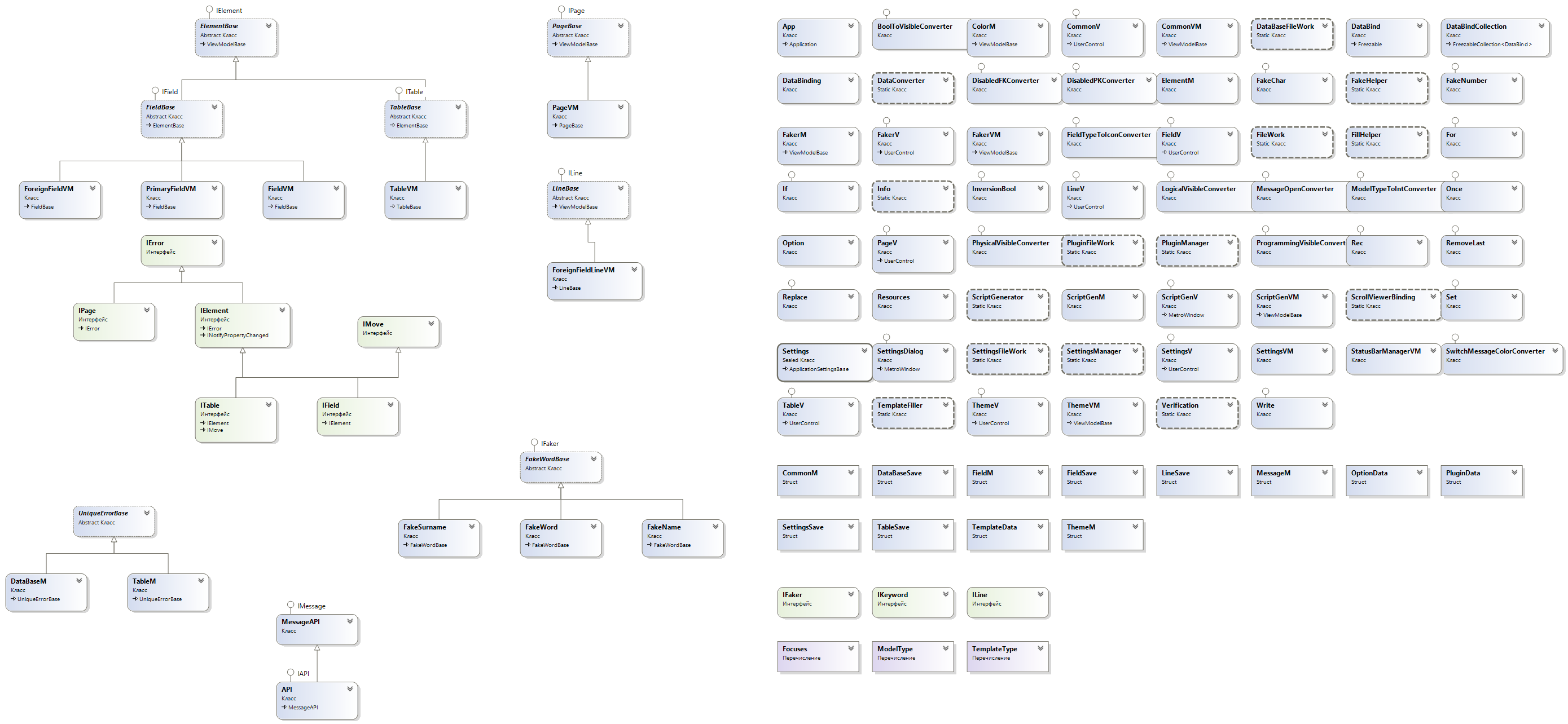


Рисунок 4 - Диаграмма классов

2.3.3. Функциональная схема

Основные функциональные возможности программного обеспечения представлены на рисунке 5.

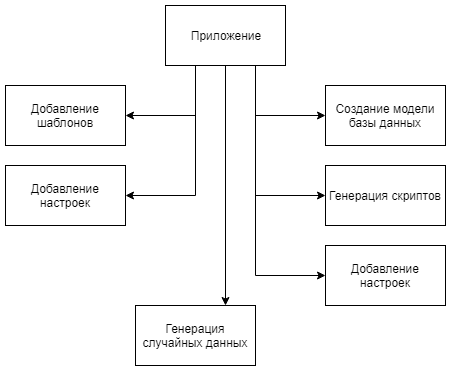


Рисунок 5 - Функциональная схема

.

2.3.4. Структурная схема

На структурной схеме программного обеспечения, представленной на рисунке 6, продемонстрирована зависимость модулей программы, а в таблицы 6 приведено назначение основных групп модулей.

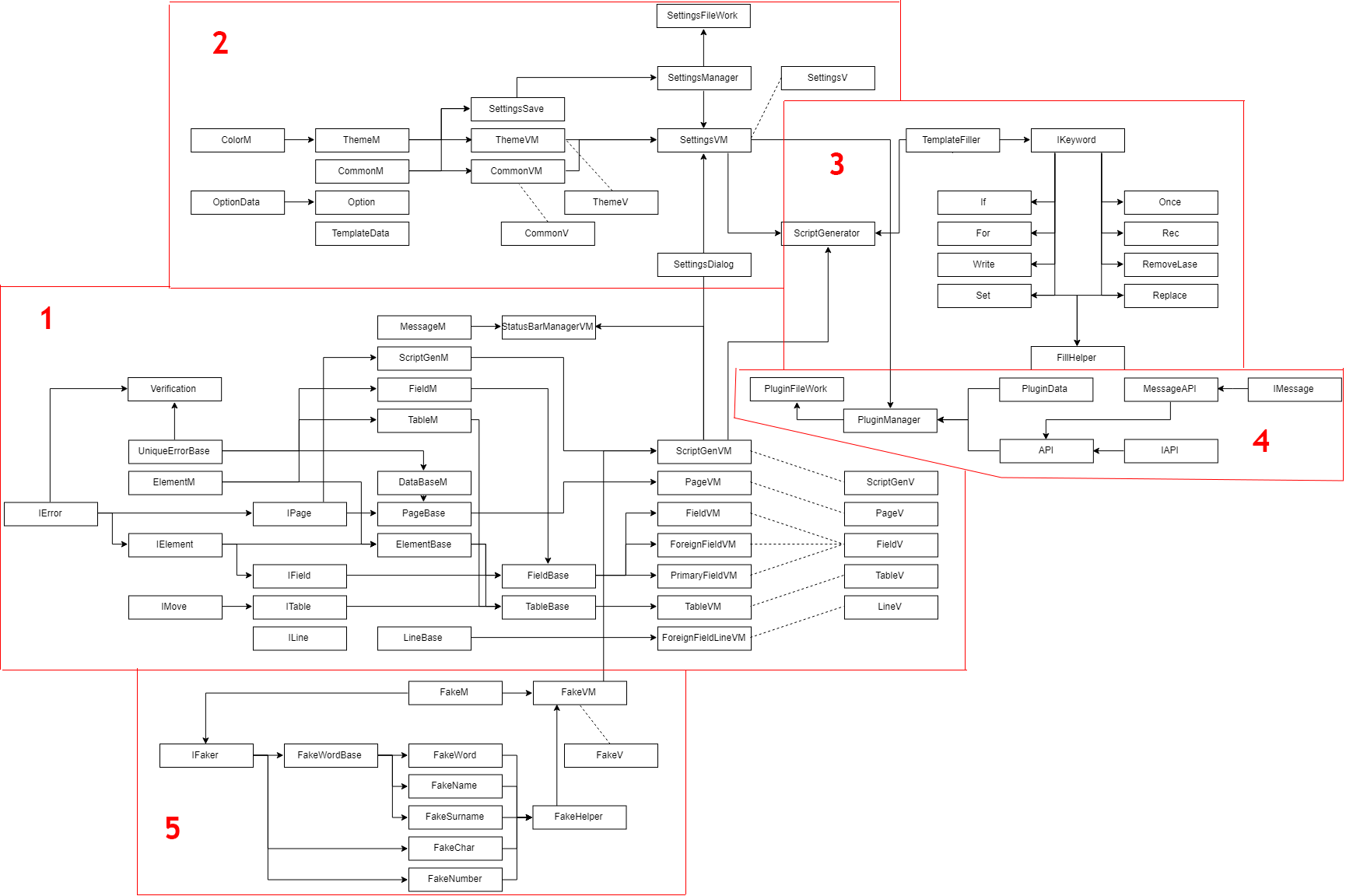


Рисунок 6 - Структурная схема

Таблица 6 - Описание модулей приложения

|  |  |  |
| --- | --- | --- |
| Номер | Название | Описание |
| 1 | Main | Осуществляет сбор данных и контролирует взаимодействие пользователя с функциональными возможностями приложения. |
| 2 | Settings | Осуществляет действия с настройками приложения, а также позволяет добавлять шаблоны для генерации скриптов. |
| 3 | Generator | Осуществляет генерацию скриптов по шаблону с использованием данных модели. |
| 4 | API | Осуществляет работу с плагинами приложения. |
| 5 | Faker | Осуществляет создание случайных данных по шаблону. |

2.3.5. Схема данных

Программное обеспечение «ScriptGen» использует несколько типов файлов, структура которых приведена ниже.

1. Настройки приложения

<OptionData>

<Name>Название варианта настроек</Name>

<FalseWords>Перечисление слов, запрещённых в названии атрибутов и сущностей </FalseWords>

<SQLTypes>SQL типы данных для подсказки </SQLTypes>

<ProgrammingTypes>C# типы данных для подсказки</ProgrammingTypes>

</OptionData>

1. Файл шаблона для генерации скрипта

<TemplateData>

<Name>Название шаблона</Name>

<Type>Тип шаблона (SQL или Programming)</Type>

<Template>Шаблон</Template>

</TemplateData>

1. Конфигурация плагина

<PluginData >

<DllPath>Путь к файлу dll </DllPath>

<Namespace>Главное пространство имён внутри файла dll</Namespace>

<ClassName>Стартовый класс плагина</ClassName>

<Files>

[<string>Список дополнительных файлов </string>]

</Files>

<Name>Название плагина</Name>

<Description>Описание плагина</Description>

<ImagePath>Путь к иконке плагина</ImagePath>

</PluginData>

1. Файл сохранённой модели

<DataBaseSave>

<Name>Название модели</Name>

<Tables>

<TableSave>

<Id>Id таблицы</Id>

<Width>Ширина таблицы</Width>

<Margin>

<Left>Отступ слева</Left>

<Top>Отступ сверху</Top>

<Right> Отступ справа</Right>

<Bottom>Отступ снизу</Bottom>

</Margin>

<Name>Название сущности</Name>

<Fields>

<FieldSave>

<Type>Тип атрибута</Type>

<Name>Название атрибута</Name>

<DataType>Тип данных</DataType>

<IsNull> </IsNull>

<IsUnique> </IsUnique>

<RefTableId>Id таблицы (если внешний ключ)</RefTableId >

</FieldSave>

</Fields>

</TableSave>

</Tables>

<Lines>

<LineSave>

<Field>

<Type>Тип атрибута</Type>

<Name>Название атрибута</Name>

<DataType>Тип данных</DataType>

<IsNull> </IsNull>

<IsUnique> </IsUnique>

<RefTableId>Id таблицы (если внешний ключ)</RefTableId >

</Field>

<SourceId>Источник связи</SourceId>

<TargetId>Цель связи</TargetId>

<SourceX1>Координата X начала линии</SourceX1>

<TargetX1>Координата X конца линии </TargetX1>

<SourceY1>Координата Y начала линии </SourceY1>

<TargetY1>Координата Y конца линии </TargetY1>

<ConnectionX1> Координата X центральной части линии </ConnectionX1>

</LineSave>

</Lines>

</DataBaseSave>

2.3.6. Схема интерфейса

На рисунке приведена схема пользовательского интерфейса, пользователю доступно 2 окна, главное и окно настроек, переход на которое осуществляется при нажатии на кнопку в меню.

Главное окно позволяет создавать интерактивную модель базу данных, генерировать случайные значения, а также создавать скрипты по шаблону.

Окно настроек позволяет выбирать шаблоны для генерации, действующий экземпляр настроек и загружать новые плагины.

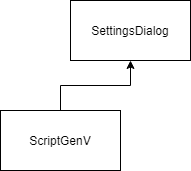


Рисунок 7 - Схема интерфейса

2.3.7. Схема и подход к тестированию

На рисунке 8 представлена схема тестирования приложения «ScriptGen». Описание проведенных тестов, а также их результаты приведены в ПРИЛОЖЕНИИ Б. «Сценарий тестов и результаты тестовых испытаний».

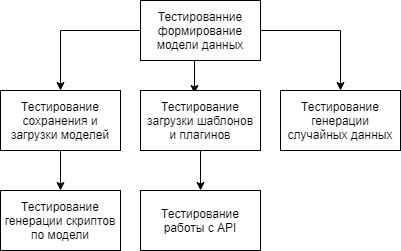


Рисунок 8 - Схема тестирования

* + 1. Результат работы программы

Ниже продемонстрирован результат работы программы, более подробное описание работы программы приведено в ПРИЛОЖЕНИИ Г. «Руководство пользователя».

На рисунке 9 продемонстрированно отображение ошибки при неверном заполнении модели.

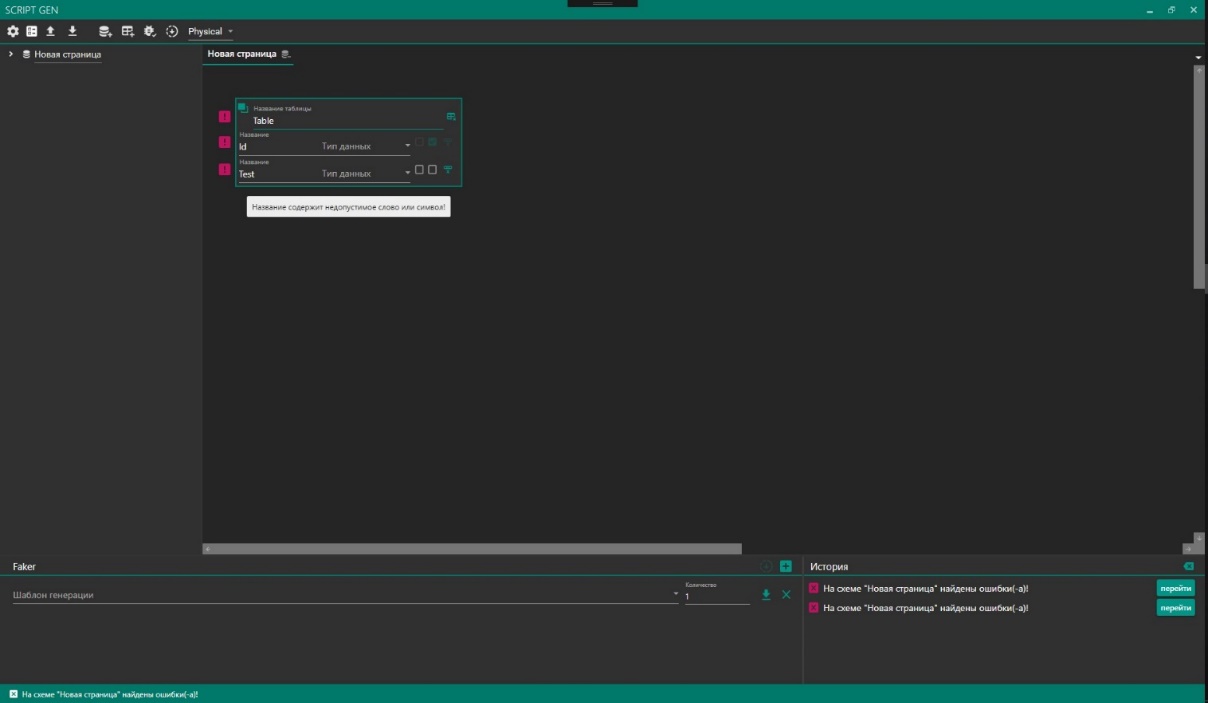


Рисунок 9 - Отображение сообщения об ошибке

На рисунке 10 продемонстрирован результат генерации скрипта.

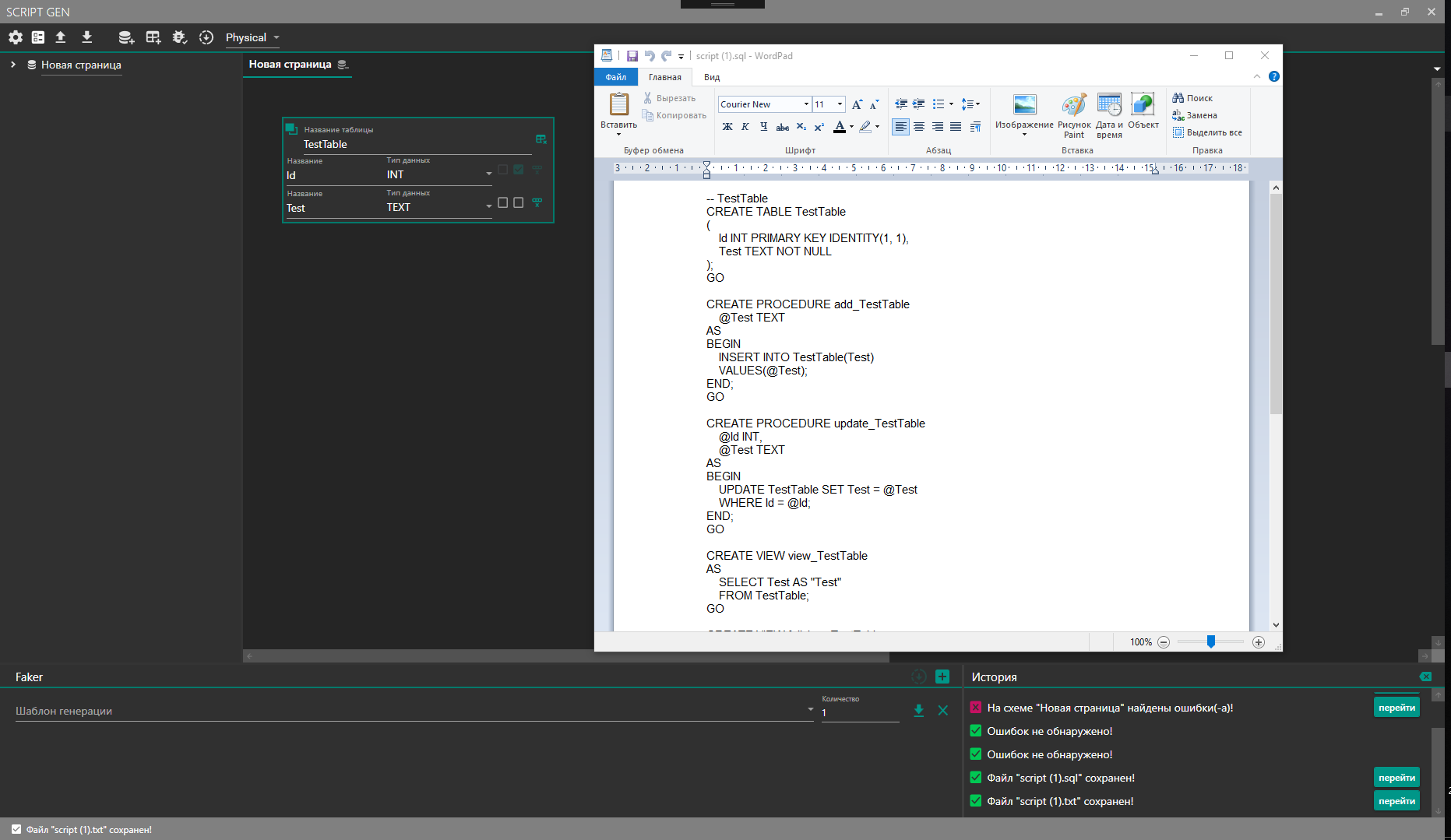


Рисунок 10 - Результат работы программы

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1. Инструментальные средства

Для разработки программного продукта использовалась интегрированная среда «Microsoft Visual Studio».

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Microsoft Visio — векторный графический редактор, редактор диаграмм и блок-схем для Windows. Предоставляет набор новых функций, которые делают создание диаграмм более наглядными, включая новые и обновленные фигуры и наборы элементов, улучшенные эффекты и темы, а также функцию соавторства, которая упрощает работу в команде.

3.2. Отладка программы

Во время отладки программы использовались встроенные средства программной среды «Visual Studio 2017 Community».

Во время разработки распространёнными ошибками были:

1. Неверное заполнение полей модели. Как правило ошибка заключалась в обращении к неверным полям.

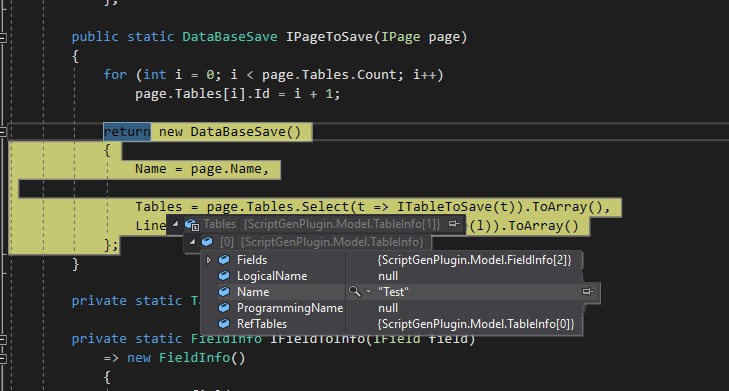


Рисунок 11 - Проверка заполнения модели

1. Неверное разбитие шаблона на блоки кода для обработки.

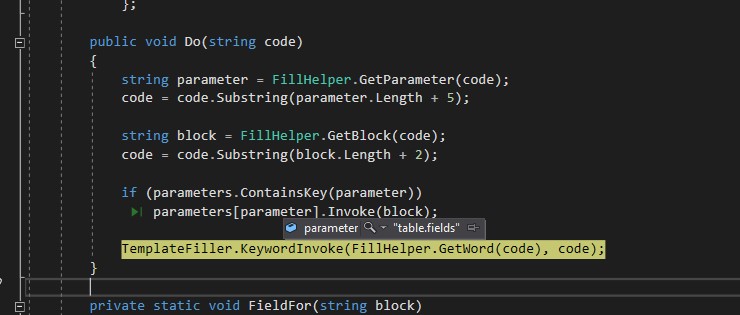


Рисунок 12 - Проверка параметров команды шаблона

3.3. Защитное программирование

В качестве защитного программирования производилась проверка корректности данных модели, а также обработка исключений при работе с файлами:

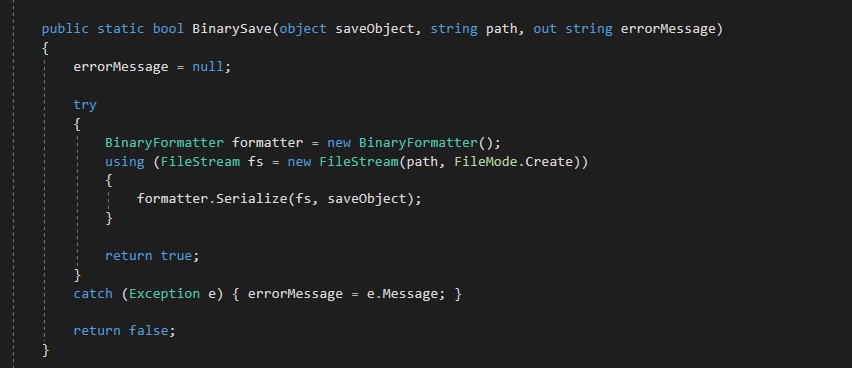


Рисунок 13 - Обработка исключений при работе с файлами

3.4. Характеристика программы

В таблице 7 представлены характеристики разработанного программного обеспечения Указаны модули и их размер.

Таблица 7 - Характеристики программы

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Наименование модуля программы | Строчек кода | Размер (Кб) |
| 1 | App.xaml | 56 | 3 |
| 2 | ElementBase.cs | 189 | 6 |
| 3 | FakeWordBase.cs | 46 | 1 |
| 4 | FieldBase.cs | 190 | 6 |
| 5 | LineBase.cs | 143 | 4 |
| 6 | PageBase.cs | 254 | 7 |
| 7 | TableBase.cs | 227 | 7 |
| 8 | UniqueErrorBase.cs | 32 | 1 |
| 9 | API.cs | 73 | 3 |
| 10 | MessageAPI.cs | 124 | 7 |
| 11 | PluginData.cs | 99 | 3 |
| 12 | PluginManager.cs | 61 | 2 |
| 13 | BoolToVisibleConverter.cs | 16 | 1 |
| 14 | CSharpVisibleConverter.cs | 18 | 1 |
| 15 | DisabledFKConverter.cs | 17 | 1 |
| 16 | DisabledPKConverter.cs | 17 | 1 |
| 17 | FieldTypeToIconConverter.cs | 25 | 1 |
| 18 | InversionBool.cs | 15 | 1 |
| 19 | LogicalVisibleConverter.cs | 18 | 1 |
| 20 | MessageOpenConverter.cs | 15 | 1 |
| 21 | ModelTypeToIntConverter.cs | 17 | 1 |
| 22 | PhysicalVisibleConverter.cs | 18 | 1 |
| 23 | SwitchMessageColorConverter.cs | 29 | 1 |
| 24 | ReadonlyBinding.cs | 51 | 2 |
| 25 | ScrollViewerBinding.cs | 92 | 4 |
| 26 | DataBaseFileWork.cs | 46 | 2 |
| 27 | DataConverter.cs | 129 | 4 |
| 28 | Enums.cs | 67 | 2 |
| 29 | FileWork.cs | 172 | 6 |
| 30 | Info.cs | 170 | 6 |
| 31 | PluginFileWork.cs | 155 | 6 |
| 32 | SettingsFileWork.cs | 144 | 6 |
| 33 | Verification.cs | 86 | 5 |
| 34 | DataBaseSave.cs | 10 | 1 |
| 35 | FieldSave.cs | 21 | 1 |
| 36 | LineSave.cs | 17 | 1 |
| 37 | TableSave.cs | 18 | 1 |
| 38 | FakeHelper.cs | 40 | 1 |
| 39 | FakeChar.cs | 56 | 2 |
| 40 | FakeName.cs | 24 | 1 |
| 41 | FakeNumber.cs | 43 | 1 |
| 42 | FakeSurname.cs | 24 | 1 |
| 43 | FakeWord.cs | 24 | 1 |
| 44 | FakerM.cs | 153 | 5 |
| 45 | FakerV.xaml | 335 | 14 |
| 46 | FakerVM.cs | 144 | 4 |
| 47 | FillHelper.cs | 81 | 3 |
| 48 | ScriptGenerator.cs | 59 | 2 |
| 49 | TemplateFiller.cs | 185 | 7 |
| 50 | For.cs | 66 | 2 |
| 51 | If.cs | 91 | 3 |
| 52 | Once.cs | 26 | 1 |
| 53 | Rec.cs | 68 | 2 |
| 54 | RemoveLast.cs | 18 | 1 |
| 55 | Replace.cs | 19 | 1 |
| 56 | Set.cs | 37 | 1 |
| 57 | Write.cs | 52 | 2 |
| 58 | IElement.cs | 68 | 2 |
| 59 | IError.cs | 23 | 1 |
| 60 | IFaker.cs | 9 | 1 |
| 61 | IField.cs | 62 | 2 |
| 62 | IKeyword.cs | 14 | 1 |
| 63 | ILine.cs | 51 | 2 |
| 64 | IMove.cs | 55 | 2 |
| 65 | IPage.cs | 86 | 3 |
| 66 | ITable.cs | 59 | 2 |
| 67 | DataBaseM.cs | 41 | 1 |
| 68 | ElementM.cs | 38 | 2 |
| 69 | FieldM.cs | 44 | 2 |
| 70 | MessageM.cs | 28 | 1 |
| 71 | ScriptGenM.cs | 17 | 1 |
| 72 | TableM.cs | 43 | 1 |
| 73 | AssemblyInfo.cs | 55 | 3 |
| 74 | Resources.Designer.cs | 456 | 13 |
| 75 | Settings.Designer.cs | 26 | 1 |
| 76 | SettingsDialog.xaml | 18 | 1 |
| 77 | SettingsManager.cs | 66 | 2 |
| 78 | ColorM.cs | 45 | 1 |
| 79 | CommonM.cs | 12 | 1 |
| 80 | Option.cs | 28 | 1 |
| 81 | OptionData.cs | 12 | 1 |
| 82 | SettingsSave.cs | 21 | 1 |
| 83 | TemplateData.cs | 12 | 1 |
| 84 | ThemeM.cs | 8 | 1 |
| 85 | CommonV.xaml | 216 | 9 |
| 86 | SettingsV.xaml | 60 | 2 |
| 87 | ThemeV.xaml | 68 | 3 |
| 88 | CommonVM.cs | 131 | 4 |
| 89 | SettingsVM.cs | 53 | 1 |
| 90 | ThemeVM.cs | 95 | 2 |
| 91 | FieldV.xaml | 211 | 10 |
| 92 | LineV.xaml | 49 | 2 |
| 93 | PageV.xaml | 68 | 3 |
| 94 | ScriptGenV.xaml | 496 | 26 |
| 95 | TableV.xaml | 203 | 9 |
| 96 | FieldVM.cs | 41 | 1 |
| 97 | ForeignFieldLineVM.cs | 97 | 3 |
| 98 | ForeignFieldVM.cs | 61 | 2 |
| 99 | PageVM.cs | 146 | 4 |
| 100 | PrimaryFieldVM.cs | 30 | 1 |
| 101 | ScriptGenVM.cs | 238 | 7 |
| 102 | StatusBarManagerVM.cs | 150 | 5 |
| 103 | TableVM.cs | 89 | 2 |

ЗАКЛЮЧЕНИЕ

В ходе данной разработки был создан программный продукт, цель использования которого обязуется упростить разработку и документирование баз данных. Данный продукт, названный как «ScriptGen», предоставляет все необходимые возможности, которые были предъявлены на основании данных заказчика, предоставленных для разработки, и представляет собой более гибкую систему генерации скриптов и документации для баз данных, в отличие от существующего и весьма распространенного на данный момент CASE-средства для проектирования и документирования баз данных AllFusion ERwin Data Modeler.

Главными возможностями данного продукта являются:

1. Интерактивное создание модели базы данных;

Создание сущностей, связей между ними, атрибутов и их параметров (тип данных, является ли поле уникальным и может ли поле содержать null) при помощи графического интерфейса. Также названия сущностей и атрибутов должны быть представлены в трёх видах (физическое, логическое, программное).

1. Сохранение и загрузка созданной модели;

Сохранение созданных моделей в файл (xml) и последующая загрузка модели из данного файла.

1. Проверка модели на наличие ошибок;

Осуществление проверки модели на уникальность названий атрибутов и сущностей и наличие запрещённых слов и символов в названиях атрибутов и сущностей.

1. Генерация скриптов на основе модели по шаблону;
2. Создание скрипта базы данных и любых дополнительных скриптов по шаблону, указанному в настройках приложения.
3. Добавление новых шаблонов;

Добавление пользовательских шаблонов в программное обеспечение.

1. Добавление настроек приложения, которые включают типы данных, которые подсказывает приложение и список слов, запрещённых в названии сущностей и атрибутов.
2. Генерация случайных значений по указанному шаблону;

Генерация случайных данных с использованием шаблона генерации.

1. Добавление плагинов.

Возможность написания и внедрения плагинов в программное обеспечение.

При разработке был предусмотрен сбор данных из интерактивной схемы в модель для дальнейшей обработки с возможностью воссоздания интерактивной схемы из сохранённой модели. А также были использованы некоторые методы, а именно, шаблон проектирования архитектуры приложения MVVM, делящийся на три части: модель, представление и модель представления и принцип объектно-ориентированного программирования, выделяющий такие принципы ООП, как: абстракция данных, инкапсуляция, наследование и полиморфизм.

В рамках разработки программного обеспечения были получены знания по работе с шаблонами со вложенным языком команд, а также был получен опыт во внедрении системы плагинов в данный учебный проект.

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19.404-79 ЕСПД;
2. ГОСТ 19.101-77(входит в состав документов на стадиях разработки эскизного и технического проектов программы);
3. ГОСТ 19.105-78(структура оформления документа);
4. ГОСТ 19.001-77 Единая система программной документации. Общие положения;
5. ГОСТ 19.002-80 Единая система программной документации. Схемы алгоритмов и программ. Правила выполнения;
6. ГОСТ 19.004-80 Единая система программной документации. Термины и определения;
7. ГОСТ 19 105-78 Единая система программной документации. Общие требования к программным документам;
8. C# 4.0 Полное руководство (Герберт Шилдт);
9. C# Программирование на языке высокого уровня (Павловская Т.А.);
10. C# 6.0. Справочник. Полное описание языка (Джозеф Албахари, Бен Албахари);
11. LINQ. Карманный справочник (Джозеф Албахари, Бен Албахари);
12. Самоучитель С#. Начала программирования (Элеонора Ишкова);
13. Программирование на Java и С# для студента (Герман О.В., Герман Ю.О.);
14. Как программировать на Visual C# 2012 (Пол Дейтел, Харви Дейтел);
15. Программирование в тональности С# (Петцольд Ч.);
16. Язык программирования C# (Хейлсберг А.);
17. Изучаем C# (Эндрю Стиллмен);
18. Microsoft Visual C# (Шарп Джон);
19. Язык C#. Краткое описание и введение в технологии программирования (Котов О.М.);
20. Принципы, паттерны и методики гибкой разработки на языке C# (Мартин Р., Мартин М.);
21. Основы программирования на С# 3.0. Ядро языка (Биллиг В.А.);
22. C#. Советы программистам (Климов А.);
23. Язык программирования С# 2008 и платформа .NET 3.5 (Троелсен Эндрю);
24. C# и CSLA .NET Framework. Разработка бизнес-объектов (Рокфорд Лотка);
25. Visual C# 2010 на примерах (Виктор Зиборов);
26. Visual C# 2012 на примерах (Зиборов В.В.);
27. Ахо Компиляторы. Принципы, технологии и инструментарий (Альфред В);
28. C# в кратком изложении (Дж. Бишоп, Н. Хорспул);
29. Профессиональное программирование на языке си. Управление ресурсами (Касаткин А. И.);
30. Silverlight 5 с примерами на C# для профессионалов (Мэтью Мак-Дональд).

ПРИЛОЖЕНИЕ А. Эскизный проект

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

АННОТАЦИЯ

В приложении «Эскизный проект» продемонстрирован эскиз и макет программного обеспечения «ScriptGen».

Приложение состоит из нескольких разделов.

В разделе «Эскиз» представлен предварительный набросок, фиксирующий замысел художественного отображения интерфейса программы или отдельной его части.

В разделе «Макет» представлена модель программы в натуральную величину, без функциональности представляемого объекта.

ЭСКИЗ

На рисунке 1 представлен эскиз главного окна приложения, а в таблице 1 приведено описание компонентов на рисунке.

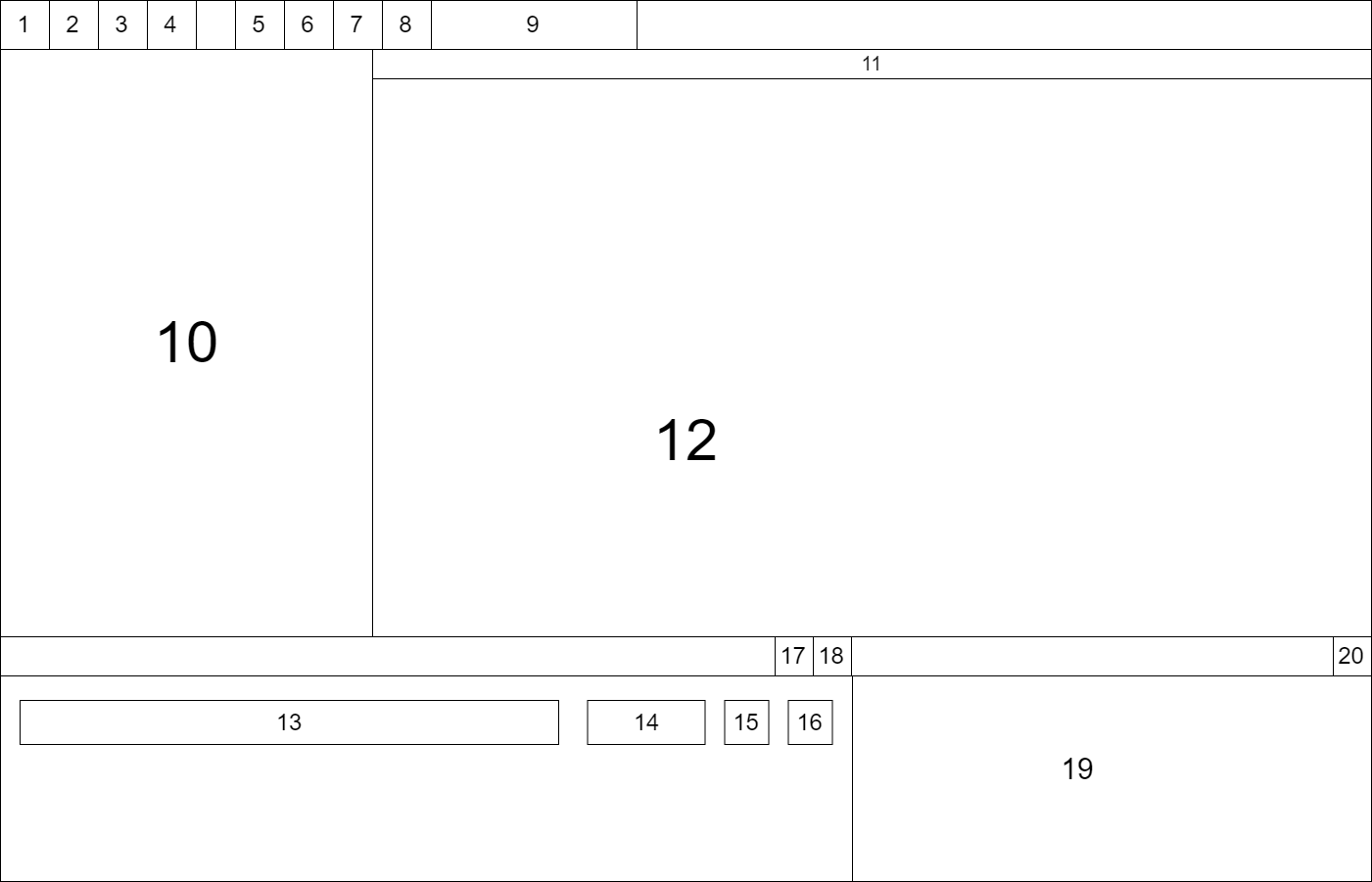


Рисунок 1 – Эскиз главного окна

Таблица 1 – Описание эскиза главного окна

| Номер | Описание |
| --- | --- |
| 1 | Кнопка для вызова окна настроек |
| 2 | Выпадающий список плагинов |
| 3 | Кнопка загрузки модели |
| 4 | Кнопка сохранения модели |
| 5 | Кнопка создания новой модели |
| 6 | Кнопка создания новой таблицы в текущей модели |
| 7 | Кнопка проверки ошибок в модели |
| 8 | Кнопка генерации скриптов для текущей модели |
| 9 | Переключение режимов модели |
| 10 | Обозреватель моделий |
| 11 | Меню вкладок с открытыми моделями |
| 12 | Поле для интерактивной работы с моделью |
| 13 | Поле ввода шаблона для генерации случайных данных |
| 14 | Поле выбора количества повторной генерации |
| 15 | Кнопка сохранения шаблона |
| 16 | Кнопка удаления шаблона |
| 17 | Кнопка запуска генерации случайных значений по шаблону |
| 18 | Кнопка добавления нового случайного значения |
| 19 | Поле вывода сообщений программы |
| 20 | Кнопка очистки истории |

На рисунке 2 представлен эскиз главного окна настроек, а в таблице 2 приведено описание компонентов на рисунке.

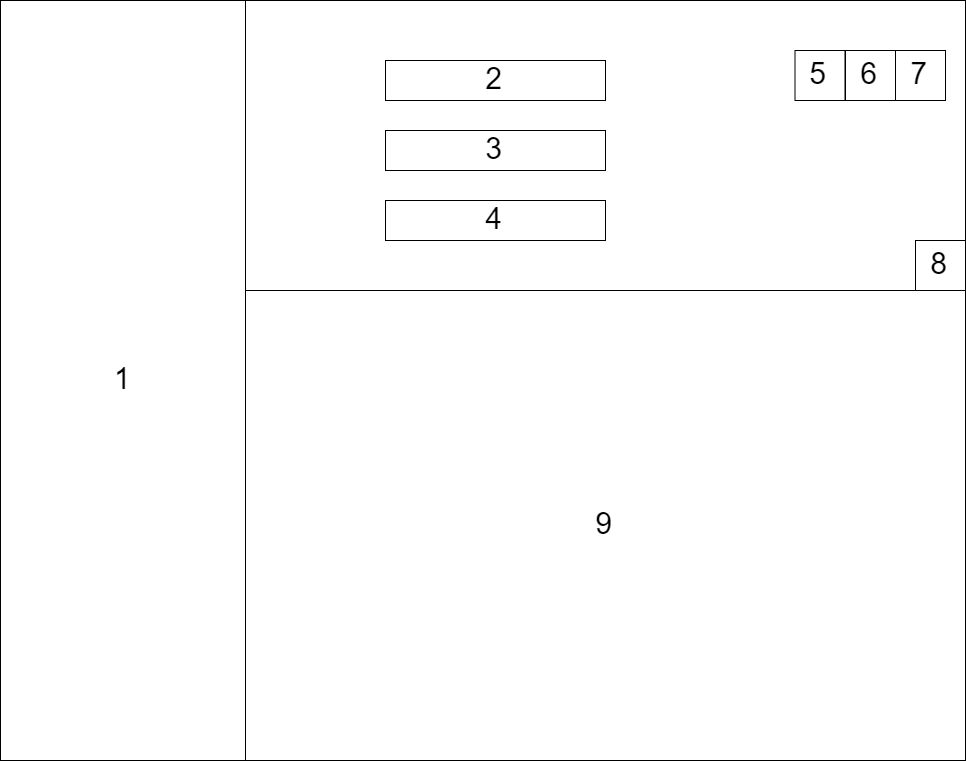


Рисунок 2 – Эскиз главного окна настроек

Таблица 2 – Описание эскиза главного окна настроек

| Номер | Описание |
| --- | --- |
| 1 | Вкладки настроек |
| 2 | Выбор экземпляра пользовательских настроек |
| 3 | Выбор основного шаблона для генерации |
| 4 | Выбор дополнительного шаблона для генерации |
| 5 | Кнопка добавления нового экземпляра настроек |
| 6 | Кнопка добавления нового шаблона |
| 7 | Кнопка открытия папки с настройками и плагинами в проводнике |
| 8 | Кнопка добавления плагинов |
| 9 | Список плагинов |

На рисунке 3 представлен эскиз окна настроек для выбора темы приложения, а в таблице 3 приведено описание компонентов на рисунке.

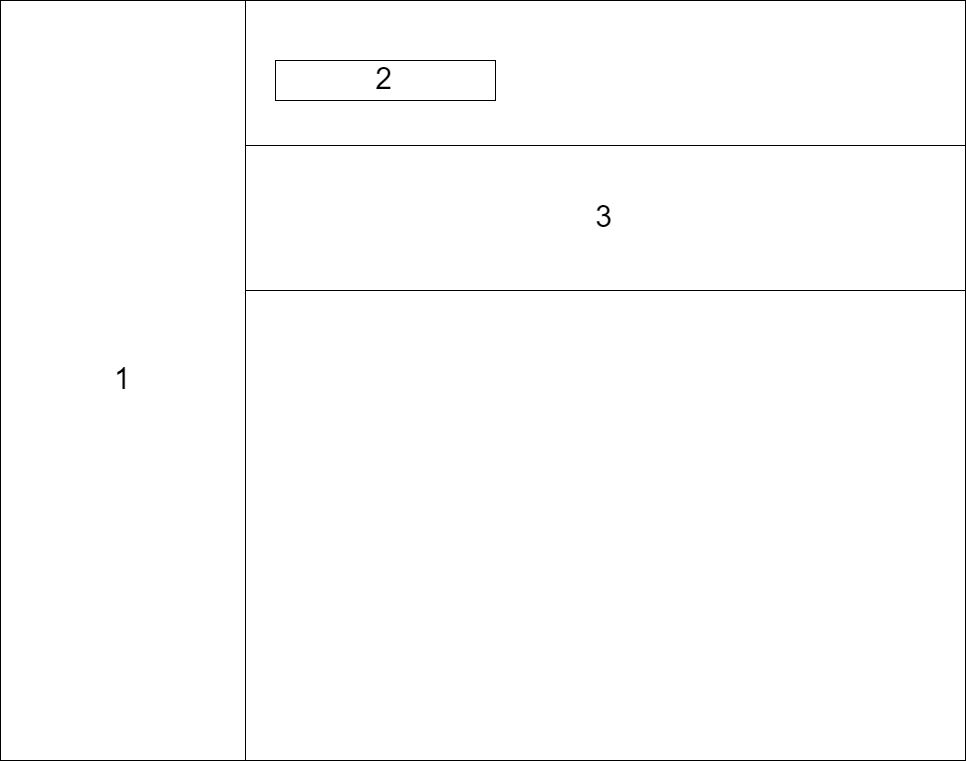


Рисунок 3 – Эскиз окна настроек для выбора темы приложения

Таблица 3 – Описание эскиза окна настроек для выбора темы приложения

| Номер | Описание |
| --- | --- |
| 1 | Вкладки настроек |
| 2 | Выбор тёмной или светлой темы |
| 3 | Выбор палитры цветов для приложения |

МАКЕТ

На рисунке 4 представлен макет главного окна приложения «ScriptGen».

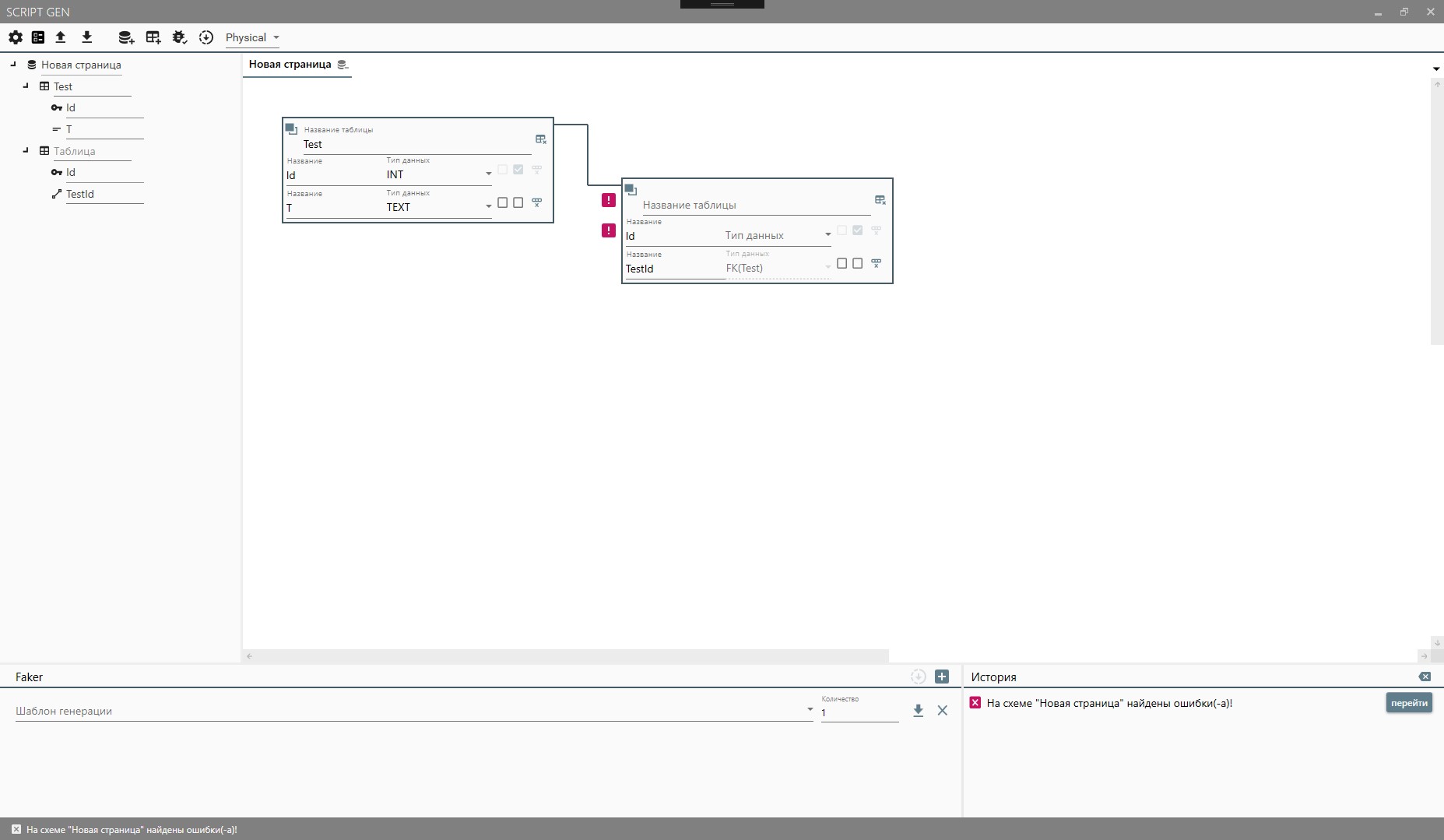


Рисунок 4 - Макет главного окна

На рисунке 5 представлен макет главного окна настроек приложения.

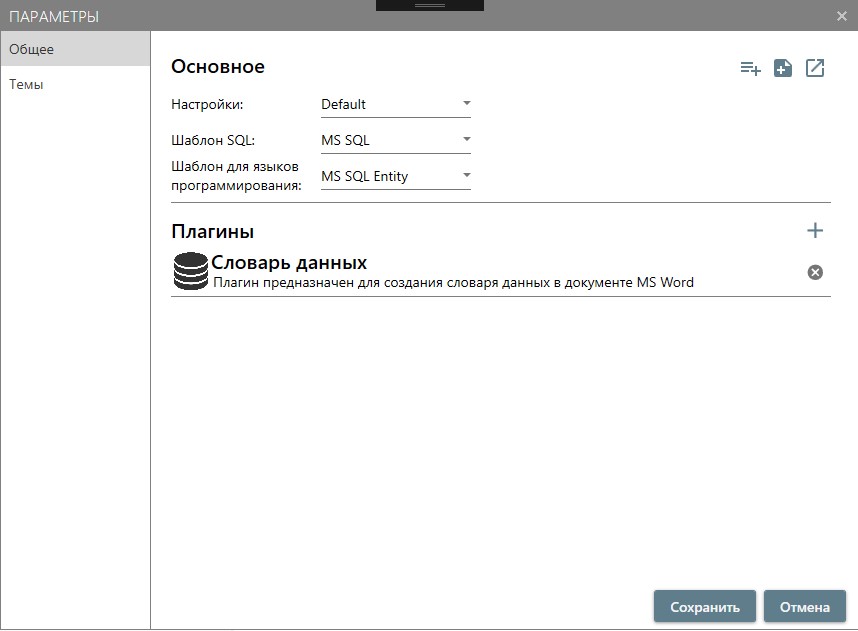


Рисунок 5 – Макет главного окна настроек

На рисунке 6 представлен макет окна настроек для выбора темы приложения.

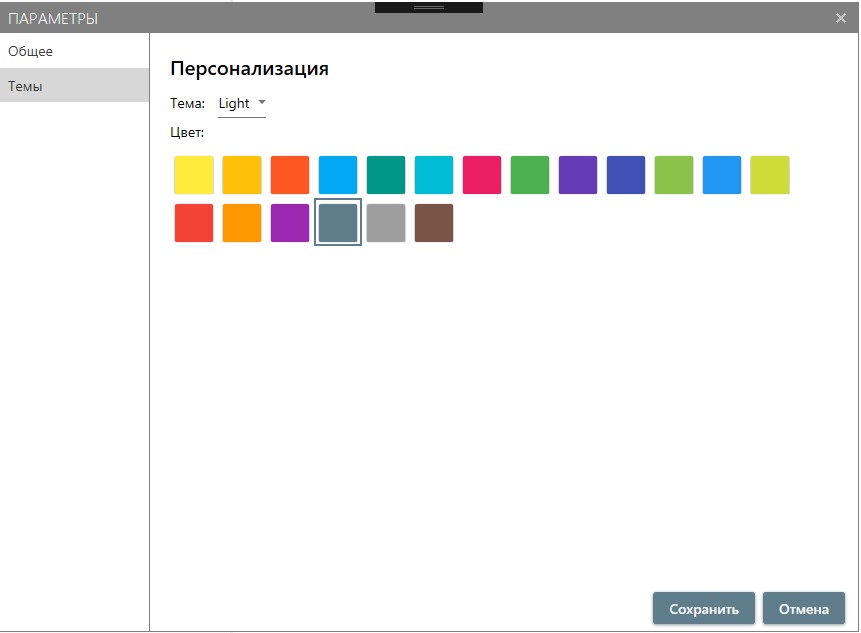


Рисунок 6 – Макет окна настроек для выбора темы приложения

ПРИЛОЖЕНИЕ Б. Сценарий тестов и результаты тестовых испытаний

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

АННОТАЦИЯ

В приложении «Сценарий тестов и результаты тестовых испытаний» продемонстрированы тестовые данные и результаты тестирования программного обеспечения «ScriptGen».

Приложение состоит из трёх разделов.

В разделе «Сценарий тестов» представлено описание последовательности тестирования системы.

В разделе «Тестовые наборы данных» приведены входные данные для проведения тестирования системы.

В разделе «Результат тестовых испытаний» представлено описание процесса тестирования и результаты тестирования по тестовым наборам из раздела 2.

СЦЕНАРИЙ ТЕСТОВ

Для тестирования формирования модели данных необходимо составить тестовую модель, после чего, используя средства отладки Visual Studio, проверить соответствие собранных данных графической модели.

Для тестирования сохранения и загрузки модели данных необходимо сохранить модель в файл, а после загрузить её в приложение. Загруженная модель не должна отличаться от оригинала.

Для тестирования скрипта, сгенерированного по модели, необходимо выполнить его в SQL Server Management Studio, построить диаграмму базы данных и сравнить её с исходной моделью.

Для тестирования загрузки плагинов и шаблонов необходимо:

1. Написать любой плагин, составить файл конфигурации и загрузить его в программу через меню настроек.
2. Составить файл с данными о шаблоне и загрузить его в программу через меню настроек.

Для тестирования работы API необходимо запустить программу и выполнить действие плагина, после чего убедится в правильности результата его работы.

Для тестирования генератора случайных данных необходимо ввести шаблон генерации и шаблоны случайных значений, после чего завершить создание случайных данных и убедится в их правильности.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

На рисунке 1 представленны данные модели для тестирования.

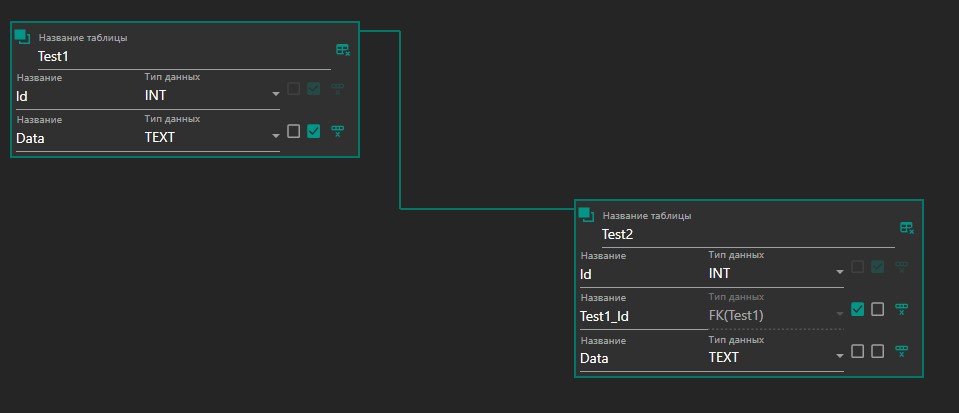


Рисунок 1 – Данные модели

На рисунке 2 представлен файл конфигурации плагина для тестирования.

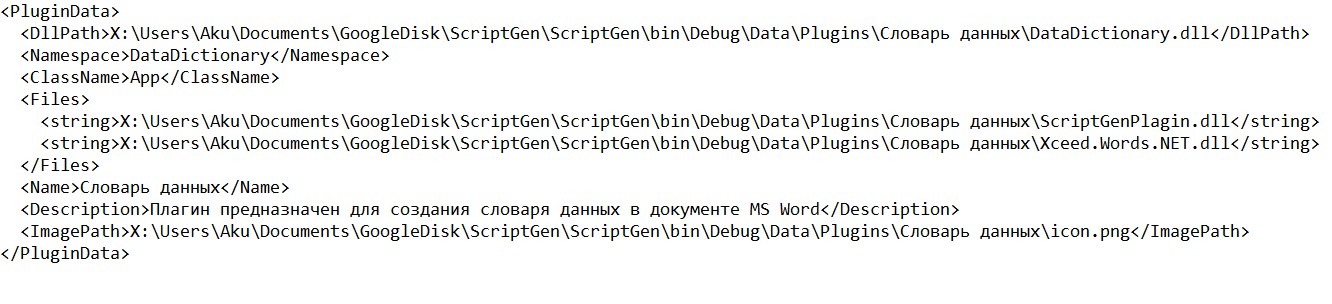


Рисунок 2 – Файл конфигурации плагина

На рисунке 3 представлен файл с данными о шаблоне.

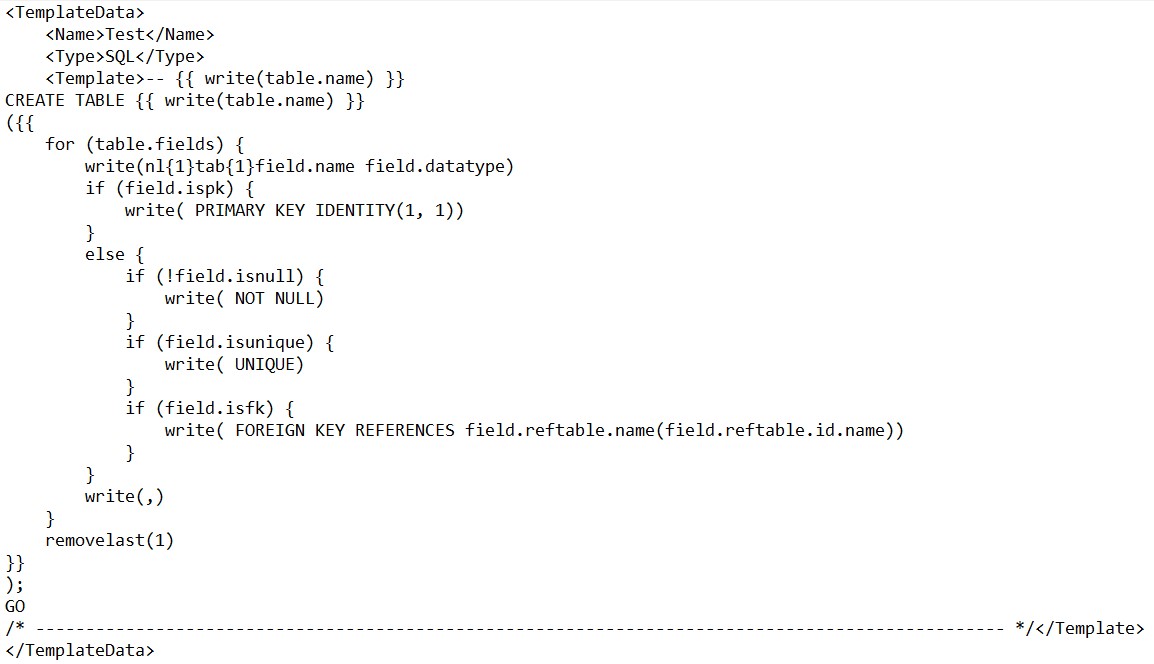


Рисунок 3 – Файл с данными о шаблоне

На рисунке 4 представлены данные для генерации случайных данных.

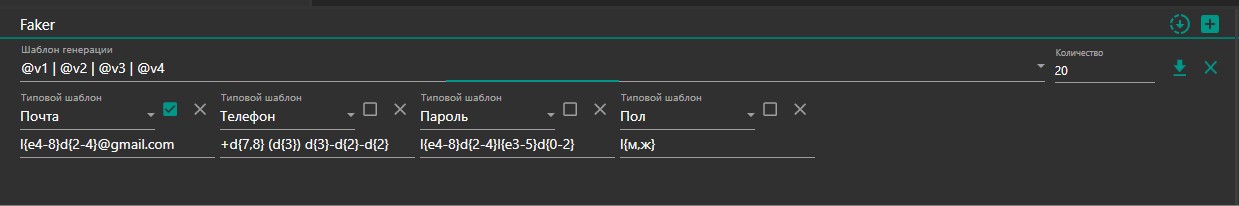


Рисунок 4 – Данные для генерации случайных данных

РЕЗУЛЬТАТ ТЕСТОВЫХ ИСПЫТАНИЙ

На рисунке 5 представлен результат тестирования формирования модели данных. Содержимое переменой соответствует введённым данным модели.

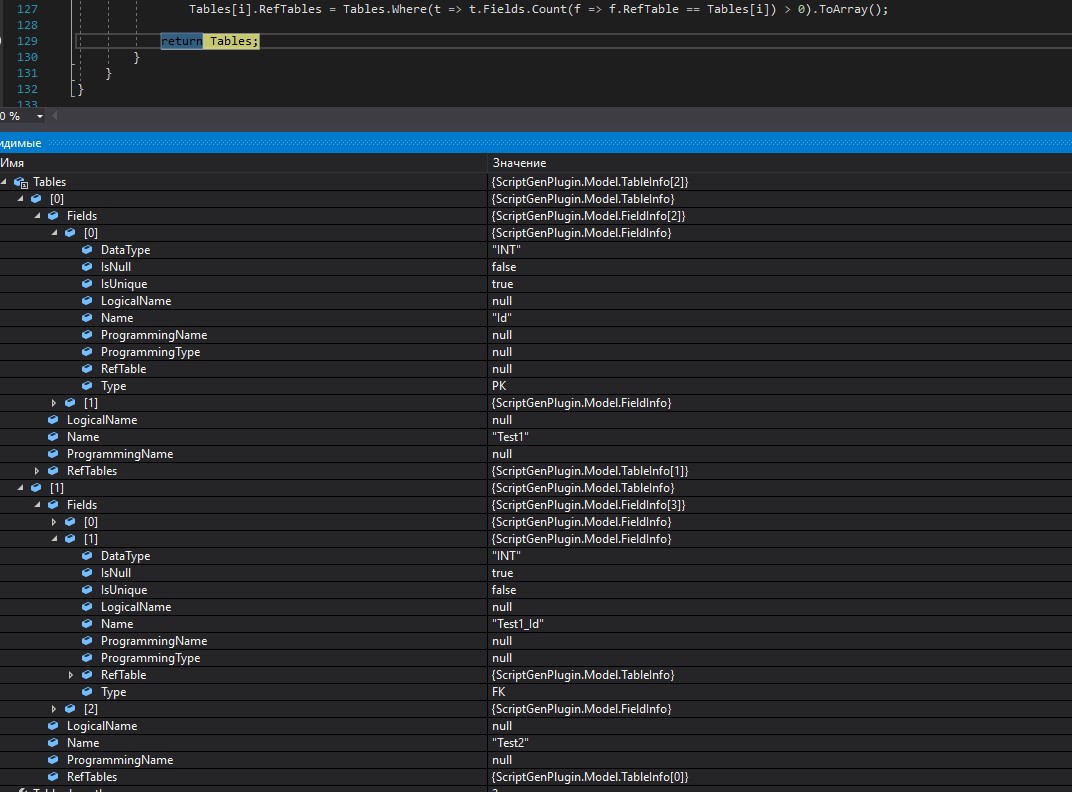


Рисунок 5 – Результат тестирования формирования модели данных

При сохранении и загрузки модели ошибок не возникло. После загрузки модель соответствовала своему изначальному виду.

После генерации скрипта базы данных и загрузки его в SQL Server Management Studio, диаграмма базы данных соответствовала модели для тестирования. Диаграмма базы данных представлена на рисунке 6.

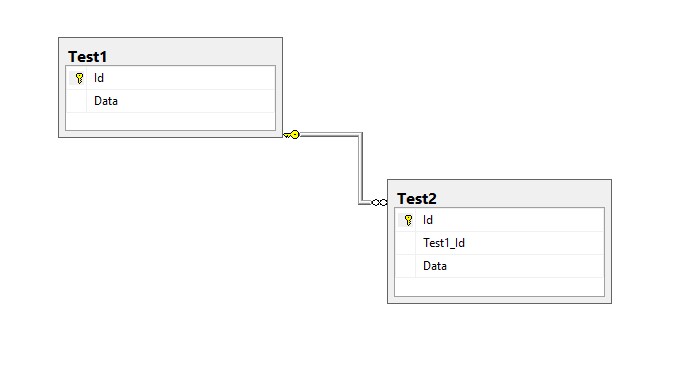


Рисунок 6 - Диаграмма базы данных

После выбора файла конфигурации плагина в меню настроек приложения ошибок не произошло, плагин успешно загрузился и отобразился в списке установленных плагинов, показанном на рисунке 7.

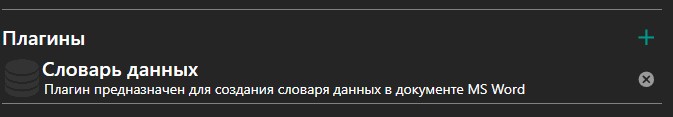


Рисунок 7 – Список плагинов

После выбора файла с данными о шаблоне ошибок не произошло, название плагина отобразилось в списке доступных для генерации шаблонов, показанном на рисунке 8.

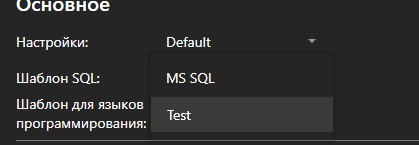


Рисунок 8 – Список шаблонов

При выполнении действия установленного плагина ошибок не обнаружено, программа продолжила стабильную работу.

После генерации случайных данных по шаблону был получен верный результат, представленный на рисунке 9.

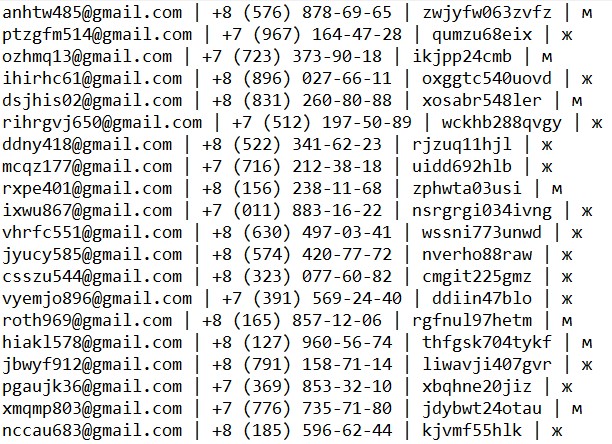


Рисунок 9 – Результат генерации случайных данных

ПРИЛОЖЕНИЕ В. Исходный текст программы

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

АННОТАЦИЯ

В приложении «Исходный текст программы» продемонстрирован код всех модулей приложения «ScriptGen».

Приложение состоит из нескольких разделов.

В разделе «Модули программы» представлена таблица всех модулей системы с описание и размером каждого модуля.

В разделе «Текст программы» представлен код каждого модуля системы.

МОДУЛИ ПРОГРАММЫ

В таблице 1 приведён список модулей программного обеспечения с их описанием и указанием степени авторства.

Таблица 1 – Модули программы

| Номер | Наименование модуля программы | Описание | Степень авторства |
| --- | --- | --- | --- |
| 1 | App.xaml | Точка входа в программу | Авторский |
| 2 | ElementBase.cs | Базовый класс графического элемента приложения | Авторский |
| 3 | FakeWordBase.cs | Базовый класс для генерации случайных слов | Авторский |
| 4 | FieldBase.cs | Базовый класс поля таблицы | Авторский |
| 5 | LineBase.cs | Базовый класс линия связи между таблицами | Авторский |
| 6 | PageBase.cs | Базовый класс страницы приложения | Авторский |
| 7 | TableBase.cs | Базовый класс таблицы | Авторский |
| 8 | UniqueErrorBase.cs | Базовый класс для проверки на уникальность | Авторский |
| 9 | API.cs | API приложения | Авторский |
| 10 | MessageAPI.cs | Часть API для сообщений | Авторский |
| 11 | PluginData.cs | Модель загружаемого плагина | Авторский |
| 12 | PluginManager.cs | Позволяет осуществлять работу с плагинами | Авторский |
| 13 | BoolToVisibleConverter.cs | Конвертер значений для Binding | Авторский |
| 14 | DisabledFKConverter.cs | Конвертер значений для Binding | Авторский |
| 15 | DisabledPKConverter.cs | Конвертер значений для Binding | Авторский |
| 16 | FieldTypeToIconConverter.cs | Конвертер значений для Binding | Авторский |
| 17 | InversionBool.cs | Конвертер значений для Binding | Авторский |
| 18 | LogicalVisibleConverter.cs | Конвертер значений для Binding | Авторский |
| 19 | MessageOpenConverter.cs | Конвертер значений для Binding | Авторский |
| 20 | ModelTypeToIntConverter.cs | Конвертер значений для Binding | Авторский |
| 21 | PhysicalVisibleConverter.cs | Конвертер значений для Binding | Авторский |
| 22 | ProgrammingVisibleConverter.cs | Конвертер значений для Binding | Авторский |
| 23 | SwitchMessageColorConverter.cs | Конвертер значений для Binding | Авторский |
| 24 | ReadonlyBinding.cs | Конвертер значений для Binding | Авторский |
| 25 | ScrollViewerBinding.cs | Конвертер значений для Binding | Авторский |
| 26 | DataBaseFileWork.cs | Осуществляет работу с моделями базы данных | Авторский |
| 27 | DataConverter.cs | Осуществляет преобразование данных | Авторский |
| 28 | Enums.cs | Перечисления | Авторский |
| 29 | FileWork.cs | Осуществляет работу с файлами | Авторский |
| 30 | Info.cs | Содержит общую информацию приложения | Авторский |
| 31 | PluginFileWork.cs | Работа с файлами для плагинов | Авторский |
| 32 | SettingsFileWork.cs | Работа с файлами для настроек | Авторский |
| 33 | Verification.cs | Проверка строк | Авторский |
| 34 | DataBaseSave.cs | Модель сохранения данных о базе данных | Авторский |
| 35 | FieldSave.cs | Модель сохранения поля таблицы | Авторский |
| 36 | LineSave.cs | Модель сохранения линии связи | Авторский |
| 37 | TableSave.cs | Модель сохранения таблицы | Авторский |
| 38 | FakeHelper.cs | Осуществляет выделение параметров шаблона | Авторский |
| 39 | FakeChar.cs | Генерирует случайные символы | Авторский |
| 40 | FakeName.cs | Генерирует случайные имена | Авторский |
| 41 | FakeNumber.cs | Генерирует случайные цифры | Авторский |
| 42 | FakeSurname.cs | Генерирует случайные фамилии | Авторский |
| 43 | FakeWord.cs | Генерирует случайные слова | Авторский |
| 44 | FakerM.cs | Модель генерации случайного значения | Авторский |
| 45 | FakerV.xaml | Представление генератора случайных значений | Авторский |
| 46 | FakerVM.cs | Осуществляет генерации записей по шаблону с случайными значениями | Авторский |
| 47 | FillHelper.cs | Осуществляет выделение ключевых слов и параметров из текста | Авторский |
| 48 | ScriptGenerator.cs | Осуществляет генерацию по шаблону | Авторский |
| 49 | TemplateFiller.cs | Осуществляет нормализацию шаблона | Авторский |
| 50 | For.cs | Ключевое слово цикла | Авторский |
| 51 | If.cs | Ключевое слово условия | Авторский |
| 52 | Once.cs | Ключевое слово блока кода для выполнения один раз за генерацию | Авторский |
| 53 | Rec.cs | Ключевое слово рекурсии | Авторский |
| 54 | RemoveLast.cs | Ключевое слово удаления последних символов | Авторский |
| 55 | Replace.cs | Ключевое слово замены слов | Авторский |
| 56 | Set.cs | Ключевое слово запоминания значения | Авторский |
| 57 | Write.cs | Ключевое слово вывода информации | Авторский |
| 58 | IElement.cs | Графический элемент приложения | Авторский |
| 59 | IError.cs | Управление ошибкой элемента | Авторский |
| 60 | IFaker.cs | Интерфейс для генератора случайных значений | Авторский |
| 61 | IField.cs | Поле таблицы | Авторский |
| 62 | IKeyword.cs | Ключевого слово для выстроенного языка шаблонизации | Авторский |
| 63 | ILine.cs | Линия связи между таблицами | Авторский |
| 64 | IMove.cs | Движение элемента | Авторский |
| 65 | IPage.cs | Страница приложения | Авторский |
| 66 | ITable.cs | Таблица | Авторский |
| 67 | DataBaseM.cs | Модель базы данных | Авторский |
| 68 | ElementM.cs | Модель элемента | Авторский |
| 69 | FieldM.cs | Модель поля таблицы | Авторский |
| 70 | MessageM.cs | Модель сообщения | Авторский |
| 71 | ScriptGenM.cs | Модель приложения | Авторский |
| 72 | TableM.cs | Модель таблицы | Авторский |
| 73 | AssemblyInfo.cs | Системная информация | Авторский |
| 74 | Resources.Designer.cs | Класс ресурса со строгой типизацией для поиска локализованных строк и т.д. | Авторский |
| 75 | Settings.Designer.cs | Настройки проекта | Авторский |
| 76 | SettingsDialog.xaml | Настройки проекта | Авторский |
| 77 | SettingsManager.cs | Осуществляет работу с данными настроек | Авторский |
| 78 | ColorM.cs | Модель цвета | Авторский |
| 79 | CommonM.cs | Модель настроек приложения | Авторский |
| 80 | Option.cs | Модель экземпляра настроек | Авторский |
| 81 | OptionData.cs | Модель настроек для загрузки | Авторский |
| 82 | SettingsSave.cs | Модель для сохранения настроек | Авторский |
| 83 | TemplateData.cs | Модель для загрузки шаблонов | Авторский |
| 84 | ThemeM.cs | Модель темы | Авторский |
| 85 | CommonV.xaml | Представление основных настроек | Авторский |
| 86 | SettingsV.xaml | Представление настроек | Авторский |
| 87 | ThemeV.xaml | Представление настроек темы | Авторский |
| 88 | CommonVM.cs | Управляет основными настройками приложения | Авторский |
| 89 | SettingsVM.cs | Управляет настройками | Авторский |
| 90 | ThemeVM.cs | Управляет темами приложения | Авторский |
| 91 | FieldV.xaml | Представление поля | Авторский |
| 92 | LineV.xaml | Представление линии | Авторский |
| 93 | PageV.xaml | Представление страницы | Авторский |
| 94 | ScriptGenV.xaml | Представление приложения | Авторский |
| 95 | TableV.xaml | Представление таблицы | Авторский |
| 96 | FieldVM.cs | Работа с полями таблицы | Авторский |
| 97 | ForeignFieldLineVM.cs | Работа с соединительной линией | Авторский |
| 98 | ForeignFieldVM.cs | Работа с внешним ключом | Авторский |
| 99 | PageVM.cs | Работа с страницей приложения | Авторский |
| 100 | PrimaryFieldVM.cs | Работа с первичным ключом | Авторский |
| 101 | ScriptGenVM.cs | Работа с приложение | Авторский |
| 102 | StatusBarManagerVM.cs | Осуществляет вывод сообщений | Авторский |
| 103 | TableVM.cs | Работа с таблицей | Авторский |

ТЕКСТ ПРОГРАММЫ

1. App.xaml. Точка входа в программу.

<Application

x:Class="ScriptGen.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:conv="clr-namespace:ScriptGen.BindingConverter"

Startup="ScriptGenStart"

ShutdownMode="OnExplicitShutdown">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<!-- MahApps -->

<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />

<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />

<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />

<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseDark.xaml" />

<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Teal.xaml" />

<!-- Material Design -->

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Dark.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/MaterialDesignColor.Teal.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/MaterialDesignColor.Teal.xaml" />

</ResourceDictionary.MergedDictionaries>

<conv:BoolToVisibleConverter

x:Key="BoolVisible"/>

<conv:InversionBool

x:Key="InvBool"/>

<conv:PhysicalVisibleConverter

x:Key="PhysicalVisible"/>

<conv:LogicalVisibleConverter

x:Key="LogicalVisible"/>

<conv:ProgrammingVisibleConverter

x:Key="ProgrammingVisible"/>

<conv:SwitchMessageColorConverter

x:Key="SwitchMessageColor"/>

<conv:MessageOpenConverter

x:Key="MessageOpen"/>

<conv:DisabledPKConverter

x:Key="DisabledPK"/>

<conv:DisabledFKConverter

x:Key="DisabledFK"/>

<conv:FieldTypeToIconConverter

x:Key="Icon"/>

<conv:ModelTypeToIntConverter

x:Key="ModelTypeToInt"/>

</ResourceDictionary>

</Application.Resources>

</Application>

2. ElementBase.cs. Базовый класс графического элемента приложения.

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using ScriptGen.Common;

using ScriptGen.Interface;

using ScriptGen.Model;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс графического элемента приложения

/// </summary>

public abstract class ElementBase : ViewModelBase, IElement

{

/// <summary>

/// Модель элемента

/// </summary>

protected ElementM element = new ElementM();

/// <summary>

/// Тип модели (физическая/логическая/программирование)

/// </summary>

private ModelType modelType;

/// <summary>

/// Определяет элемент с фокусом

/// </summary>

private Focuses focusedElement;

/// <summary>

/// Создаёт объект базового элемента

/// </summary>

public ElementBase()

=> LostFocus = new SimpleCommand(() => FocusedElement = Focuses.None);

/// <summary>

/// Команда нажатия "Enter" в поле ввода для названия

/// </summary>

public SimpleCommand NameEnter { get; set; }

/// <summary>

/// Команда нажатия "Up" в поле ввода для названия

/// </summary>

public SimpleCommand NameUp { get; set; }

/// <summary>

/// Команда нажатия "Down" в поле ввода для названия

/// </summary>

public SimpleCommand NameDown { get; set; }

/// <summary>

/// Команда потери элементом фокуса

/// </summary>

public SimpleCommand LostFocus { get; set; }

/// <summary>

/// Команда загрузки элемента

/// </summary>

public SimpleCommand Loaded { get; set; }

/// <summary>

/// Команда удаления элемента

/// </summary>

public SimpleCommand RemoveElement { get; set; }

/// <summary>

/// Название

/// </summary>

public virtual string Name

{

get => element.Name;

set

{

element.Name = value;

OnPropertyChanged();

}

}

/// <summary>

/// Логическое название

/// </summary>

public virtual string LogicalName

{

get => element.LogicalName;

set

{

element.LogicalName = value;

OnPropertyChanged();

}

}

/// <summary>

/// Название модели для языков программирования

/// </summary>

public virtual string ProgrammingName

{

get => element.ProgrammingName;

set

{

element.ProgrammingName = value;

OnPropertyChanged();

}

}

/// <summary>

/// true/false существует ли ошибка у элемента

/// </summary>

public virtual bool IsError

{

get => element.IsError;

set

{

element.IsError = value;

OnPropertyChanged();

}

}

/// <summary>

/// Сообщение об ошибке

/// </summary>

public virtual string Message

{

get => element.Message;

set

{

element.Message = value;

OnPropertyChanged();

}

}

/// <summary>

/// Тип модели (физическая/логическая/программирование)

/// </summary>

public virtual ModelType TypeModel

{

get => modelType;

set

{

modelType = value;

OnPropertyChanged();

}

}

/// <summary>

/// Определяет элемент с фокусом

/// </summary>

public virtual Focuses FocusedElement

{

get => focusedElement;

set

{

focusedElement = value;

OnPropertyChanged();

}

}

/// <summary>

/// Устанавливает фокус элемента в зависимости от его свойств

/// </summary>

public virtual void SetFocus()

{

switch (TypeModel)

{

case ModelType.Physical:

FocusedElement = Focuses.Name;

break;

case ModelType.Logical:

FocusedElement = Focuses.LogicalName;

break;

case ModelType.Programming:

FocusedElement = Focuses.ProgrammingName;

break;

}

}

/// <summary>

/// Устанавливает ошибку с переданным сообщением

/// </summary>

/// <param name="message">Сообщение</param>

public virtual void SetError(string message)

{

IsError = message != null;

Message = message;

}

}

}

3. FakeWordBase.cs. Базовый класс для генерации случайных слов.

using System;

using ScriptGen.Faker;

using ScriptGen.Interface;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс для генерации случайных слов

/// </summary>

public abstract class FakeWordBase : IFaker

{

/// <summary>

/// Возвращает count слов для указанного языка

/// </summary>

/// <param name="count">Количество слов</param>

/// <param name="language">Язык</param>

/// <returns>Случайные слов</returns>

public string Next(int count, char language)

{

string result = "";

string[] words = GetWords(language);

Random random = FakeHelper.GetRandom();

for (int i = 0; i < count; i++)

result += $" {words[random.Next(0, words.Length)]}";

return result.Substring(1);

}

/// <summary>

/// Возвращает от minCount до maxCount слов для указанного языка

/// </summary>

/// <param name="minCount">Минимальное количество слов</param>

/// <param name="maxCount">Максимальное количество слов</param>

/// <param name="language">Язык</param>

/// <returns>Случайные слов</returns>

public string Next(int minCount, int maxCount, char language)

{

string result = "";

string[] words = GetWords(language);

Random random = FakeHelper.GetRandom();

for (int i = 0; i < random.Next(minCount, maxCount); i++)

result += $" {words[random.Next(0, words.Length)]}";

return result.Length > 1

? result.Substring(1)

: result;

}

/// <summary>

/// Возвращает случайное слова из представленных

/// </summary>

/// <param name="values">Возможные значения</param>

/// <returns>Случайное слово</returns>

public string Next(string[] values)

{

Random random = FakeHelper.GetRandom();

return values[random.Next(0, values.Length)];

}

/// <summary>

/// Возвращает словарь слов для выборки

/// </summary>

/// <param name="language">Язык</param>

/// <returns>Словарь для выборки</returns>

protected virtual string[] GetWords(char language)

=> new string[0];

}

}

4. FieldBase.cs. Базовый класс поля таблицы.

using System;

using BaseMVVM.Command;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGen.Model;

using ScriptGenPlugin.Model;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс поля таблицы

/// </summary>

public abstract class FieldBase : ElementBase, IField

{

/// <summary>

/// Событие удаления поля

/// </summary>

public event Action<IField> OnRemoveField;

/// <summary>

/// Модель поля

/// </summary>

private FieldM field;

/// <summary>

/// Создаёт объект базового поля

/// </summary>

public FieldBase()

{

field = new FieldM

{

Element = element,

RefTable = null

};

NameEnter = new SimpleCommand(() =>

{

if (TypeModel == ModelType.Physical)

FocusedElement = Focuses.DataType;

else if (TypeModel == ModelType.Programming)

FocusedElement = Focuses.ProgrammingType;

else

NextField?.Invoke();

});

NameDown = DataTypeEnter = ProgrammingTypeEnter = new SimpleCommand(() => NextField?.Invoke());

NameUp = new SimpleCommand(() => PrevField?.Invoke());

RemoveElement = new SimpleCommand(() => OnRemoveField?.Invoke(this));

}

/// <summary>

/// Вызывается при действиях, осуществляющих переключение на следующее поле

/// </summary>

public Action NextField { get; set; }

/// <summary>

/// Вызывается при действиях, осуществляющих переключение на предыдущее поле

/// </summary>

public Action PrevField { get; set; }

/// <summary>

/// Команда нажатия "Enter" в поле ввода для SQL типа данных

/// </summary>

public SimpleCommand DataTypeEnter { get; }

/// <summary>

/// Команда нажатия "Enter" в поле ввода для типа данных для языков программирования

/// </summary>

public SimpleCommand ProgrammingTypeEnter { get; }

/// <summary>

/// Тип поля

/// </summary>

public virtual FieldType Type

{

get => field.Type;

set

{

field.Type = value;

OnPropertyChanged();

}

}

/// <summary>

/// SQL тип данных поля

/// </summary>

public virtual string DataType

{

get => field.DataType;

set

{

field.DataType = value;

OnPropertyChanged();

}

}

/// <summary>

/// Тип данных поля для языков программирования

/// </summary>

public virtual string ProgrammingType

{

get => field.ProgrammingType;

set

{

field.ProgrammingType = value;

OnPropertyChanged();

}

}

/// <summary>

/// Список SQL типов данных

/// </summary>

public virtual string[] DataTypeSource

=> null;

/// <summary>

/// Список типов данных для языков программирования

/// </summary>

public virtual string[] ProgrammingTypeSource

=> null;

/// <summary>

/// true/false может ли поле быть равно null

/// </summary>

public virtual bool IsNull

{

get => field.IsNull;

set

{

field.IsNull = value;

OnPropertyChanged();

}

}

/// <summary>

/// true/false является ли поле уникальным

/// </summary>

public virtual bool IsUnique

{

get => field.IsUnique;

set

{

field.IsUnique = value;

OnPropertyChanged();

}

}

/// <summary>

/// Информация о таблице на которую ссылается поле (только для FK)

/// </summary>

public virtual ITable RefTable

{

get => field.RefTable;

set => field.RefTable = value;

}

/// <summary>

/// Загружает данные поля

/// </summary>

/// <param name="fieldSave">Сохранённая информация</param>

public virtual void Load(FieldSave fieldSave)

{

Name = fieldSave.Name;

LogicalName = fieldSave.LogicalName;

ProgrammingName = fieldSave.ProgrammingName;

IsNull = fieldSave.IsNull;

IsUnique = fieldSave.IsUnique;

}

/// <summary>

/// Устанавливает ошибку с переданным сообщением

/// </summary>

/// <param name="message">Сообщение</param>

public override void SetError(string message)

{

FocusedElement = Focuses.None;

base.SetError(Verification.ConcatString(message, field.GetError()));

}

}

}

5. LineBase.cs. Базовый класс линия связи между таблицами.

using System;

using BaseMVVM.Abstraction;

using ScriptGen.Interface;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс линия связи между таблицами

/// </summary>

public abstract class LineBase : ViewModelBase, ILine

{

/// <summary>

/// Вызывается при удалении линии

/// </summary>

public event Action<ILine> OnRemoveLine;

/// <summary>

/// Координата X для начальной точки линии

/// </summary>

private double sourceX;

/// <summary>

/// Координата X для конечной точки линии

/// </summary>

private double targetX;

/// <summary>

/// Координата Y для начальной точки линии

/// </summary>

private double sourceY;

/// <summary>

/// Координата Y для конечной точки линии

/// </summary>

private double targetY;

/// <summary>

/// Координата X для разделительной линии

/// </summary>

private double connectionX;

/// <summary>

/// Создаёт объект базовой линии связи

/// </summary>

/// <param name="source">Исходная таблица</param>

/// <param name="target">Конечная таблица</param>

public LineBase(ITable source, ITable target)

{

Source = source;

Target = target;

}

/// <summary>

/// Координата X для начальной точки линии

/// </summary>

public virtual double SourceX

{

get => sourceX;

set

{

sourceX = value;

OnPropertyChanged();

}

}

/// <summary>

/// Координата X для конечной точки линии

/// </summary>

public virtual double TargetX

{

get => targetX;

set

{

targetX = value;

OnPropertyChanged();

}

}

/// <summary>

/// Координата Y для начальной точки линии

/// </summary>

public virtual double SourceY

{

get => sourceY;

set

{

sourceY = value;

OnPropertyChanged();

}

}

/// <summary>

/// Координата Y для конечной точки линии

/// </summary>

public virtual double TargetY

{

get => targetY;

set

{

targetY = value;

OnPropertyChanged();

}

}

/// <summary>

/// Координата X для разделительной линии

/// </summary>

public virtual double ConnectionX

{

get => connectionX;

set

{

connectionX = value;

OnPropertyChanged();

}

}

/// <summary>

/// Таблица, являющаяся источником

/// </summary>

public ITable Source { get; }

/// <summary>

/// Таблица, являющаяся целью

/// </summary>

public ITable Target { get; }

/// <summary>

/// Поле FK, с которым связанна линия

/// </summary>

public virtual IField Field { get; }

/// <summary>

/// Вызывается для удаления линии связи

/// </summary>

/// <param name="line">Линия связи</param>

protected void RemoveLine(ILine line)

=> OnRemoveLine?.Invoke(line);

}

}

6. PageBase.cs. Базовый класс страницы приложения.

using System;

using System.Collections.ObjectModel;

using System.Linq;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGen.Model;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс страницы приложения

/// </summary>

public abstract class PageBase : ViewModelBase, IPage

{

/// <summary>

/// Вызывается при удалении страницы

/// </summary>

public event Action<IPage> OnRemovePage;

/// <summary>

/// Вызывается при выделении страницы

/// </summary>

public event Action<IPage> OnSelectPage;

/// <summary>

/// Модель базы данных

/// </summary>

private DataBaseM dataBase = new DataBaseM();

/// <summary>

/// true/false выбрана ли страница

/// </summary>

private bool isSelect;

/// <summary>

/// Выделенная таблица

/// </summary>

private ITable selectTable;

/// <summary>

/// Тип модели

/// </summary>

private ModelType modelType;

/// <summary>

/// Смещения по вертикали

/// </summary>

public double scrollVertical;

// <summary>

/// Смещения по горизонтали

/// </summary>

public double scrollHorizontal;

/// <summary>

/// Ширина

/// </summary>

private double actualWidth;

/// <summary>

/// Высота

/// </summary>

private double actualHeight;

/// <summary>

/// Создаёт базовый объект страницы приложения

/// </summary>

public PageBase()

{

RemovePage = new SimpleCommand(() => OnRemovePage?.Invoke(this));

SelectPage = new SimpleCommand(() => OnSelectPage?.Invoke(this));

}

/// <summary>

/// Команда удаления страницы

/// </summary>

public SimpleCommand RemovePage { get; }

/// <summary>

/// Команда выделения (выбора) страницы

/// </summary>

public SimpleCommand SelectPage { get; }

/// <summary>

/// Таблицы

/// </summary>

public ObservableCollection<ITable> Tables

=> dataBase.Tables;

/// <summary>

/// Линии

/// </summary>

public ObservableCollection<ILine> Lines

=> dataBase.Lines;

/// <summary>

/// Название

/// </summary>

public virtual string Name

{

get => dataBase.Name;

set

{

dataBase.Name = value;

OnPropertyChanged();

}

}

/// <summary>

/// true/false выделена ли страница

/// </summary>

public virtual bool IsSelect

{

get => isSelect;

set

{

isSelect = value;

OnPropertyChanged();

}

}

/// <summary>

/// Выделенная таблица

/// </summary>

public virtual ITable SelectTable

{

get => selectTable;

set

{

selectTable = value;

OnPropertyChanged();

}

}

/// <summary>

/// Тип модели

/// </summary>

public virtual ModelType TypeModel

{

get => modelType;

set

{

modelType = value;

OnPropertyChanged();

}

}

/// <summary>

/// Смещения по вертикали

/// </summary>

public virtual double ScrollVertical

{

get => scrollVertical;

set

{

scrollVertical = value;

OnPropertyChanged();

}

}

/// <summary>

/// Смещения по горизонтали

/// </summary>

public virtual double ScrollHorizontal

{

get => scrollHorizontal;

set

{

scrollHorizontal = value;

OnPropertyChanged();

}

}

/// <summary>

/// Ширина

/// </summary>

public virtual double ActualWidth

{

get => actualWidth;

set => actualWidth = value;

}

/// <summary>

/// Высота

/// </summary>

public virtual double ActualHeight

{

get => actualHeight;

set => actualHeight = value;

}

/// <summary>

/// true/false существует ли ошибка у элемента

/// </summary>

public virtual bool IsError

{

get => dataBase.IsError;

set

{

dataBase.IsError = value;

OnPropertyChanged();

}

}

// <summary>

/// Сообщение об ошибке

/// </summary>

public virtual string Message { get; set; } = null;

/// <summary>

/// Устанавливает ошибку с переданным сообщением

/// </summary>

/// <param name="message">Сообщение</param>

public virtual void SetError(string message = null)

{

dataBase.SetTableError();

IsError = !Tables.All(t => !t.IsError && t.Fields.All(f => !f.IsError));

}

/// <summary>

/// Добавляет таблицу

/// </summary>

/// <param name="table">Таблица</param>

public virtual void AddTable(ITable table)

{

table.OnRemoveTable += RemoveTable;

table.OnSelectTable += (t) => SelectTable = t;

Tables.Add(table);

}

/// <summary>

/// Удаляет таблицу

/// </summary>

/// <param name="table">Таблица</param>

public virtual void RemoveTable(ITable table)

=> Tables.Remove(table);

/// <summary>

/// Загружает данные страницы

/// </summary>

/// <param name="dataBaseSave">Сохранённая информация</param>

public virtual void Load(DataBaseSave dataBaseSave)

=> Name = dataBaseSave.Name;

}

}

7. TableBase.cs. Базовый класс таблицы.

using System;

using System.Collections.ObjectModel;

using System.Windows;

using BaseMVVM.Command;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGen.Model;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс таблицы

/// </summary>

public abstract class TableBase : ElementBase, ITable

{

/// <summary>

/// Происходит при удалении таблицы

/// </summary>

public event Action<ITable> OnRemoveTable;

/// <summary>

/// Происходит при выделении таблицы

/// </summary>

public event Action<ITable> OnSelectTable;

/// <summary>

/// Событие изменения ширины элемента

/// </summary>

public event Action<double, IMove> OnWidthChanged;

/// <summary>

/// Событие изменения высоты элемента

/// </summary>

public event Action<double, IMove> OnHeightChanged;

/// <summary>

/// Событие изменения положения элемента

/// </summary>

public event Action<Thickness, IMove> OnMarginChanged;

/// <summary>

/// Модель таблицы

/// </summary>

private TableM table;

/// <summary>

/// true/false выделена ли таблица

/// </summary>

private bool isSelect;

/// <summary>

/// Создаёт базовый объект таблицы

/// </summary>

public TableBase()

{

table = new TableM()

{

Element = element,

Width = 350,

Height = 50

};

RemoveElement = new SimpleCommand(() => OnRemoveTable?.Invoke(this));

SelectElement = new SimpleCommand(() => OnSelectTable?.Invoke(this));

}

/// <summary>

/// Команда выделения таблицы

/// </summary>

public SimpleCommand SelectElement { get; set; }

/// <summary>

/// true/false можно ли изменить ширину

/// </summary>

public Func<double, IMove, bool> CanWidthChange { get; set; }

/// <summary>

/// true/false можно ли изменить высоту

/// </summary>

public Func<double, IMove, bool> CanHeightChange { get; set; }

/// <summary>

/// Функция, определяющая можно ли изменить положение элемента по горизонтали

/// </summary>

public Func<double, IMove, bool> CanHorizontalChange { get; set; }

/// <summary>

/// Функция, определяющая можно ли изменить положение элемента по вертикали

/// </summary>

public Func<double, IMove, bool> CanVerticalChange { get; set; }

/// <summary>

/// Получает стандартное поле для добавления

/// </summary>

public Func<IField> GetNewField { get; set; }

/// <summary>

/// Id

/// </summary>

public virtual int Id

{

get => table.Id;

set => table.Id = value;

}

/// <summary>

/// true/false выделена ли таблица

/// </summary>

public virtual bool IsSelect

{

get => isSelect;

set

{

isSelect = value;

OnPropertyChanged();

}

}

/// <summary>

/// Поля таблицы

/// </summary>

public ObservableCollection<IField> Fields

=> table.Fields;

/// <summary>

/// Ширина таблицы

/// </summary>

public virtual double Width

{

get => table.Width;

set

{

if (CanWidthChange?.Invoke(value, this) == true)

{

table.Width = value;

OnWidthChanged?.Invoke(value, this);

OnPropertyChanged();

}

}

}

/// <summary>

/// Высота таблицы

/// </summary>

public virtual double Height

{

get => table.Height;

set

{

table.Height = value;

OnHeightChanged?.Invoke(value, this);

}

}

/// <summary>

/// Положение таблицы

/// </summary>

public virtual Thickness Margin

{

get => table.Margin;

set

{

if (CanHorizontalChange?.Invoke(value.Left, this) == false)

value.Left = Margin.Left;

if (CanVerticalChange?.Invoke(value.Top, this) == false)

value.Top = Margin.Top;

OnMarginChanged?.Invoke(value, this);

table.Margin = value;

OnPropertyChanged();

}

}

/// <summary>

/// Устанавливает ошибку с переданным сообщением

/// </summary>

/// <param name="message">Сообщение</param>

public override void SetError(string message)

{

FocusedElement = Focuses.None;

table.SetFieldError();

if (Fields.Count < 2)

message = Verification.ConcatString(message, "Таблица должна содержать поля, кроме PK!");

base.SetError(Verification.ConcatString(message, table.Element.GetError()));

}

/// <summary>

/// Добавляет поле

/// </summary>

/// <param name="field">Поле</param>

public virtual void AddField(IField field)

{

field.OnRemoveField += RemoveField;

Fields.Add(field);

}

/// <summary>

/// Удаляет поле

/// </summary>

/// <param name="field">Поле</param>

public virtual void RemoveField(IField field)

=> Fields.Remove(field);

/// <summary>

/// Загружает данные таблицы

/// </summary>

/// <param name="tableSave">Сохранённая информация</param>

public virtual void Load(TableSave tableSave)

{

Id = tableSave.Id;

Name = tableSave.Name;

LogicalName = tableSave.LogicalName;

ProgrammingName = tableSave.ProgrammingName;

Width = tableSave.Width;

Margin = tableSave.Margin;

}

}

}

8. UniqueErrorBase.cs. Базовый класс для проверки на уникальность.

using System.Collections.Generic;

using ScriptGen.Common;

namespace ScriptGen.Abstraction

{

/// <summary>

/// Базовый класс для проверки на уникальность

/// </summary>

public abstract class UniqueErrorBase

{

/// <summary>

/// Добавляет ошибку к сообщению

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="error">Ошибка</param>

public virtual void SetError(ref string message, string error)

=> message = Verification.ConcatString(message, error);

/// <summary>

/// Проверяет, является ли указанный элемент уникальным для коллекции

/// </summary>

/// <param name="data">Коллекция</param>

/// <param name="index">Индекс элемента для проверки</param>

/// <param name="message">Сообщение в случае ошибки</param>

/// <returns>null или сообщение об ошибке</returns>

public virtual string GetUniqueError(IEnumerable<string> data, int index, string message)

=> Verification.IsUniqueElement(data, index) == false

? message

: null;

}

}

9. API.cs. API приложения.

using System;

using Explorer;

using ScriptGenPlugin.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.API

{

/// <summary>

/// API приложения

/// </summary>

public class API : MessageAPI, IAPI

{

/// <summary>

/// Получает ошибку текущей страницы

/// </summary>

public Func<bool> GetError;

/// <summary>

/// Получает информацию о текущей странице

/// </summary>

public Func<TableInfo[]> GetTables;

/// <summary>

/// Проводник

/// </summary>

private FileDialog explorer = new FileDialog();

/// <summary>

/// Существуют ли ошибки на выбранной вкладке приложения

/// </summary>

/// <returns>true/false есть ли ошибки</returns>

public bool IsError()

=> (bool)GetError?.Invoke();

/// <summary>

/// Получает информацию о таблицах на выбранной вкладке приложения

/// </summary>

/// <returns>Информация о таблицах</returns>

public TableInfo[] GetTableInfo()

=> GetTables?.Invoke();

/// <summary>

/// Вызывает проводник для открытия файла

/// </summary>

/// <param name="canSelectFolder">true/false можно ли выделять папки</param>

/// <param name="acceptFileTypes">Разрешённые расширения файлов</param>

/// <returns>null если диалог отменён, иначе путь</returns>

public string OpenFile(bool canSelectFolder, params string[] acceptFileTypes)

{

ExplorerSelectType type = ExplorerSelectType.File;

if (canSelectFolder)

type = ExplorerSelectType.All;

return explorer.Open(type, ExplorerType.Open, acceptFileTypes) ? explorer.SelectPath : null;

}

/// <summary>

/// Вызывает проводник для сохранения файла

/// </summary>

/// <param name="canSelectFolder">true/false можно ли выделять папки</param>

/// <param name="acceptFileTypes">Разрешённые расширения файлов</param>

/// <returns>null если диалог отменён, иначе путь</returns>

public string SaveFile(bool canSelectFolder, params string[] acceptFileTypes)

{

ExplorerSelectType type = ExplorerSelectType.File;

if (canSelectFolder)

type = ExplorerSelectType.All;

return explorer.Open(type, ExplorerType.Save, acceptFileTypes) ? explorer.SelectPath : null;

}

}

}

10. MessageAPI.cs. Часть API для сообщений.

using System;

using BaseMVVM.Command;

using ScriptGen.ViewModel;

using ScriptGenPlugin.Interface;

namespace ScriptGen.API

{

/// <summary>

/// Часть API для сообщений

/// </summary>

public class MessageAPI : IMessage

{

/// <summary>

/// Выводит сообщение об ошибке

/// </summary>

/// <param name="message">Сообщение</param>

public void Error(string message)

=> StatusBarManagerVM.Error(message);

/// <summary>

/// Выводит сообщение об ошибке

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

public void Error(string message, string option)

=> StatusBarManagerVM.Error(message, option);

/// <summary>

/// Выводит сообщение об ошибке

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Error(string message, string actionTitle, Action action, Func<bool> canAction = null)

=> StatusBarManagerVM.Error(message, actionTitle, new SimpleCommand(action, canAction));

/// <summary>

/// Выводит сообщение об ошибке

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Error(string message, string option, string actionTitle, Action action, Func<bool> canAction = null)

=> StatusBarManagerVM.Error(message, option, actionTitle, new SimpleCommand(action, canAction));

/// <summary>

/// Выводит информационное сообщение

/// </summary>

/// <param name="message">Сообщение</param>

public void Info(string message)

=> StatusBarManagerVM.Info(message);

/// <summary>

/// Выводит информационное сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

public void Info(string message, string option)

=> StatusBarManagerVM.Info(message, option);

/// <summary>

/// Выводит информационное сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Info(string message, string actionTitle, Action action, Func<bool> canAction = null)

=> StatusBarManagerVM.Info(message, actionTitle, new SimpleCommand(action, canAction));

/// <summary>

/// Выводит информационное сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Info(string message, string option, string actionTitle, Action action, Func<bool> canAction = null)

=> StatusBarManagerVM.Info(message, option, actionTitle, new SimpleCommand(action, canAction));

/// <summary>

/// Выводит сообщение

/// </summary>

/// <param name="message">Сообщение</param>

public void Message(string message)

=> StatusBarManagerVM.Message(message);

/// <summary>

/// Выводит сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

public void Message(string message, string option)

=> StatusBarManagerVM.Message(message, option);

/// <summary>

/// Выводит сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Message(string message, string actionTitle, Action action, Func<bool> canAction)

=> StatusBarManagerVM.Message(message, actionTitle, new SimpleCommand(action, canAction));

/// <summary>

/// Выводит сообщение

/// </summary>

/// <param name="message">Сообщение</param>

/// <param name="option">Дополнительное сообщение</param>

/// <param name="actionTitle">Заголовок кнопки действия</param>

/// <param name="action">Действие кнопки</param>

/// <param name="canAction">Функция, определяющая можно ли выполнить действие кнопки</param>

public void Message(string message, string option, string actionTitle, Action action, Func<bool> canAction)

=> StatusBarManagerVM.Message(message, option, actionTitle, new SimpleCommand(action, canAction));

}

}

11. PluginData.cs. Модель загружаемого плагина.

using System;

using System.Linq;

using System.IO;

using System.Reflection;

using System.Xml.Serialization;

using BaseMVVM.Command;

using ScriptGen.ViewModel;

using ScriptGenPlugin.Interface;

using ScriptGen.Common;

namespace ScriptGen.API

{

/// <summary>

/// Модель загружаемого плагина

/// </summary>

public struct PluginData

{

/// <summary>

/// Название плагина

/// </summary>

public string Name { get; set; }

/// <summary>

/// Описание плагина

/// </summary>

public string Description { get; set; }

/// <summary>

/// Путь к изображению

/// </summary>

public string ImagePath { get; set; }

/// <summary>

/// Путь к dll файлу плагина

/// </summary>

public string DllPath;

/// <summary>

/// Пространство имён для точки входа в плагин

/// </summary>

public string Namespace;

/// <summary>

/// Название класса, являющийся точкой входа в плагин

/// </summary>

public string ClassName;

/// <summary>

/// Дополнительные файлы плагина

/// </summary>

public string[] Files;

/// <summary>

/// Контент кнопки действия плагина

/// </summary>

[XmlIgnore]

public object ActionContent { get; private set; }

/// <summary>

/// Подсказка кнопки действия плагина

/// </summary>

[XmlIgnore]

public string ActionToolTip { get; private set; }

/// <summary>

/// Команда плагина

/// </summary>

[XmlIgnore]

public SimpleCommand PluginCommand { get; set; }

/// <summary>

/// Удаление плагина

/// </summary>

[XmlIgnore]

public SimpleCommand RemovePlaugin { get; set; }

/// <summary>

/// Команда открытия папки плагина

/// </summary>

[XmlIgnore]

public SimpleCommand OpenInExplorer { get; set; }

/// <summary>

/// Загруженный плагин

/// </summary>

[XmlIgnore]

public IPlugin Plugin;

/// <summary>

/// Проверяет, содержат ли данные плагина ошибки

/// </summary>

/// <returns></returns>

public bool IsValid()

{

if (IsNotEmptyStringsError())

{

StatusBarManagerVM.Info("Для загрузки плагина необходимо заполнить все данные!");

return false;

}

if (IsFileExistsError())

{

StatusBarManagerVM.Info($"Не удалось загрузить файлы плагина \"{Name}\"!");

return false;

}

return IsDataError();

}

/// <summary>

/// Проверяет строковые значения плагина на пустоту

/// </summary>

/// <returns>Есть ли пустые значения</returns>

private bool IsNotEmptyStringsError()

=> string.IsNullOrWhiteSpace(Name) || string.IsNullOrWhiteSpace(Description) ||

string.IsNullOrWhiteSpace(ImagePath) || string.IsNullOrWhiteSpace(DllPath) ||

string.IsNullOrWhiteSpace(Namespace) || string.IsNullOrWhiteSpace(ClassName);

/// <summary>

/// Проверяет существование всех файлов плагина

/// </summary>

/// <returns>Есть ли несуществующий файл</returns>

private bool IsFileExistsError()

=> !File.Exists(ImagePath) || !File.Exists(DllPath) || (Files.Length > 0 && !Files.All(f => File.Exists(f)));

/// <summary>

/// Проверяет корректность точки входа в плагин

/// </summary>

private bool IsDataError()

{

try

{

Assembly asm = Assembly.LoadFrom(DllPath);

Type type = asm.GetType($"{Namespace}.{ClassName}", true, false);

if (type.GetInterface("IPlugin") == null)

{

StatusBarManagerVM.Info($"Не удалось найти реализацию интерфейса в плагине \"{Name}\"!");

return false;

}

Plugin = (IPlugin)Activator.CreateInstance(type);

Plugin.Start(PluginManager.CommonAPI);

ActionContent = Plugin.ActionContent;

ActionToolTip = Plugin.ActionToolTip;

PluginCommand = new SimpleCommand(Plugin.PluginAction, Plugin.CanPluginAction);

OpenInExplorer = new SimpleCommand(OpenPluginFolder);

return true;

}

catch (Exception e) { StatusBarManagerVM.Error($"При загрузки плагина \"{Name}\" обнаружена ошибка!", e.Message); }

return false;

}

/// <summary>

/// Открывает папку плагина в проводнике

/// </summary>

private void OpenPluginFolder()

=> FileWork.OpenInExplorer(DllPath);

}

}

12. PluginManager.cs. Позволяет осуществлять работу с плагинами.

using System;

using System.Collections.ObjectModel;

using BaseMVVM.Command;

using ScriptGen.Common;

namespace ScriptGen.API

{

/// <summary>

/// Позволяет осуществлять работу с плагинами

/// </summary>

public static class PluginManager

{

/// <summary>

/// Происходит при изменении списка плагинов

/// </summary>

public static Action OnPluginsUpdate;

/// <summary>

/// Хранит экземпляр API приложения

/// </summary>

public static API CommonAPI = new API();

/// <summary>

/// Список плагинов

/// </summary>

public static ObservableCollection<PluginData> Plugins { get; }

= new ObservableCollection<PluginData>();

/// <summary>

/// Загружает плагин

/// </summary>

/// <param name="path">Путь к файлу конфигурации</param>

public static void LoadPlugin(string path)

{

if (PluginFileWork.LoadPlugin(path, out PluginData plugin) && plugin.IsValid())

if (PluginFileWork.SaveLocalPlugin(plugin, out plugin))

{

plugin.RemovePlaugin = new SimpleCommand(() => RemovePlugin(plugin));

Plugins.Add(plugin);

}

OnPluginsUpdate?.Invoke();

}

/// <summary>

/// Загружает все установленные плагины

/// </summary>

public static void LoadPlugins()

{

if (PluginFileWork.GetAllPluginsPath(out string[] paths))

for (int i = 0; i < paths.Length; i++)

if (PluginFileWork.LoadPlugin(paths[i], out PluginData plugin) && plugin.IsValid())

{

plugin.RemovePlaugin = new SimpleCommand(() => RemovePlugin(plugin));

Plugins.Add(plugin);

}

OnPluginsUpdate?.Invoke();

}

/// <summary>

/// Удаляет плагин

/// </summary>

/// <param name="plugin">Данные плагина</param>

private static void RemovePlugin(PluginData plugin)

{

if (PluginFileWork.DisablePlugin(plugin.DllPath))

{

plugin.Plugin.Close();

Plugins.Remove(plugin);

OnPluginsUpdate?.Invoke();

}

}

/// <summary>

/// Останавливает работу всех плагинов

/// </summary>

public static void ClosePlugins()

{

for (int i = 0; i < Plugins.Count; i++)

Plugins[i].Plugin.Close();

}

}

}

13. BoolToVisibleConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

namespace ScriptGen.BindingConverter

{

public class BoolToVisibleConverter : IValueConverter

{

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

=> (bool)value ? Visibility.Visible : Visibility.Collapsed;

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

=> null;

}

}

14. DisabledFKConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

using ScriptGenPlugin.Model;

namespace ScriptGen.BindingConverter

{

public class DisabledFKConverter : IValueConverter

{

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

=> (FieldType)value == FieldType.FK ? false : true;

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

=> null;

}

}

15. DisabledPKConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

using ScriptGenPlugin.Model;

namespace ScriptGen.BindingConverter

{

public class DisabledPKConverter : IValueConverter

{

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

=> (FieldType)value == FieldType.PK ? false : true;

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

=> null;

}

}

16. FieldTypeToIconConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

using MaterialDesignThemes.Wpf;

namespace ScriptGen.BindingConverter

{

public class FieldTypeToIconConverter : IValueConverter

{

private static readonly PackIconKind[] kinds =

{

PackIconKind.Key,

PackIconKind.VectorLine,

PackIconKind.TextShort,

PackIconKind.Bracket

};

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

=> kinds[(int)value];

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

=> null;

}

}

17. InversionBool.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

namespace ScriptGen.BindingConverter

{

public class InversionBool : IValueConverter

{

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

=> !(bool)value;

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

=> null;

}

}

18. LogicalVisibleConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

using ScriptGen.Common;

namespace ScriptGen.BindingConverter

{

public class LogicalVisibleConverter : IValueConverter

{

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

=> (ModelType)value == ModelType.Logical ? Visibility.Visible : Visibility.Collapsed;

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

=> null;

}

}

19. MessageOpenConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

namespace ScriptGen.BindingConverter

{

public class MessageOpenConverter : IValueConverter

{

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

=> string.IsNullOrWhiteSpace((string)value) ? false : true;

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

=> null;

}

}

20. ModelTypeToIntConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

using ScriptGen.Common;

namespace ScriptGen.BindingConverter

{

public class ModelTypeToIntConverter : IValueConverter

{

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

=> (int)(ModelType)value;

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

=> (ModelType)(int)value;

}

}

21. PhysicalVisibleConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

using ScriptGen.Common;

namespace ScriptGen.BindingConverter

{

public class PhysicalVisibleConverter : IValueConverter

{

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

=> (ModelType)value == ModelType.Physical ? Visibility.Visible : Visibility.Collapsed;

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

=> null;

}

}

22. ProgrammingVisibleConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

using ScriptGen.Common;

namespace ScriptGen.BindingConverter

{

public class ProgrammingVisibleConverter : IValueConverter

{

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

=> (ModelType)value == ModelType.Programming ? Visibility.Visible : Visibility.Collapsed;

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

=> null;

}

}

23. SwitchMessageColorConverter.cs. Конвертер значений.

using System;

using System.Globalization;

using System.Windows.Data;

using System.Windows.Media;

using MaterialDesignThemes.Wpf;

using ScriptGen.ViewModel;

namespace ScriptGen.BindingConverter

{

public class SwitchMessageColorConverter : IValueConverter

{

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

{

PackIconKind kind = (PackIconKind)value;

if (kind == StatusBarManagerVM.ErrorIcon)

return new SolidColorBrush(Color.FromRgb(197, 17, 98));

else if (kind == StatusBarManagerVM.InfoIcon)

return new SolidColorBrush(Color.FromRgb(255, 171, 0));

return new SolidColorBrush(Color.FromRgb(0, 200, 83));

}

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

=> null;

}

}

24. ReadonlyBinding.cs. Конвертер значений.

using System.Windows;

namespace ScriptGen.BindingProperty

{

public class DataBinding

{

public static readonly DependencyProperty DataBindProperty =

DependencyProperty.RegisterAttached("DataBind", typeof(DataBindCollection),

typeof(DataBinding), new UIPropertyMetadata(null));

public static void SetDataBind(DependencyObject obj, DataBindCollection value)

=> obj.SetValue(DataBindProperty, value);

public static DataBindCollection GetDataBind(DependencyObject obj)

=> (DataBindCollection)obj.GetValue(DataBindProperty);

}

public class DataBindCollection : FreezableCollection<DataBind> { }

public class DataBind : Freezable

{

public static readonly DependencyProperty SourceProperty =

DependencyProperty.Register("Source", typeof(object), typeof(DataBind),

new FrameworkPropertyMetadata(null, new PropertyChangedCallback(OnSourceChanged)));

public object Source

{

get => GetValue(SourceProperty);

set => SetValue(SourceProperty, value);

}

private static void OnSourceChanged(DependencyObject depObj, DependencyPropertyChangedEventArgs e)

=> ((DataBind)depObj).OnSourceChanged(e);

protected virtual void OnSourceChanged(DependencyPropertyChangedEventArgs e)

=> Target = e.NewValue;

public static readonly DependencyProperty TargetProperty =

DependencyProperty.Register("Target", typeof(object),

typeof(DataBind), new FrameworkPropertyMetadata(null));

public object Target

{

get => GetValue(TargetProperty);

set => SetValue(TargetProperty, value);

}

protected override Freezable CreateInstanceCore()

=> new DataBind();

}

}

25. ScrollViewerBinding.cs. Конвертер значений.

using System.Windows;

using System.Windows.Controls;

namespace ScriptGen.BindingProperty

{

public static class ScrollViewerBinding

{

public static readonly DependencyProperty VerticalOffsetProperty =

DependencyProperty.RegisterAttached("VerticalOffset", typeof(double),

typeof(ScrollViewerBinding), new FrameworkPropertyMetadata(double.NaN, OnVerticalOffsetPropertyChanged));

private static readonly DependencyProperty VerticalScrollBindingProperty =

DependencyProperty.RegisterAttached("VerticalScrollBinding", typeof(bool?), typeof(ScrollViewerBinding));

public static double GetVerticalOffset(DependencyObject depObj)

=> (double)depObj.GetValue(VerticalOffsetProperty);

public static void SetVerticalOffset(DependencyObject depObj, double value)

=> depObj.SetValue(VerticalOffsetProperty, value);

private static void OnVerticalOffsetPropertyChanged(DependencyObject depObj, DependencyPropertyChangedEventArgs e)

{

if (!(depObj is ScrollViewer scrollViewer))

return;

double value = (double)e.NewValue;

if (value.Equals(double.NaN))

return;

BindVerticalOffset(scrollViewer);

scrollViewer.ScrollToVerticalOffset(value);

}

public static void BindVerticalOffset(ScrollViewer scrollViewer)

{

if (scrollViewer.GetValue(VerticalScrollBindingProperty) != null)

return;

scrollViewer.SetValue(VerticalScrollBindingProperty, true);

scrollViewer.ScrollChanged += (s, se) =>

{

if (se.VerticalChange == 0)

return;

SetVerticalOffset(scrollViewer, se.VerticalOffset);

};

}

public static readonly DependencyProperty HorizontalOffsetProperty =

DependencyProperty.RegisterAttached("HorizontalOffset", typeof(double),

typeof(ScrollViewerBinding), new FrameworkPropertyMetadata(double.NaN, OnHorizontalOffsetPropertyChanged));

private static readonly DependencyProperty HorizontalScrollBindingProperty =

DependencyProperty.RegisterAttached("HorizontalScrollBinding", typeof(bool?), typeof(ScrollViewerBinding));

public static double GetHorizontalOffset(DependencyObject depObj)

=> (double)depObj.GetValue(HorizontalOffsetProperty);

public static void SetHorizontalOffset(DependencyObject depObj, double value)

=> depObj.SetValue(HorizontalOffsetProperty, value);

private static void OnHorizontalOffsetPropertyChanged(DependencyObject depObj, DependencyPropertyChangedEventArgs e)

{

if (!(depObj is ScrollViewer scrollViewer))

return;

double value = (double)e.NewValue;

if (value.Equals(double.NaN))

return;

BindHorizontalOffset(scrollViewer);

scrollViewer.ScrollToHorizontalOffset(value);

}

public static void BindHorizontalOffset(ScrollViewer scrollViewer)

{

if (scrollViewer.GetValue(HorizontalScrollBindingProperty) != null)

return;

scrollViewer.SetValue(HorizontalScrollBindingProperty, true);

scrollViewer.ScrollChanged += (s, se) =>

{

if (se.HorizontalChange == 0)

return;

SetHorizontalOffset(scrollViewer, se.HorizontalOffset);

};

}

}

}

26. DataBaseFileWork.cs. Осуществляет работу с моделями базы данных.

using System.IO;

using BaseMVVM.Command;

using ScriptGen.Common.SaveModel;

using ScriptGen.ViewModel;

namespace ScriptGen.Common

{

/// <summary>

/// Осуществляет работу с моделями базы данных

/// </summary>

public static class DataBaseFileWork

{

public static void Save(DataBaseSave dataBaseSave, string path)

{

if (!File.Exists(path) && Directory.Exists(path))

path = string.IsNullOrWhiteSpace(dataBaseSave.Name)

? FileWork.GetUniqueFileName(path, "dbSave", ".xml")

: FileWork.GetUniqueFileName(path, dataBaseSave.Name, ".xml");

if (FileWork.XmlSave(dataBaseSave, path, typeof(DataBaseSave), out string message))

{

StatusBarManagerVM.Message($"Модель \"{dataBaseSave.Name}\" сохранена!", "перейти",

new SimpleCommand(() => FileWork.OpenInExplorer(path)));

}

else

StatusBarManagerVM.Error($"Не удалось сохранить модель!", message);

}

public static bool Load(string path, out DataBaseSave dataBase)

{

dataBase = new DataBaseSave();

if (FileWork.XmlLoad(path, typeof(DataBaseSave), out string message, out object db))

{

dataBase = (DataBaseSave)db;

return true;

}

else

{

StatusBarManagerVM.Error($"Не удалось загрузить модель!", message, "перейти",

new SimpleCommand(() => FileWork.OpenInExplorer(path)));

}

return false;

}

}

}

27. DataConverter.cs. Осуществляет преобразование данных.

using System.Linq;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Common

{

/// <summary>

/// Осуществляет преобразование данных

/// </summary>

public static class DataConverter

{

private static FieldSave IFieldToSave(IField field)

=> new FieldSave()

{

Type = field.Type,

Name = field.Name,

LogicalName = field.LogicalName,

ProgrammingName = field.ProgrammingName,

DataType = field.DataType,

ProgrammingType = field.ProgrammingType,

IsNull = field.IsNull,

IsUnique = field.IsUnique,

RefTableId = field.RefTable != null ? (int?)field.RefTable.Id : null

};

private static TableSave ITableToSave(ITable table)

=> new TableSave()

{

Id = table.Id,

Width = table.Width,

Margin = table.Margin,

Name = table.Name,

LogicalName = table.LogicalName,

ProgrammingName = table.ProgrammingName,

Fields = table.Fields

.Where(f => f.Type != FieldType.FK)

.Select(f => IFieldToSave(f)).ToArray()

};

private static LineSave ILineToSave(ILine line)

=> new LineSave()

{

Field = IFieldToSave(line.Field),

SourceId = line.Source.Id,

TargetId = line.Target.Id,

SourceX1 = line.SourceX,

TargetX1 = line.TargetX,

SourceY1 = line.SourceY,

TargetY1 = line.TargetY,

ConnectionX1 = line.ConnectionX

};

public static DataBaseSave IPageToSave(IPage page)

{

for (int i = 0; i < page.Tables.Count; i++)

page.Tables[i].Id = i + 1;

return new DataBaseSave()

{

Name = page.Name,

Tables = page.Tables.Select(t => ITableToSave(t)).ToArray(),

Lines = page.Lines.Select(l => ILineToSave(l)).ToArray()

};

}

private static TableInfo[] Tables;

private static FieldInfo IFieldToInfo(IField field)

=> new FieldInfo()

{

Type = field.Type,

Name = field.Name,

LogicalName = field.LogicalName,

ProgrammingName = field.ProgrammingName,

DataType = field.Type == FieldType.FK

? field.RefTable.Fields[0].DataType

: field.DataType,

ProgrammingType = field.Type == FieldType.FK

? field.RefTable.Fields[0].ProgrammingType

: field.ProgrammingType,

IsNull = field.IsNull,

IsUnique = field.IsUnique,

RefTable = field.RefTable != null

? Tables[field.RefTable.Id]

: null

};

private static TableInfo ITableToInfo(ITable table)

=> new TableInfo()

{

Name = table.Name,

LogicalName = table.LogicalName,

ProgrammingName = table.ProgrammingName

};

public static TableInfo[] IPageToInfo(IPage page)

{

Tables = new TableInfo[page.Tables.Count];

for (int i = 0; i < page.Tables.Count; i++)

{

page.Tables[i].Id = i;

Tables[i] = ITableToInfo(page.Tables[i]);

}

for (int i = 0; i < Tables.Length; i++)

Tables[i].Fields = page.Tables[i].Fields.Select(f => IFieldToInfo(f)).ToArray();

for (int i = 0; i < Tables.Length; i++)

Tables[i].RefTables = Tables.Where(t => t.Fields.Count(f => f.RefTable == Tables[i]) > 0).ToArray();

return Tables;

}

}

}

28. Enums.cs. Перечисления.

namespace ScriptGen.Common

{

/// <summary>

/// Тип модели

/// </summary>

public enum ModelType

{

/// <summary>

/// Физическая модель

/// </summary>

Physical = 0,

/// <summary>

/// Логическая модель

/// </summary>

Logical = 1,

/// <summary>

/// Модель программирования

/// </summary>

Programming = 2

}

/// <summary>

/// Элементы для фокуса

/// </summary>

public enum Focuses

{

/// <summary>

/// Отсутствует

/// </summary>

None = 0,

/// <summary>

/// Поле названия

/// </summary>

Name = 1,

/// <summary>

/// Поле логического названия

/// </summary>

LogicalName = 2,

/// <summary>

/// Поле названия для языков программирования

/// </summary>

ProgrammingName = 3,

/// <summary>

/// Поле типа данных

/// </summary>

DataType = 4,

/// <summary>

/// Поле типа данных для языков программирования

/// </summary>

ProgrammingType = 5

}

/// <summary>

/// Тип шаблона

/// </summary>

public enum TemplateType

{

/// <summary>

/// Шаблон SQL

/// </summary>

SQL = 1,

/// <summary>

/// Шаблон для языков программирования

/// </summary>

Programming = 2

}

}

29. FileWork.cs. осуществляет работу с файлами.

using System;

using System.Diagnostics;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Xml.Serialization;

using BaseMVVM.Command;

using ScriptGen.ViewModel;

namespace ScriptGen.Common

{

/// <summary>

/// осуществляет работу с файлами

/// </summary>

public static class FileWork

{

public static void OpenInExplorer(string path)

{

if (Directory.Exists(path) || File.Exists(path))

Process.Start(new ProcessStartInfo("explorer.exe", " /select, " + path));

else

StatusBarManagerVM.Info("Путь не найден, возможно файл удалён или перемещён!", path);

}

public static string GetUniqueFileName(string path, string startName, string extension)

{

string name = $"{startName}{extension}";

int count = 1;

while (File.Exists(Path.Combine(path, name)))

{

name = $"{startName} ({count}){extension}";

count++;

}

return Path.Combine(path, name);

}

public static string GetUniqueDirectoryName(string path, string startName)

{

string name = startName;

int count = 1;

while (Directory.Exists(Path.Combine(path, name)))

{

name = $"{startName} ({count})";

count++;

}

return Path.Combine(path, name);

}

public static string MoveFile(string filePath, string folderPath)

{

FileInfo file = new FileInfo(filePath);

string newPath = Path.Combine(folderPath, file.Name);

if (File.Exists(newPath))

newPath = GetUniqueFileName(folderPath, file.Name.Substring(0, file.Name.Length - file.Extension.Length), file.Extension);

File.Copy(file.FullName, newPath);

return newPath;

}

public static bool XmlSave(object saveObject, string path, Type type, out string errorMessage)

{

errorMessage = null;

try

{

XmlSerializer formatter = new XmlSerializer(type);

using (FileStream fs = new FileStream(path, FileMode.Create))

{

formatter.Serialize(fs, saveObject);

}

return true;

}

catch (Exception e) { errorMessage = e.Message; }

return false;

}

public static bool BinarySave(object saveObject, string path, out string errorMessage)

{

errorMessage = null;

try

{

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream fs = new FileStream(path, FileMode.Create))

{

formatter.Serialize(fs, saveObject);

}

return true;

}

catch (Exception e) { errorMessage = e.Message; }

return false;

}

public static bool XmlLoad(string path, Type type, out string errorMessage, out object loadObject)

{

loadObject = null;

errorMessage = null;

try

{

XmlSerializer formatter = new XmlSerializer(type);

using (FileStream fs = new FileStream(path, FileMode.Open))

{

loadObject = formatter.Deserialize(fs);

}

return true;

}

catch (Exception e) { errorMessage = e.Message; }

return false;

}

public static bool BinaryLoad(string path, out string errorMessage, out object loadObject)

{

loadObject = null;

errorMessage = null;

try

{

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream fs = new FileStream(path, FileMode.Open))

{

loadObject = formatter.Deserialize(fs);

}

return true;

}

catch (Exception e) { errorMessage = e.Message; }

return false;

}

public static void TextSave(StringBuilder script, string path)

{

try

{

File.WriteAllText(path, script.ToString(), Encoding.UTF8);

FileInfo file = new FileInfo(path);

StatusBarManagerVM.Message($"Файл \"{file.Name}\" сохранен!", "перейти",

new SimpleCommand(() => OpenInExplorer(path)));

}

catch (Exception e) { StatusBarManagerVM.Error($"Не удалось сохранить файл!", e.Message); }

}

public static void TextSave(StringBuilder script, string path, string name, string extension)

{

if (!File.Exists(path) && Directory.Exists(path))

path = GetUniqueFileName(path, name, extension);

try

{

File.WriteAllText(path, script.ToString(), Encoding.UTF8);

FileInfo file = new FileInfo(path);

StatusBarManagerVM.Message($"Файл \"{file.Name}\" сохранен!", "перейти",

new SimpleCommand(() => OpenInExplorer(path)));

}

catch (Exception e) { StatusBarManagerVM.Error($"Не удалось сохранить файл!", e.Message); }

}

}

}

30. Info.cs. Содержит общую информацию приложения.

using System;

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Settings.Model;

namespace ScriptGen.Common

{

/// <summary>

/// Содержит общую информацию приложения

/// </summary>

public static class Info

{

/// <summary>

/// Происходит при изменении настроек

/// </summary>

public static event Action OnSettingsChange;

/// <summary>

/// Происходит при добавлении настроек

/// </summary>

public static event Action OnSettingsAdd;

/// <summary>

/// Экземпляры настроек приложения

/// </summary>

private static List<Option> options = new List<Option>();

/// <summary>

/// Шаблоны для SQL

/// </summary>

private static List<TemplateData> sqlTemplates = new List<TemplateData>();

/// <summary>

/// Шаблоны для языков программирования

/// </summary>

private static List<TemplateData> programmingTemplates = new List<TemplateData>();

/// <summary>

/// Индекс активных настроек

/// </summary>

private static int activeOptionIndex = -1;

/// <summary>

/// Индекс активного SQL шаблона

/// </summary>

private static int activeTemplateSQLIndex = -1;

/// <summary>

/// Индекс активного шаблона для языков программирования

/// </summary>

private static int activeTemplateProgrammingIndex = -1;

/// <summary>

/// Индекс активных настроек

/// </summary>

public static int ActiveOptionIndex

{

get => activeOptionIndex;

set

{

activeOptionIndex = (value >= 0 && value < options.Count) ? value : -1;

OnSettingsChange?.Invoke();

}

}

/// <summary>

/// Индекс активного SQL шаблона

/// </summary>

public static int ActiveTemplateSQLIndex

{

get => activeTemplateSQLIndex;

set

{

activeTemplateSQLIndex = (value >= 0 && value < options.Count) ? value : -1;

OnSettingsChange?.Invoke();

}

}

/// <summary>

/// Индекс активного шаблона для языков программирования

/// </summary>

public static int ActiveTemplateProgrammingIndex

{

get => activeTemplateProgrammingIndex;

set

{

activeTemplateProgrammingIndex = (value >= 0 && value < options.Count) ? value : -1;

OnSettingsChange?.Invoke();

}

}

/// <summary>

/// Активные настройки

/// </summary>

private static Option option

=> ActiveOptionIndex != -1 ? options[ActiveOptionIndex] : null;

/// <summary>

/// Активный SQL шаблон

/// </summary>

public static string TemplateSQL

=> ActiveTemplateSQLIndex != -1 ? sqlTemplates[ActiveTemplateSQLIndex].Template : "";

/// <summary>

/// Активный шаблон для языков программирования

/// </summary>

public static string TemplateProgramming

=> ActiveTemplateProgrammingIndex != -1 ? programmingTemplates[ActiveTemplateProgrammingIndex].Template : "";

/// <summary>

/// Названия всех настроек

/// </summary>

public static string[] OptionNames

=> options.Select(x => x.Name).ToArray();

/// <summary>

/// Названия всех шаблонов SQL

/// </summary>

public static string[] SqlNames

=> sqlTemplates.Select(x => x.Name).ToArray();

/// <summary>

/// Названия всех шаблонов для языков программирования

/// </summary>

public static string[] ProgrammingNames

=> programmingTemplates.Select(x => x.Name).ToArray();

/// <summary>

/// Список недопустимых слов

/// </summary>

public static string[] FalseWords

=> option != null ? option.FalseWords : new string[0];

/// <summary>

/// Список SQL типов данных

/// </summary>

public static string[] SQLTypes

=> option != null ? option.SQLTypes : new string[0];

/// <summary>

/// Список типов данных для языков программирования

/// </summary>

public static string[] ProgrammingTypes

=> option != null ? option.ProgrammingTypes : new string[0];

/// <summary>

/// Добавляет экземпляр настроек

/// </summary>

/// <param name="optionData">Экземпляр настроек</param>

public static void AddOption(OptionData optionData)

{

options.Add(new Option(optionData));

OnSettingsAdd?.Invoke();

}

/// <summary>

/// Добавляет шаблон

/// </summary>

/// <param name="templateData">Данные о шаблоне</param>

public static void AddTemplate(TemplateData templateData)

{

if (templateData.Type == TemplateType.SQL)

sqlTemplates.Add(templateData);

else if (templateData.Type == TemplateType.Programming)

programmingTemplates.Add(templateData);

OnSettingsAdd?.Invoke();

}

}

}

31. PluginFileWork.cs. Работа с файлами для плагинов.

using System;

using System.IO;

using System.Linq;

using BaseMVVM.Command;

using ScriptGen.API;

using ScriptGen.ViewModel;

namespace ScriptGen.Common

{

/// <summary>

/// Работа с файлами для плагинов

/// </summary>

public static class PluginFileWork

{

/// <summary>

/// Сохраняет файлы плагина локально

/// </summary>

/// <param name="pluginData">Данные о плагине</param>

/// <param name="newPluginData">Измененные данные о плагине</param>

/// <returns>true/false успешно ли сохранение</returns>

public static bool SaveLocalPlugin(PluginData pluginData, out PluginData newPluginData)

{

newPluginData = new PluginData();

string path = "";

try

{

DirectoryInfo directory = new DirectoryInfo(@"Data\Plugins");

if (!directory.Exists)

directory = Directory.CreateDirectory(@"Data\Plugins");

path = FileWork.GetUniqueDirectoryName(directory.FullName, pluginData.Name);

Directory.CreateDirectory(path);

pluginData.ImagePath = FileWork.MoveFile(pluginData.ImagePath, path);

pluginData.DllPath = FileWork.MoveFile(pluginData.DllPath, path);

for (int i = 0; i < pluginData.Files.Length; i++)

pluginData.Files[i] = FileWork.MoveFile(pluginData.Files[i], path);

}

catch (Exception e)

{

StatusBarManagerVM.Error($"Не удалось скопировать ресурсы плагин \"{pluginData.Name}\"!", e.Message);

return false;

}

path = Path.Combine(path, "config.xml");

if (FileWork.XmlSave(pluginData, path, typeof(PluginData), out string message))

{

StatusBarManagerVM.Message($"Плагин \"{pluginData.Name}\" скопирован локально!");

newPluginData = pluginData;

return true;

}

else

StatusBarManagerVM.Error($"Не удалось скопировать плагин \"{pluginData.Name}\"!");

return false;

}

/// <summary>

/// Загружает данные о плагине по указанному пути

/// </summary>

/// <param name="path">Путь к папке плагина</param>

/// <param name="optionData">Данные о плагине</param>

/// <returns>true/false успешна ли загрузка</returns>

public static bool LoadPlugin(string path, out PluginData pluginData)

{

pluginData = new PluginData();

if (FileWork.XmlLoad(path, typeof(PluginData), out string message, out object pd))

{

pluginData = (PluginData)pd;

return true;

}

else

{

StatusBarManagerVM.Error($"Не удалось загрузить плагин!", message, "перейти",

new SimpleCommand(() => FileWork.OpenInExplorer(path)));

}

return false;

}

/// <summary>

/// Получает пути всех папок плагинов

/// </summary>

/// <param name="paths">Пути</param>

/// <returns>true/false были ли получены пути</returns>

public static bool GetAllPluginsPath(out string[] paths)

{

paths = new string[0];

try

{

if (Directory.Exists(@"Data\Plugins"))

paths = Directory.GetDirectories(@"Data\Plugins").Select(p => Path.Combine(p, "config.xml")).ToArray();

if (paths.Length > 0)

paths = RemoveDisablePlugin(paths);

return paths.Length > 0;

}

catch (Exception e) { StatusBarManagerVM.Error($"Не удалось загрузить плагины!", e.Message); }

return false;

}

/// <summary>

/// Удаляет папки отключённых плагинов

/// </summary>

/// <param name="paths">Пути к папкам плагинов</param>

/// <returns>Пути активных плагинов</returns>

private static string[] RemoveDisablePlugin(string[] paths)

{

try

{

for (int i = 0; i < paths.Length; i++)

if (!File.Exists(paths[i]))

{

FileInfo file = new FileInfo(paths[i]);

Directory.Delete(file.Directory.FullName, true);

paths[i] = null;

}

}

catch (Exception e) { StatusBarManagerVM.Error("Не удалось удалить папку плагина!", e.Message); }

return paths.Where(p => p != null).ToArray();

}

/// <summary>

/// Отключает плагин

/// </summary>

/// <param name="path">Путь к папке плагина</param>

/// <returns>true/false успешно ли отключение</returns>

public static bool DisablePlugin(string path)

{

try

{

FileInfo file = new FileInfo(path);

File.Delete(Path.Combine(file.Directory.FullName, "config.xml"));

StatusBarManagerVM.Info("Плагин отключён!", "Файлы плагина будут удалены при следующем запуске программы!");

return true;

}

catch (Exception e) { StatusBarManagerVM.Error("Не удалось удалить плагин!", e.Message); }

return false;

}

}

}

32. SettingsFileWork.cs. Работа с файлами для настроек.

using System;

using System.IO;

using BaseMVVM.Command;

using ScriptGen.Settings.Model;

using ScriptGen.ViewModel;

namespace ScriptGen.Common

{

/// <summary>

/// Работа с файлами для настроек

/// </summary>

public static class SettingsFileWork

{

/// <summary>

/// Сохраняет файл локально

/// </summary>

/// <param name="filePath">Путь к файлу</param>

/// <param name="folderName">Путь к папку</param>

/// <param name="errorMessage">Сообщение при ошибке</param>

/// <returns>true/false успешно ли сохранение</returns>

private static bool SaveLocal(string filePath, string folderName, string errorMessage)

{

try

{

DirectoryInfo directory = new DirectoryInfo(folderName);

if (!directory.Exists)

directory = Directory.CreateDirectory(folderName);

FileWork.MoveFile(filePath, directory.FullName);

return true;

}

catch (Exception e) { StatusBarManagerVM.Error(errorMessage, e.Message); }

return false;

}

/// <summary>

/// Локально сохраняет файл шаблона

/// </summary>

/// <param name="path">Путь к файлу шаблона</param>

/// <returns>true/false успешно ли сохранение</returns>

public static bool SaveLocalTemplate(string path)

=> SaveLocal(path, @"Data\Templates", "Не удалось скопировать файл шаблона!");

/// <summary>

/// Локально сохраняет файл настроек

/// </summary>

/// <param name="path">Путь к файлу настроек</param>

/// <returns>true/false успешно ли сохранение</returns>

public static bool SaveLocalOption(string path)

=> SaveLocal(path, @"Data\Options", "Не удалось скопировать файл настроек!");

/// <summary>

/// Загружает шаблон по указанному пути

/// </summary>

/// <param name="path">Путь к файлу</param>

/// <param name="optionData">Шаблон</param>

/// <returns>true/false успешна ли загрузка</returns>

public static bool LoadTemplate(string path, out TemplateData templateData)

{

templateData = new TemplateData();

if (FileWork.XmlLoad(path, typeof(TemplateData), out string message, out object td))

{

templateData = (TemplateData)td;

return true;

}

else

{

StatusBarManagerVM.Error($"Не удалось загрузить шаблон!", message, "перейти",

new SimpleCommand(() => FileWork.OpenInExplorer(path)));

}

return false;

}

/// <summary>

/// Загружает экземпляр настройки по указанному пути

/// </summary>

/// <param name="path">Путь к файлу</param>

/// <param name="optionData">Экземпляр настроек</param>

/// <returns>true/false успешна ли загрузка</returns>

public static bool LoadOption(string path, out OptionData optionData)

{

optionData = new OptionData();

if (FileWork.XmlLoad(path, typeof(OptionData), out string message, out object od))

{

optionData = (OptionData)od;

return true;

}

else

{

StatusBarManagerVM.Error($"Не удалось загрузить вариант настроек!", message, "перейти",

new SimpleCommand(() => FileWork.OpenInExplorer(path)));

}

return false;

}

/// <summary>

/// Получает пути всех файлов в папке

/// </summary>

/// <param name="paths">Пути</param>

/// <param name="folderName">Папка</param>

/// <param name="errorMessage">Сообщение при ошибке</param>

/// <returns>true/false были ли получены пути</returns>

private static bool GetAllPath(out string[] paths, string folderName, string errorMessage)

{

paths = new string[0];

try

{

if (Directory.Exists(folderName))

paths = Directory.GetFiles(folderName);

return paths.Length > 0;

}

catch (Exception e) { StatusBarManagerVM.Error(errorMessage, e.Message); }

return false;

}

/// <summary>

/// Получает пути всех файлов шаблонов

/// </summary>

/// <param name="paths">Пути</param>

/// <returns>true/false были ли получены пути</returns>

public static bool GetAllTemplatesPath(out string[] paths)

=> GetAllPath(out paths, @"Data\Templates", "Не удалось загрузить шаблон!");

/// <summary>

/// Получает пути всех файлов настроек

/// </summary>

/// <param name="paths">Пути</param>

/// <returns>true/false были ли получены пути</returns>

public static bool GetAllOptionsPath(out string[] paths)

=> GetAllPath(out paths, @"Data\Options", "Не удалось загрузить вариант настроек!");

}

}

33. Verification.cs. Проверка строк.

using System.Collections.Generic;

using System.Linq;

namespace ScriptGen.Common

{

/// <summary>

/// Проверка строк

/// </summary>

public static class Verification

{

/// <summary>

/// Возвращает объединение двух строк через \n, если одна из строк пустая вернёт другую

/// </summary>

/// <param name="string1">Первая строка</param>

/// <param name="string2">Вторая строка</param>

/// <returns>Результат</returns>

public static string ConcatString(string string1, string string2)

=> !string.IsNullOrEmpty(string1) && !string.IsNullOrEmpty(string2)

? $"{string1}\n{string2}"

: !string.IsNullOrEmpty(string1)

? string1

: string2;

/// <summary>

/// Проверяет, содержит ли строка только английские буквы

/// </summary>

/// <param name="data">Строка</param>

/// <returns>true/false содержит ли строка только английские буквы</returns>

public static bool IsOnlyEnLetter(string data)

=> data.ToLower().All(c => "qwertyuiopasdfghjklzxcvbnm".Contains(c));

/// <summary>

/// Проверяет, содержит ли строка только английские буквы и разрешённые символы

/// </summary>

/// <param name="data">Строка</param>

/// <param name="allowedChar">Разрешённые символы</param>

/// <returns>true/false содержит ли строка только английские буквы и разрешённые символы</returns>

public static bool IsOnlyEnLetterOrAllowedChar(string data, params char[] allowedChar)

=> data.ToLower().All(c => "qwertyuiopasdfghjklzxcvbnm".Contains(c) || allowedChar.Contains(c));

/// <summary>

/// Проверяет, содержит ли строка только буквы и разрешённые символы

/// </summary>

/// <param name="data">Строка</param>

/// <param name="allowedChar">Разрешённые символы</param>

/// <returns>true/false содержит ли строка только буквы и разрешённые символы</returns>

public static bool IsOnlyLetterOrAllowedChar(string data, params char[] allowedChar)

=> data.ToLower().All(c => char.IsLetter(c) || allowedChar.Contains(c));

/// <summary>

/// Проверяет, содержит ли строка только буквы, цифры и разрешённые символы

/// </summary>

/// <param name="data">Строка</param>

/// <param name="allowedChar">Разрешённые символы</param>

/// <returns>true/false содержит ли строка только буквы, цифры и разрешённые символы</returns>

public static bool IsOnlyEnLetterOrDigitOrAllowedChar(string data, params char[] allowedChar)

=> data.ToLower().All(c => "qwertyuiopasdfghjklzxcvbnm1234567890".Contains(c) || allowedChar.Contains(c));

/// <summary>

/// Проверяет, является ли строка словом из списка недопустимых

/// </summary>

/// <param name="data">Строка</param>

/// <returns>true/false является ли строка словом из списка недопустимых</returns>

public static bool IsFalseWord(string data)

=> Info.FalseWords.Contains(data.ToLower());

/// <summary>

/// Проверяет является ли элемент уникальным для последовательности

/// </summary>

/// <param name="data">Последовательность</param>

/// <param name="index">Индекс элемента, который необходимо проверить</param>

/// <returns>true/false является ли элемент уникальным для последовательности</returns>

public static bool IsUniqueElement(IEnumerable<string> data, int index)

{

if (index < 0 || index > data.Count())

return true;

string element = data.ElementAt(index);

return string.IsNullOrEmpty(element)

? true

: data.Count(s => !string.IsNullOrWhiteSpace(s) && s.Equals(element)) == 1;

}

}

}

34. DataBaseSave.cs. Модель сохранения данных о базе данных.

namespace ScriptGen.Common.SaveModel

{

/// <summary>

/// Модель сохранения данных о базе данных

/// </summary>

public struct DataBaseSave

{

public string Name;

public TableSave[] Tables;

public LineSave[] Lines;

}

}

35. FieldSave.cs. Модель сохранения поля таблицы.

using ScriptGenPlugin.Model;

namespace ScriptGen.Common.SaveModel

{

/// <summary>

/// Модель сохранения поля таблицы

/// </summary>

public struct FieldSave

{

public FieldType Type;

public string Name;

public string LogicalName;

public string ProgrammingName;

public string DataType;

public string ProgrammingType;

public bool IsNull;

public bool IsUnique;

public int? RefTableId;

}

}

36. LineSave.cs. Модель сохранения линии связи.

namespace ScriptGen.Common.SaveModel

{

/// <summary>

/// Модель сохранения линии связи

/// </summary>

public struct LineSave

{

public FieldSave Field;

public int SourceId;

public int TargetId;

public double SourceX1;

public double TargetX1;

public double SourceY1;

public double TargetY1;

public double ConnectionX1;

}

}

37. TableSave.cs. Модель сохранения таблицы.

using System.Windows;

namespace ScriptGen.Common.SaveModel

{

/// <summary>

/// Модель сохранения таблицы

/// </summary>

public struct TableSave

{

public int Id;

public double Width;

public Thickness Margin;

public string Name;

public string LogicalName;

public string ProgrammingName;

public FieldSave[] Fields;

}

}

38. FakeHelper.cs. Осуществляет выделение параметров шаблона.

using System;

using System.Linq;

using System.Text.RegularExpressions;

namespace ScriptGen.Faker

{

/// <summary>

/// Осуществляет выделение параметров шаблона

/// </summary>

public static class FakeHelper

{

public static Random GetRandom()

=> new Random(Guid.NewGuid().GetHashCode());

public static int[] GetRange(string parameter)

=> parameter.Split('-').Select(s => int.TryParse(s, out int value) ? value : -1).Where(i => i > 0).ToArray();

public static string[] GetParameters(string parameter)

=> parameter.Split(',').Where(s => !string.IsNullOrEmpty(s)).ToArray();

public static char GetLanguageChar(ref string parameter)

{

char result = '-';

if (parameter.Length > 0 && char.IsLetter(parameter[0]))

{

result = parameter[0];

parameter = parameter.Substring(1);

}

return result;

}

public static string[] LoadWords(string resource)

{

string result = Properties.Resources.ResourceManager.GetString(resource);

return !string.IsNullOrEmpty(result)

? Regex.Split(result, "\\s+")

: new string[1] { "" };

}

}

}

39. FakeChar.cs. Генерирует случайные символы.

using System;

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Interface;

namespace ScriptGen.Faker.Common

{

/// <summary>

/// Генерирует случайные символы

/// </summary>

public class FakeChar : IFaker

{

private static readonly Dictionary<char, string> data

= new Dictionary<char, string>()

{

{ 'r', "йцукенгшщзхъфывапролджэячсмитьбю" },

{ 'e', "qwertyuiopasdfghjklzxcvbnm" }

};

public string Next(int count, char language)

{

string result = "";

string chars = GetChars(language);

Random random = FakeHelper.GetRandom();

for (int i = 0; i < count; i++)

result += chars[random.Next(0, chars.Length)];

return result;

}

public string Next(int minCount, int maxCount, char language)

{

string result = "";

string chars = GetChars(language);

Random random = FakeHelper.GetRandom();

for (int i = 0; i < random.Next(minCount, maxCount); i++)

result += chars[random.Next(0, chars.Length)];

return result;

}

public string Next(string[] values)

{

Random random = FakeHelper.GetRandom();

return values.All(s => s.Length == 1)

? values[random.Next(0, values.Length)]

: "";

}

private static string GetChars(char language)

=> data.ContainsKey(language)

? data[language]

: string.Join("", data.Select(d => d.Value));

}

}

40. FakeName.cs. Генерирует случайные имена.

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Abstraction;

namespace ScriptGen.Faker.Common

{

/// <summary>

/// Генерирует случайные имена

/// </summary>

public class FakeName : FakeWordBase

{

private static Dictionary<char, string[]> data

= new Dictionary<char, string[]>();

public static void Load()

{

data.Add('r', FakeHelper.LoadWords("RU\_Name"));

data.Add('e', FakeHelper.LoadWords("EN\_Name"));

}

protected override string[] GetWords(char language)

=> data.ContainsKey(language)

? data[language]

: data.SelectMany(d => d.Value).ToArray();

}

}

41. FakeNumber.cs. Генерирует случайные цифры.

using System;

using System.Linq;

using ScriptGen.Interface;

namespace ScriptGen.Faker.Common

{

/// <summary>

/// Генерирует случайные цифры

/// </summary>

public class FakeNumber : IFaker

{

private static readonly string data = "0123456789";

public string Next(int count, char language)

{

string result = "";

Random random = FakeHelper.GetRandom();

for (int i = 0; i < count; i++)

result += data[random.Next(0, 10)];

return result;

}

public string Next(int minCount, int maxCount, char language)

{

string result = "";

Random random = FakeHelper.GetRandom();

for (int i = 0; i < random.Next(minCount, maxCount); i++)

result += data[random.Next(0, 10)];

return result;

}

public string Next(string[] values)

{

Random random = FakeHelper.GetRandom();

return values.All(s => double.TryParse(s.Replace('.', ','), out double value))

? values[random.Next(0, values.Length)]

: "";

}

}

}

42. FakeSurname.cs. Генерирует случайные фамилии.

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Abstraction;

namespace ScriptGen.Faker.Common

{

/// <summary>

/// Генерирует случайные фамилии

/// </summary>

public class FakeSurname : FakeWordBase

{

private static Dictionary<char, string[]> data

= new Dictionary<char, string[]>();

public static void Load()

{

data.Add('r', FakeHelper.LoadWords("RU\_Surname"));

data.Add('e', FakeHelper.LoadWords("EN\_Surname"));

}

protected override string[] GetWords(char language)

=> data.ContainsKey(language)

? data[language]

: data.SelectMany(d => d.Value).ToArray();

}

}

43. FakeWord.cs. Генерирует случайные слова.

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Abstraction;

namespace ScriptGen.Faker.Common

{

/// <summary>

/// Генерирует случайные слова

/// </summary>

public class FakeWord : FakeWordBase

{

private static Dictionary<char, string[]> data

= new Dictionary<char, string[]>();

public static void Load()

{

data.Add('r', FakeHelper.LoadWords("RU\_Word"));

data.Add('e', FakeHelper.LoadWords("EN\_Word"));

}

protected override string[] GetWords(char language)

=> data.ContainsKey(language)

? data[language]

: data.SelectMany(d => d.Value).ToArray();

}

}

44. FakerM.cs. Модель генерации случайного значения.

using System;

using System.Collections.Generic;

using System.Linq;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using ScriptGen.Faker.Common;

using ScriptGen.Interface;

namespace ScriptGen.Faker.Model

{

/// <summary>

/// Модель генерации случайного значения

/// </summary>

public class FakerM : ViewModelBase

{

public event Action<FakerM> OnRemoveFaker;

private static readonly Dictionary<char, IFaker> keywords

= new Dictionary<char, IFaker>()

{

{ 'd', new FakeNumber() },

{ 'l', new FakeChar() },

{ 'w', new FakeWord() },

{ 'n', new FakeName() },

{ 's', new FakeSurname() }

};

private string templateText = "Пользовательский";

private bool isUnique = false;

private string template = "";

private int reRandomCount;

private List<string> wasValues = new List<string>();

public FakerM()

=> RemoveFaker = new SimpleCommand(() => OnRemoveFaker?.Invoke(this));

public SimpleCommand RemoveFaker { get; }

public static Dictionary<string, string> DefaultTemplates { get; }

= new Dictionary<string, string>()

{

{ "Пользовательский", "" },

{ "Имя", "n{r1}" },

{ "Фамилия", "s{r1}" },

{ "Текст", "w{r10-100}" },

{ "Телефон", "+d{7,8} (d{3}) d{3}-d{2}-d{2}" },

{ "Логин", "l{e4-8}d{2-4}@gmail.com" },

{ "Пароль", "l{e4-8}d{2-4}l{e3-5}d{0-2}" },

{ "Пол", "l{м,ж}" }

};

public string TemplateText

{

get => templateText;

set

{

templateText = value;

Template = DefaultTemplates[templateText];

OnPropertyChanged();

}

}

public bool IsUnique

{

get => isUnique;

set

{

isUnique = value;

OnPropertyChanged();

}

}

public string Template

{

get => template;

set

{

if (!string.IsNullOrEmpty(value))

{

template = value;

if (TemplateText != "Пользовательский" && !DefaultTemplates.ContainsValue(value))

TemplateText = "Пользовательский";

}

OnPropertyChanged();

}

}

public bool Randomize(out string result)

{

result = "";

string template = Template;

while (!string.IsNullOrEmpty(template))

{

char command = template[0];

template = template.Substring(1);

if (keywords.ContainsKey(command) && template.Length > 2 && template[0] == '{' && template.Contains('}'))

{

string parameter = template.Substring(1, template.IndexOf('}') - 1);

template = template.Substring(parameter.Length + 2);

result += KeywordInvoke(command, parameter);

}

else

result += command;

}

if (IsUnique && wasValues.Contains(result))

return reRandomCount++ < 100

? Randomize(out result)

: false;

wasValues.Add(result);

reRandomCount = 0;

return true;

}

public void Reset()

=> wasValues.Clear();

private string KeywordInvoke(char command, string parameter)

{

char language = FakeHelper.GetLanguageChar(ref parameter);

if (int.TryParse(parameter, out int count) && count > 0)

return keywords[command].Next(count, language);

else if (parameter.Contains('-'))

{

int[] parameters = FakeHelper.GetRange(parameter);

if (parameters.Length == 2 && parameters[0] < parameters[1] && parameters[0] >= 0)

return keywords[command].Next(parameters[0], parameters[1], language);

}

else if (parameter.Contains(','))

{

string[] parameters = FakeHelper.GetParameters(parameter);

if (parameters.Length > 0)

return keywords[command].Next(parameters);

}

return "";

}

}

}

45. FakerV.xaml. Представления для генератора случайных значений.

<UserControl

x:Class="ScriptGen.Faker.View.FakerV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="44"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Border

BorderBrush="{DynamicResource PrimaryHueDarkBrush}"

BorderThickness="0 0 0 2"

VerticalAlignment="Bottom"/>

<StackPanel

Orientation="Horizontal"

Margin="20 0 0 0">

<TextBlock

Style="{StaticResource MaterialDesignSubheadingTextBlock}"

VerticalAlignment="Center"

Text="Faker"/>

</StackPanel>

<StackPanel

Orientation="Horizontal"

HorizontalAlignment="Right"

Margin="0 0 10 0">

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding Randomize}"

VerticalAlignment="Center"

ToolTip="Генерировать результат"

Grid.Column="3"

Width="30"

Height="30"

Padding="0"

Click="ClearFocus">

<md:PackIcon

Width="24"

Height="24"

Kind="ProgressDownload"/>

</Button>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding AddFaker}"

VerticalAlignment="Top"

ToolTip="Добавить параметр"

Height="30"

Width="30"

Padding="0" >

<md:PackIcon

Width="24"

Height="24"

Kind="AddBox"/>

</Button>

</StackPanel>

<Grid

Margin="20 0 10 0"

Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="120"/>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<ComboBox

ItemsSource="{Binding Templates, Mode=OneWay}"

Text="{Binding Template, Mode=OneWayToSource}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

md:HintAssist.Hint="Шаблон генерации"

IsEditable="True"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

<TextBox

Text="{Binding Count, Mode=TwoWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

md:HintAssist.Hint="Количество"

Margin="10 0"

Grid.Column="1"

KeyDown="LostFocusOnEnter"/>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding SaveTemplate}"

VerticalAlignment="Bottom"

ToolTip="Сохранить шаблон"

Grid.Column="2"

Width="30"

Height="30"

Padding="0"

Click="ClearFocus">

<md:PackIcon

Width="22"

Height="22"

Kind="Download"/>

</Button>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveTemplate}"

VerticalAlignment="Bottom"

ToolTip="Удалить шаблон"

Grid.Column="3"

Width="32"

Padding="0"

Click="ClearFocus">

<md:PackIcon

Width="22"

Height="22"

Kind="Close"/>

</Button>

</Grid>

<ScrollViewer

HorizontalScrollBarVisibility="Auto"

VerticalScrollBarVisibility="Auto"

Padding="20 5 10 5"

Grid.Row="2">

<ItemsControl

ItemsSource="{Binding Fakers, Mode=OneWay}"

VerticalAlignment="Top">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid

Margin="0 0 5 0"

Width="195">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<ComboBox

ItemsSource="{Binding DefaultTemplates, Mode=OneTime}"

Text="{Binding TemplateText, Mode=TwoWay}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

DisplayMemberPath="Key"

md:HintAssist.Hint="Типовой шаблон"

FontSize="14"/>

<TextBox

Text="{Binding Template, Mode=TwoWay, UpdateSourceTrigger=LostFocus}"

Grid.ColumnSpan="3"

Grid.Row="1"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

<CheckBox

IsChecked="{Binding IsUnique, Mode=TwoWay}"

Style="{DynamicResource MaterialDesignCheckBox}"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Grid.Column="1"

ToolTip="IsUnique"/>

<Button

Foreground="{DynamicResource MaterialDesignBodyLight}"

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveFaker}"

ToolTip="Удалить"

Grid.Column="2"

Padding="0"

Height="26"

Width="26">

<md:PackIcon

Height="20"

Width="20"

Kind="Close"/>

</Button>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel

Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</ScrollViewer>

<!--<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="10"/>

<RowDefinition Height="80"/>

<RowDefinition Height="10"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="600"/>

<ColumnDefinition Width="10"/>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<ComboBox

ItemsSource="{Binding Templates, Mode=OneTime}"

Text="{Binding Template, Mode=OneWayToSource}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

md:HintAssist.Hint="Шаблон генерации"

IsEditable="True"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding SaveTemplate}"

VerticalAlignment="Center"

ToolTip="Сохранить шаблон"

Grid.Column="2"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="Download"/>

</Button>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveTemplate}"

VerticalAlignment="Center"

ToolTip="Удалить шаблон"

Grid.Column="3"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="Close"/>

</Button>

</Grid>

<StackPanel

Orientation="Horizontal"

Grid.Row="2">

<ItemsControl

ItemsSource="{Binding Fakers, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid

Margin="0 0 5 0"

Width="195">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<ComboBox

ItemsSource="{Binding DefaultTemplates, Mode=OneTime}"

Text="{Binding TemplateText, Mode=TwoWay}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

DisplayMemberPath="Key"

md:HintAssist.Hint="Типовой шаблон"

FontSize="14"/>

<TextBox

Text="{Binding Template, Mode=TwoWay, UpdateSourceTrigger=LostFocus}"

Grid.ColumnSpan="3"

Grid.Row="1"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

<CheckBox

IsChecked="{Binding IsUnique, Mode=TwoWay}"

Style="{DynamicResource MaterialDesignCheckBox}"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Grid.Column="1"

ToolTip="IsUnique"/>

<Button

Foreground="{DynamicResource MaterialDesignBodyLight}"

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveFaker}"

ToolTip="Удалить"

Grid.Column="2"

Padding="0"

Height="26"

Width="26">

<md:PackIcon

Height="20"

Width="20"

Kind="Close"/>

</Button>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel

Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding AddFaker}"

VerticalAlignment="Top"

ToolTip="Добавить"

Margin="-6 0 0 0"

Width="32"

Padding="0" >

<md:PackIcon

Width="24"

Height="24"

Kind="AddBox"/>

</Button>

</StackPanel>

<Grid

Grid.Row="4">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="600"/>

<ColumnDefinition Width="10"/>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<TextBlock

Text="{Binding TestResult, Mode=OneWay}"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveTemplate}"

VerticalAlignment="Center"

ToolTip="Генерировать"

Grid.Column="3"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="ProgressDownload"/>

</Button>

</Grid>-->

</Grid>

</UserControl>

46. FakerVM.cs. Осуществляет генерации записей по шаблону с случайными значениями.

using System.Collections.ObjectModel;

using System.Text;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using Explorer;

using ScriptGen.Common;

using ScriptGen.Faker.Common;

using ScriptGen.Faker.Model;

using ScriptGen.ViewModel;

namespace ScriptGen.Faker.ViewModel

{

/// <summary>

/// Осуществляет генерации записей по шаблону с случайными значениями

/// </summary>

public class FakerVM : ViewModelBase

{

private int count = 1;

private string template = "";

private FileDialog fileDialog

= new FileDialog();

public FakerVM()

{

if (FileWork.BinaryLoad(@"Data\faker.bin", out string errorMessage, out object loadTemplates))

Templates = (ObservableCollection<string>)loadTemplates;

FakeWord.Load();

FakeName.Load();

FakeSurname.Load();

AddFaker = new SimpleCommand(NewFaker);

SaveTemplate = new SimpleCommand(() =>

{

if (!Templates.Contains(Template))

{

Templates.Add(Template);

StatusBarManagerVM.Message("Шаблон сохранён!", Template);

}

});

RemoveTemplate = new SimpleCommand(() =>

{

if (Templates.Contains(Template))

{

StatusBarManagerVM.Message("Шаблон удалён!", Template);

Templates.Remove(Template);

}

});

Randomize = new SimpleCommand(Generate, () => Fakers.Count > 0 && Template.Length > 0);

}

public SimpleCommand AddFaker { get; }

public SimpleCommand SaveTemplate { get; }

public SimpleCommand RemoveTemplate { get; }

public SimpleCommand Randomize { get; }

public static ObservableCollection<string> Templates { get; private set; }

public static void SaveDefaultTemplates()

=> FileWork.BinarySave(Templates, @"Data\faker.bin", out string errorMessage);

public ObservableCollection<FakerM> Fakers { get; }

= new ObservableCollection<FakerM>();

public string Count

{

get => count.ToString();

set

{

if (int.TryParse(value, out int v) && v > 0 && v < 101)

count = v;

OnPropertyChanged();

}

}

public string Template

{

get => template;

set

{

template = value;

OnPropertyChanged();

}

}

private void NewFaker()

{

FakerM faker = new FakerM();

faker.OnRemoveFaker += (f) => Fakers.Remove(f);

Fakers.Add(faker);

}

private void Generate()

{

StringBuilder result = new StringBuilder();

for (int i = 0; i < Fakers.Count; i++)

Fakers[i].Reset();

for (int i = 0; i < count; i++)

{

string templateResult = GetTemplateResult();

if (!string.IsNullOrEmpty(templateResult))

result.AppendLine(templateResult);

else

{

StatusBarManagerVM.Info("Не удалось получить уникальное значение!", $"Завершено {i} из {count} итерация!",

"сохранить", new SimpleCommand(() => Save(result)));

return;

}

}

Save(result);

}

private string GetTemplateResult()

{

string result = Template;

for (int i = 0; i < Fakers.Count; i++)

if (Fakers[i].Randomize(out string value))

result = result.Replace($"@v{i + 1}", value);

else

return null;

return result;

}

private void Save(StringBuilder result)

{

if (fileDialog.Open(ExplorerSelectType.All, ExplorerType.Save, ".txt", ".sql"))

FileWork.TextSave(result, fileDialog.SelectPath, "faker", ".txt");

}

}

}

47. FillHelper.cs. Осуществляет выделение ключевых слов и параметров из текста.

using System.Linq;

namespace ScriptGen.Generator

{

/// <summary>

/// Осуществляет выделение ключевых слов и параметров из текста

/// </summary>

public static class FillHelper

{

public static string GetSpace(string code)

=> new string(code.TakeWhile(c => !char.IsLetter(c)).ToArray());

public static string GetWord(string code)

=> new string(code.TakeWhile(c => char.IsLetter(c)).ToArray());

public static string GetConstant(string code)

{

string result = new string(code.TakeWhile(c => char.IsLetterOrDigit(c) || ".?\\[]{".Contains(c)).ToArray());

return code.Length > result.Length && code[result.Length] == '}'

? result + '}'

: result;

}

public static string GetParameter(string code)

=> GetBlock(code, '(', ')');

public static string GetBlock(string code)

=> GetBlock(code, '{', '}');

private static string GetBlock(string code, char startDivider, char endDivider)

{

if (code.Contains(startDivider) && code.Contains(endDivider))

{

code = code.Substring(code.IndexOf(startDivider));

int openCount = 0;

int closeCount = 0;

for (int i = 0; i < code.Length; i++)

{

if (code[i] == startDivider)

openCount++;

else if (code[i] == endDivider)

closeCount++;

if (openCount == closeCount && openCount != 0)

return code.Substring(1, i - 1);

}

}

return "";

}

public static int GetConstantParameter(string constant, out string newConstant)

{

newConstant = constant;

int startIndex = 0;

int endIndex = 0;

if (constant.Contains('{') && constant.Contains('}'))

{

startIndex = constant.IndexOf('{');

endIndex = constant.IndexOf('}');

}

else if (constant.Contains('[') && constant.Contains(']'))

{

startIndex = constant.IndexOf('[');

endIndex = constant.IndexOf(']');

}

if (startIndex == 0 || endIndex == 0)

return 0;

if (int.TryParse(constant.Substring(startIndex + 1, endIndex - startIndex - 1), out int parameter))

{

newConstant = constant.Substring(0, startIndex) + constant.Substring(endIndex + 1);

return parameter;

}

else

return 0;

}

}

}

48. ScriptGenerator.cs. Осуществляет генерацию по шаблону.

using System.Collections.Generic;

using System.Text;

using System.Linq;

using ScriptGen.Generator.Keywords;

using ScriptGenPlugin.Model;

using ScriptGen.Common;

namespace ScriptGen.Generator

{

/// <summary>

/// Осуществляет генерацию по шаблону

/// </summary>

public static class ScriptGenerator

{

private static StringBuilder sqlResult;

private static StringBuilder programmingResult;

private static List<string> wasTableNames;

public static void Generate(TableInfo[] tables, string savePath)

{

Reset();

TemplateFiller.Tables = tables;

for (int i = 0; i < tables.Length; i++)

NextTable(tables[i]);

Save(savePath);

}

private static void NextTable(TableInfo table)

{

if (wasTableNames.Contains(table.Name) || table == null)

return;

foreach (FieldInfo field in table.Fields.Where(f => f.Type == FieldType.FK))

NextTable(field.RefTable);

sqlResult.Append(TemplateFiller.Fill(table, Info.TemplateSQL));

programmingResult.Append(TemplateFiller.Fill(table, Info.TemplateProgramming));

wasTableNames.Add(table.Name);

}

private static void Reset()

{

Once.Reset();

wasTableNames = new List<string>();

sqlResult = new StringBuilder();

programmingResult = new StringBuilder();

}

private static void Save(string path)

{

FileWork.TextSave(sqlResult, path, "script", ".sql");

FileWork.TextSave(programmingResult, path, "script", ".txt");

}

}

}

49. TemplateFiller.cs. Осуществляет нормализацию шаблона.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using ScriptGen.Generator.Keywords;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Generator

{

/// <summary>

/// Осуществляет нормализацию шаблона

/// </summary>

public static class TemplateFiller

{

public static TableInfo[] Tables;

public static TableInfo Table;

private static string template;

private static StringBuilder result;

private static readonly Dictionary<string, IKeyword> keywords

= new Dictionary<string, IKeyword>()

{

{ "write", new Write() },

{ "if", new If() },

{ "for", new For() },

{ "rec", new Rec() },

{ "removelast", new RemoveLast() },

{ "replace", new Replace() },

{ "once", new Once() },

{ "set", new Set() }

};

private static readonly Dictionary<string, Func<int, string>> constants

= new Dictionary<string, Func<int, string>>()

{

{ "tab", (param) => Pad(" ", param) },

{ "nl", (param) => Pad("\n", param) },

{ "table.id.name", (\_) => constants["field.name"](0) },

{ "table.id.fullname", (\_) => constants["field.fullname"](0) },

{ "table.id.logicalname", (\_) => constants["field.logicalname"](0) },

{ "table.id.programmingname", (\_) => constants["field.programmingname"](0) },

{ "table.id.datatype", (\_) => constants["field.datatype"](0) },

{ "table.id.programmingtype", (\_) => constants["field.programmingtype"](0) },

{ "table.name", (\_) => Table.Name },

{ "table.logicalname", (\_) => Table.LogicalName },

{ "table.programmingname", (\_) => Table.ProgrammingName },

{ "field.name", (param) => Table.Fields[param].Name },

{ "field.fullname", (param) => $"{Table.Name}.{Table.Fields[param].Name}" },

{ "field.logicalname", (param) => Table.Fields[param].LogicalName },

{ "field.programmingname", (param) => Table.Fields[param].ProgrammingName },

{ "field.datatype", (param) => Table.Fields[param].DataType },

{ "field.programmingtype", (param) => Table.Fields[param].ProgrammingType },

{ "field.reftable.id.name", (param) => Table.Fields[param].RefTable.Fields[0].Name },

{ "field.reftable.id.fullname", (param) => $"{Table.Fields[param].RefTable.Name}.{Table.Fields[param].RefTable.Fields[0].Name}" },

{ "field.reftable.id.logicalname", (param) => Table.Fields[param].RefTable.Fields[0].LogicalName },

{ "field.reftable.id.programmingname", (param) => Table.Fields[param].RefTable.Fields[0].ProgrammingName },

{ "field.reftable.id.datatype", (param) => Table.Fields[param].RefTable.Fields[0].DataType },

{ "field.reftable.id.programmingtype", (param) => Table.Fields[param].RefTable.Fields[0].ProgrammingType },

{ "field.reftable.name", (param) => Table.Fields[param].RefTable.Name },

{ "field.reftable.logicalname", (param) => Table.Fields[param].RefTable.LogicalName },

{ "field.reftable.programmingname", (param) => Table.Fields[param].RefTable.ProgrammingName },

};

public static void Append(string content)

=> result.Append(content);

public static void Replace(string oldValue, string newValue)

=> result.Replace(oldValue, newValue);

public static void RemoveLast(int length)

=> result.Remove(result.Length - length, length);

public static string Fill(TableInfo table, string template)

{

Rec.Reset();

Set.Reset();

TemplateFiller.template = template;

result = new StringBuilder();

Table = table;

NextBlock();

return result.ToString();

}

public static bool KeywordInvoke(string keyword, string code)

{

if (keywords.ContainsKey(keyword))

{

keywords[keyword].Do(code);

return true;

}

return false;

}

public static string GetConstantValue(string word)

{

if (word.Contains('?'))

return GetNotNullConstantValue(word);

int param = FillHelper.GetConstantParameter(word.ToLower(), out string constant);

return constants.ContainsKey(constant)

? constants[constant].Invoke(param)

: word;

}

private static string GetNotNullConstantValue(string word)

{

string[] constants = word.Split('?');

return constants.Length == 2

? GetNonNullString(GetConstantValue(constants[0]), GetConstantValue(constants[1]))

: word;

}

private static void NextBlock()

{

if (template.Contains("{{") && template.Contains("}}"))

{

int index = template.IndexOf("{{");

result.Append(template.Substring(0, index));

template = template.Substring(index + 2);

index = template.IndexOf("}}");

string code = NormalizeCode(template.Substring(0, index));

template = template.Substring(index + 2);

KeywordInvoke(FillHelper.GetWord(code), code);

NextBlock();

}

else

result.Append(template);

}

private static string NormalizeCode(string code)

{

string result = "";

while (code.Contains("write"))

{

int index = code.IndexOf("write");

result += Regex.Replace(code.Substring(0, index), @"\s+", "").ToLower();

result = Regex.Replace(result, @"\[\d+\]", "");

code = code.Substring(index + 5);

if (code.Contains('(') && code.Contains(')'))

{

string block = FillHelper.GetParameter(code);

result += $"write({block})";

code = code.Substring(block.Length + 2);

}

}

result += Regex.Replace(code, @"\s+", "").ToLower();

return Regex.Replace(result, @"\[\d+\]", "");

}

private static string GetNonNullString(string content, string alternativeContent)

=> string.IsNullOrWhiteSpace(content) ? alternativeContent : content;

private static string Pad(string padding, int count)

{

string result = "";

for (int i = 0; i < count; i++)

result += padding;

return result;

}

}

}

50. For.cs. Ключевое слово цикла.

using System;

using System.Collections.Generic;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово цикла

/// </summary>

public class For : IKeyword

{

private readonly static Dictionary<string, Action<string>> parameters

= new Dictionary<string, Action<string>>()

{

{ "table.fields", (s) => FieldFor(s) },

{ "table.reftables", (s) => RefTableFor(s) },

{ "tables", (s) => AllTableFor(s) }

};

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 5);

string block = FillHelper.GetBlock(code);

code = code.Substring(block.Length + 2);

if (parameters.ContainsKey(parameter))

parameters[parameter].Invoke(block);

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

private static void FieldFor(string block)

{

for (int i = 0; i < TemplateFiller.Table.Fields.Length; i++)

TemplateFiller.KeywordInvoke(FillHelper.GetWord(block), block.Replace("field", $"field[{i}]"));

}

private static void RefTableFor(string block)

{

TableInfo saveTable = TemplateFiller.Table;

for (int i = 0; i < TemplateFiller.Table.RefTables.Length; i++)

{

TemplateFiller.Table = saveTable.RefTables[i];

TemplateFiller.KeywordInvoke(FillHelper.GetWord(block), block);

}

TemplateFiller.Table = saveTable;

}

private static void AllTableFor(string block)

{

TableInfo saveTable = TemplateFiller.Table;

for (int i = 0; i < TemplateFiller.Tables.Length; i++)

{

TemplateFiller.Table = TemplateFiller.Tables[i];

TemplateFiller.KeywordInvoke(FillHelper.GetWord(block), block);

}

TemplateFiller.Table = saveTable;

}

}

}

51. If.cs. Ключевое слово условия.

using System;

using System.Linq;

using System.Collections.Generic;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово условия

/// </summary>

public class If : IKeyword

{

private readonly static Dictionary<string, Func<int, bool>> conditions

= new Dictionary<string, Func<int, bool>>()

{

{ "true", (\_) => true },

{ "field.isfk", (param) => TemplateFiller.Table.Fields[param].Type == FieldType.FK },

{ "field.ispk", (param) => TemplateFiller.Table.Fields[param].Type == FieldType.PK },

{ "field.isany", (param) => TemplateFiller.Table.Fields[param].Type == FieldType.Any },

{ "field.isnull", (param) => TemplateFiller.Table.Fields[param].IsNull },

{ "field.isunique", (param) => TemplateFiller.Table.Fields[param].IsUnique }

};

public void Do(string code)

{

List<string> parameters = new List<string>() { FillHelper.GetParameter(code) };

List<string> blocks = new List<string>() { FillHelper.GetBlock(code) };

code = code.Substring(parameters[0].Length + blocks[0].Length + 6);

while (true)

{

if (code.StartsWith("elseif"))

{

parameters.Add(FillHelper.GetParameter(code));

blocks.Add(FillHelper.GetBlock(code));

code = code.Substring(parameters.Last().Length + blocks.Last().Length + 10);

}

else if (code.StartsWith("else"))

{

parameters.Add("true");

blocks.Add(FillHelper.GetBlock(code));

code = code.Substring(blocks.Last().Length + 6);

}

else

break;

}

Do(parameters.ToArray(), blocks.ToArray());

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

private void Do(string[] parameters, string[] blocks)

{

if (parameters.Length != blocks.Length)

return;

for (int i = 0; i < parameters.Length; i++)

if (ConditionInvoke(parameters[i]))

{

TemplateFiller.KeywordInvoke(FillHelper.GetWord(blocks[i]), blocks[i]);

break;

}

}

private bool ConditionInvoke(string condition)

{

bool isInverse = false;

if (!string.IsNullOrEmpty(condition) && condition[0] == '!')

{

condition = condition.Substring(1);

isInverse = true;

}

int param = FillHelper.GetConstantParameter(condition, out condition);

if (conditions.ContainsKey(condition))

{

bool result = conditions[condition].Invoke(param);

return isInverse ? !result : result;

}

else

return false;

}

}

}

52. Once.cs. Ключевое слово блока кода для выполнения один раз за генерацию.

using ScriptGen.Interface;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово блока кода для выполнения один раз за генерацию

/// </summary>

public class Once : IKeyword

{

private static bool was;

public void Do(string code)

{

string block = FillHelper.GetBlock(code.Substring(4));

code = code.Substring(block.Length + 6);

if (!was)

{

TemplateFiller.KeywordInvoke(FillHelper.GetWord(block), block);

was = true;

}

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

public static void Reset()

=> was = false;

}

}

53. Rec.cs. Ключевое слово рекурсии.

using System.Collections.Generic;

using System.Linq;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово рекурсии

/// </summary>

public class Rec : IKeyword

{

private static string recBlock;

private static List<TableInfo> wasTable;

private static List<string> wasTableNames;

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 5);

if (parameter == "start")

RecStart(code);

else

RecInvoke(GetRefTable(parameter));

}

private void RecStart(string code)

{

wasTable = new List<TableInfo>();

wasTableNames = new List<string>();

recBlock = FillHelper.GetBlock(code);

code = code.Substring(recBlock.Length + 2);

TemplateFiller.KeywordInvoke(FillHelper.GetWord(recBlock), recBlock);

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

private void RecInvoke(TableInfo table)

{

if (string.IsNullOrEmpty(recBlock) || table == null || wasTableNames.Contains(table.Name))

return;

wasTable.Add(TemplateFiller.Table);

wasTableNames.Add(TemplateFiller.Table.Name);

TemplateFiller.Table = table;

TemplateFiller.KeywordInvoke(FillHelper.GetWord(recBlock), recBlock);

TemplateFiller.Table = wasTable.Last();

wasTable.RemoveAt(wasTable.Count - 1);

}

private TableInfo GetRefTable(string command)

{

int param = FillHelper.GetConstantParameter(command, out command);

return TemplateFiller.Table.Fields[param].RefTable;

}

public static void Reset()

{

recBlock = "";

wasTable = null;

wasTableNames = null;

}

}

}

54. RemoveLast.cs. Ключевое слово удаления последних символов.

using ScriptGen.Interface;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово удаления последних символов

/// </summary>

public class RemoveLast : IKeyword

{

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 12);

if (int.TryParse(parameter, out int length))

TemplateFiller.RemoveLast(length);

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

}

}

55. Replace.cs. Ключевое слово замены слов.

using ScriptGen.Interface;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово замены слов

/// </summary>

public class Replace : IKeyword

{

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 9);

string[] parameters = parameter.Split(',');

if (parameters.Length == 2)

TemplateFiller.Replace(parameters[0], parameters[1]);

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

}

}

56. Set.cs. Ключевое слово запоминания значения.

using System.Collections.Generic;

using ScriptGen.Interface;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово запоминания значения

/// </summary>

public class Set : IKeyword

{

private static Dictionary<string, string> variables

= new Dictionary<string, string>() { };

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 5);

string[] parameters = parameter.Split(',');

if (parameters.Length == 2)

{

parameters[0] = $"get[{parameters[0]}]";

if (variables.ContainsKey(parameters[0]))

variables[parameters[0]] = TemplateFiller.GetConstantValue(parameters[1]);

else

variables.Add(parameters[0], TemplateFiller.GetConstantValue(parameters[1]));

}

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

public static string GetValue(string name)

=> variables.ContainsKey(name) ? variables[name] : name;

public static void Reset()

=> variables = new Dictionary<string, string>();

}

}

57. Write.cs. Ключевое слово вывода информации.

using System;

using System.Collections.Generic;

using ScriptGen.Interface;

namespace ScriptGen.Generator.Keywords

{

/// <summary>

/// Ключевое слово вывода информации

/// </summary>

public class Write : IKeyword

{

private static readonly Dictionary<string, Func<string, string>> constants

= new Dictionary<string, Func<string, string>>()

{

{ "\\fl", (word) => $"{char.ToLower(word[0])}{word.Substring(1)}" },

{ "\\fu", (word) => $"{char.ToUpper(word[0])}{word.Substring(1)}" }

};

public void Do(string code)

{

string parameter = FillHelper.GetParameter(code);

code = code.Substring(parameter.Length + 7);

while (!string.IsNullOrEmpty(parameter))

{

string word = FillHelper.GetConstant(parameter);

if (string.IsNullOrEmpty(word))

word = FillHelper.GetSpace(parameter);

if (word.StartsWith("get"))

TemplateFiller.Append(Set.GetValue(word));

else

TemplateFiller.Append(GetFormatConstantValue(word));

parameter = parameter.Substring(word.Length);

}

TemplateFiller.KeywordInvoke(FillHelper.GetWord(code), code);

}

private string GetFormatConstantValue(string word)

{

if (word.EndsWith("\\fl") || word.EndsWith("\\fu"))

{

string parameter = word.Substring(word.Length - 3);

word = TemplateFiller.GetConstantValue(word.Substring(0, word.Length - 3));

if (word.Length > 1)

return constants[parameter](word);

}

return TemplateFiller.GetConstantValue(word);

}

}

}

58. IElement.cs. Графический элемент приложения.

using System.ComponentModel;

using BaseMVVM.Command;

using ScriptGen.Common;

namespace ScriptGen.Interface

{

/// <summary>

/// Графический элемент приложения

/// </summary>

public interface IElement : IError, INotifyPropertyChanged

{

/// <summary>

/// Команда нажатия "Enter" в поле ввода для названия

/// </summary>

SimpleCommand NameEnter { get; set; }

/// <summary>

/// Команда нажатия "Up" в поле ввода для названия

/// </summary>

SimpleCommand NameUp { get; set; }

/// <summary>

/// Команда нажатия "Down" в поле ввода для названия

/// </summary>

SimpleCommand NameDown { get; set; }

/// <summary>

/// Команда потери элементом фокуса

/// </summary>

SimpleCommand LostFocus { get; set; }

/// <summary>

/// Команда загрузки элемента

/// </summary>

SimpleCommand Loaded { get; set; }

/// <summary>

/// Команда удаления элемента

/// </summary>

SimpleCommand RemoveElement { get; set; }

/// <summary>

/// Название

/// </summary>

string Name { get; set; }

/// <summary>

/// Логическое название

/// </summary>

string LogicalName { get; set; }

/// <summary>

/// Название для модели для языков программирования

/// </summary>

string ProgrammingName { get; set; }

/// <summary>

/// Тип модели (физическая/логическая/программирование)

/// </summary>

ModelType TypeModel { get; set; }

/// <summary>

/// Определяет элемент с фокусом

/// </summary>

Focuses FocusedElement { get; set; }

/// <summary>

/// Устанавливает фокус элемента в зависимости от его свойств

/// </summary>

void SetFocus();

}

}

59. IError.cs. Управление ошибкой элемента.

namespace ScriptGen.Interface

{

/// <summary>

/// Управление ошибкой элемента

/// </summary>

public interface IError

{

/// <summary>

/// true/false существует ли ошибка у элемента

/// </summary>

bool IsError { get; set; }

/// <summary>

/// Сообщение об ошибке

/// </summary>

string Message { get; set; }

/// <summary>

/// Устанавливает ошибку с переданным сообщением

/// </summary>

/// <param name="message">Сообщение</param>

void SetError(string message = null);

}

}

60. IFaker.cs. Интерфейс для генератора случайных значений.

namespace ScriptGen.Interface

{

/// <summary>

/// Интерфейс для генератора случайных значений

/// </summary>

public interface IFaker

{

/// <summary>

/// Генерирует count случайных значений

/// </summary>

/// <param name="count">Количество</param>

/// <param name="language">Язык</param>

/// <returns>Случайные значения</returns>

string Next(int count, char language);

/// <summary>

/// Генерирует от minCount до maxCount случайных значений

/// </summary>

/// <param name="minCount">Минимальное количество слов</param>

/// <param name="maxCount">Максимальное количество слов</param>

/// <param name="language">Язык</param>

/// <returns>Случайные слов</returns>

string Next(int minCount, int maxCount, char language);

/// <summary>

/// Возвращает случайное значение из представленных

/// </summary>

/// <param name="values">Возможные значения</param>

/// <returns>Случайное слово</returns>

string Next(string[] values);

}

}

61. IField.cs. Поле таблицы.

using System;

using ScriptGen.Common.SaveModel;

using ScriptGenPlugin.Model;

namespace ScriptGen.Interface

{

/// <summary>

/// Поле таблицы

/// </summary>

public interface IField : IElement

{

/// <summary>

/// Событие удаления поля

/// </summary>

event Action<IField> OnRemoveField;

/// <summary>

/// Вызывается при действиях, осуществляющих переключение на следующее поле

/// </summary>

Action NextField { get; set; }

/// <summary>

/// Вызывается при действиях, осуществляющих переключение на предыдущее поле

/// </summary>

Action PrevField { get; set; }

/// <summary>

/// Тип поля

/// </summary>

FieldType Type { get; set; }

/// <summary>

/// SQL тип данных поля

/// </summary>

string DataType { get; set; }

/// <summary>

/// Тип данных поля для языков программирования

/// </summary>

string ProgrammingType { get; set; }

/// <summary>

/// true/false может ли поле быть равно null

/// </summary>

bool IsNull { get; set; }

/// <summary>

/// true/false является ли поле уникальным

/// </summary>

bool IsUnique { get; set; }

/// <summary>

/// Информация о таблице на которую ссылается поле (только для FK)

/// </summary>

ITable RefTable { get; set; }

/// <summary>

/// Загружает данные поля

/// </summary>

/// <param name="fieldSave">Сохранённая информация</param>

void Load(FieldSave fieldSave);

}

}

62. IKeyword.cs. Ключевого слово для выстроенного языка шаблонизации.

namespace ScriptGen.Interface

{

/// <summary>

/// Ключевого слово для выстроенного языка шаблонизации

/// </summary>

public interface IKeyword

{

/// <summary>

/// Выполняет заданное действие для ключевого слова

/// </summary>

/// <param name="code">Код</param>

void Do(string code);

}

}

63. ILine.cs. Линия связи между таблицами.

using System;

namespace ScriptGen.Interface

{

/// <summary>

/// Линия связи между таблицами

/// </summary>

public interface ILine

{

/// <summary>

/// Вызывается при удалении линии

/// </summary>

event Action<ILine> OnRemoveLine;

/// <summary>

/// Координата X для начальной точки линии

/// </summary>

double SourceX { get; set; }

/// <summary>

/// Координата X для конечной точки линии

/// </summary>

double TargetX { get; set; }

/// <summary>

/// Координата Y для начальной точки линии

/// </summary>

double SourceY { get; set; }

/// <summary>

/// Координата Y для конечной точки линии

/// </summary>

double TargetY { get; set; }

/// <summary>

/// Координата X для разделительной линии

/// </summary>

double ConnectionX { get; set; }

/// <summary>

/// Таблица, являющаяся источником

/// </summary>

ITable Source { get; }

/// <summary>

/// Таблица, являющаяся целью

/// </summary>

ITable Target { get; }

/// <summary>

/// Поле FK, с которым связанна линия

/// </summary>

IField Field { get; }

}

}

64. IMove.cs. Движение элемента.

using System;

using System.Windows;

namespace ScriptGen.Interface

{

/// <summary>

/// Движение элемента

/// </summary>

public interface IMove

{

/// <summary>

/// Событие изменения ширины элемента

/// </summary>

event Action<double, IMove> OnWidthChanged;

/// <summary>

/// Событие изменения высоты элемента

/// </summary>

event Action<double, IMove> OnHeightChanged;

/// <summary>

/// Событие изменения положения элемента

/// </summary>

event Action<Thickness, IMove> OnMarginChanged;

/// <summary>

/// true/false можно ли изменить ширину

/// </summary>

Func<double, IMove, bool> CanWidthChange { get; set; }

/// <summary>

/// true/false можно ли изменить высоту

/// </summary>

Func<double, IMove, bool> CanHeightChange { get; set; }

/// <summary>

/// Функция, определяющая можно ли изменить положение элемента по горизонтали

/// </summary>

Func<double, IMove, bool> CanHorizontalChange { get; set; }

/// <summary>

/// Функция, определяющая можно ли изменить положение элемента по вертикали

/// </summary>

Func<double, IMove, bool> CanVerticalChange { get; set; }

/// <summary>

/// Ширина элемента

/// </summary>

double Width { get; set; }

/// <summary>

/// Высота элемента

/// </summary>

double Height { get; set; }

/// <summary>

/// Положение элемента

/// </summary>

Thickness Margin { get; set; }

}

}

65. IPage.cs. Страница приложения.

using System;

using System.Collections.ObjectModel;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

namespace ScriptGen.Interface

{

/// <summary>

/// Страница приложения

/// </summary>

public interface IPage : IError

{

/// <summary>

/// Вызывается при удалении страницы

/// </summary>

event Action<IPage> OnRemovePage;

/// <summary>

/// Вызывается при выделении страницы

/// </summary>

event Action<IPage> OnSelectPage;

/// <summary>

/// Название

/// </summary>

string Name { get; set; }

/// <summary>

/// true/false выбрана ли страница

/// </summary>

bool IsSelect { get; set; }

/// <summary>

/// Выделенная таблица

/// </summary>

ITable SelectTable { get; set; }

/// <summary>

/// Тип модели

/// </summary>

ModelType TypeModel { get; set; }

/// <summary>

/// Смещения по вертикали

/// </summary>

double ScrollVertical { get; set; }

/// <summary>

/// Смещения по горизонтали

/// </summary>

double ScrollHorizontal { get; set; }

/// <summary>

/// Ширина

/// </summary>

double ActualWidth { get; set; }

/// <summary>

/// Высота

/// </summary>

double ActualHeight { get; set; }

/// <summary>

/// Таблицы

/// </summary>

ObservableCollection<ITable> Tables { get; }

/// <summary>

/// Линии

/// </summary>

ObservableCollection<ILine> Lines { get; }

/// <summary>

/// Добавляет таблицу

/// </summary>

/// <param name="table">Таблица</param>

void AddTable(ITable table);

/// <summary>

/// Удаляет таблицу

/// </summary>

/// <param name="table">Таблица</param>

void RemoveTable(ITable table);

/// <summary>

/// Загружает данные страницы

/// </summary>

/// <param name="dataBaseSave">Сохранённая информация</param>

void Load(DataBaseSave dataBaseSave);

}

}

66. ITable.cs. Таблица.

using System;

using System.Collections.ObjectModel;

using ScriptGen.Common.SaveModel;

namespace ScriptGen.Interface

{

/// <summary>

/// Таблица

/// </summary>

public interface ITable : IElement, IMove

{

/// <summary>

/// Происходит при удалении таблицы

/// </summary>

event Action<ITable> OnRemoveTable;

/// <summary>

/// Происходит при выделении таблицы

/// </summary>

event Action<ITable> OnSelectTable;

/// <summary>

/// Id

/// </summary>

int Id { get; set; }

/// <summary>

/// true/false выделена ли таблица

/// </summary>

bool IsSelect { get; set; }

/// <summary>

/// Получает стандартное поле для добавления

/// </summary>

Func<IField> GetNewField { get; set; }

/// <summary>

/// Поля

/// </summary>

ObservableCollection<IField> Fields { get; }

/// <summary>

/// Добавляет поле

/// </summary>

/// <param name="field">Поле</param>

void AddField(IField field);

/// <summary>

/// Удаляет поле

/// </summary>

/// <param name="field">Поле</param>

void RemoveField(IField field);

/// <summary>

/// Загружает данные таблицы

/// </summary>

/// <param name="tableSave">Сохранённая информация</param>

void Load(TableSave tableSave);

}

}

67. DataBaseM.cs. Модель базы данных.

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using ScriptGen.Abstraction;

using ScriptGen.Interface;

namespace ScriptGen.Model

{

/// <summary>

/// Модель базы данных

/// </summary>

public class DataBaseM : UniqueErrorBase

{

public string Name;

public bool IsError;

public ObservableCollection<ITable> Tables

= new ObservableCollection<ITable>();

public ObservableCollection<ILine> Lines

= new ObservableCollection<ILine>();

public void SetTableError()

{

string message = null;

IEnumerable<string> names = Tables.Select(x => x.Name);

IEnumerable<string> logicalNames = Tables.Select(x => x.LogicalName);

IEnumerable<string> linqNames = Tables.Select(x => x.ProgrammingName);

for (int i = 0; i < Tables.Count; i++)

{

SetError(ref message, GetUniqueError(names, i, "Название должно быть уникальным!"));

SetError(ref message, GetUniqueError(logicalNames, i, "Логическое название должно быть уникальным!"));

SetError(ref message, GetUniqueError(linqNames, i, "Название для языков программирования должно быть уникальным!"));

Tables[i].SetError(message);

message = null;

}

}

}

}

68. ElementM.cs. Модель элемента.

using ScriptGen.Common;

namespace ScriptGen.Model

{

/// <summary>

/// Модель элемента

/// </summary>

public class ElementM

{

public string Name;

public string LogicalName;

public string ProgrammingName;

public bool IsError;

public string Message = null;

public string GetError()

{

string message = Verification.ConcatString(GetNameError(), GetLogicalNameError());

return Verification.ConcatString(message, GetProgrammingNameError());

}

private string GetNameError()

=> string.IsNullOrWhiteSpace(Name)

? "Название не может быть пустым!"

: !Verification.IsOnlyEnLetterOrDigitOrAllowedChar(Name, '\_') || Verification.IsFalseWord(Name)

? "Название содержит недопустимое слово или символ!"

: null;

private string GetLogicalNameError()

=> !string.IsNullOrWhiteSpace(LogicalName) && !Verification.IsOnlyLetterOrAllowedChar(LogicalName, ' ')

? "Логическое название содержит недопустимый символ!"

: null;

private string GetProgrammingNameError()

=> !string.IsNullOrWhiteSpace(ProgrammingName) && !Verification.IsOnlyEnLetter(ProgrammingName)

? "Название для языков программирования содержит недопустимый символ!"

: null;

}

}

69. FieldM.cs. Модель поля таблицы.

using ScriptGen.Common;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.Model

{

/// <summary>

/// Модель поля таблицы

/// </summary>

public struct FieldM

{

public ElementM Element;

public FieldType Type;

public string DataType;

public string ProgrammingType;

public bool IsNull;

public bool IsUnique;

public ITable RefTable;

public string GetError()

{

string message = Verification.ConcatString(Element.GetError(), GetDataTypeError());

return Verification.ConcatString(message, GetProgrammingTypeError());

}

private string GetDataTypeError()

=> Type == FieldType.FK

? null

: string.IsNullOrWhiteSpace(DataType)

? "Тип данных не может быть пустым!"

: !Verification.IsOnlyEnLetterOrDigitOrAllowedChar(DataType, '(', ')') || Verification.IsFalseWord(DataType)

? "Тип данных содержит недопустимое слово или символ!"

: null;

private string GetProgrammingTypeError()

=> Type == FieldType.FK

? null

: !string.IsNullOrWhiteSpace(ProgrammingType) && (!Verification.IsOnlyEnLetter(ProgrammingType) || Verification.IsFalseWord(ProgrammingType))

? "Тип данных для языков программирования содержит недопустимое слово или символ!"

: null;

}

}

70. MessageM.cs. Модель сообщения.

using BaseMVVM.Command;

using MaterialDesignThemes.Wpf;

namespace ScriptGen.Model

{

/// <summary>

/// Модель сообщения

/// </summary>

public struct MessageM

{

public bool IsVisible { get; }

public string Data { get; }

public PackIconKind Kind { get; }

public string ActionTitle { get; }

public SimpleCommand ActionCommand { get; }

public MessageM(string message, PackIconKind kind, string actionTitle = null, SimpleCommand action = null)

{

IsVisible = action != null;

Data = message;

Kind = kind;

ActionTitle = actionTitle;

ActionCommand = action;

}

}

}

71. ScriptGenM.cs. Модель приложения.

using System.Collections.ObjectModel;

using ScriptGen.Common;

using ScriptGen.Interface;

namespace ScriptGen.Model

{

/// <summary>

/// Модель приложения

/// </summary>

public class ScriptGenM

{

public ModelType ModelType;

public IPage SelectPage;

public ObservableCollection<IPage> Pages

= new ObservableCollection<IPage>();

}

}

72. TableM.cs. Модель таблицы.

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows;

using ScriptGen.Abstraction;

using ScriptGen.Interface;

namespace ScriptGen.Model

{

/// <summary>

/// Модель таблицы

/// </summary>

public class TableM : UniqueErrorBase

{

public int Id;

public ElementM Element;

public ObservableCollection<IField> Fields

= new ObservableCollection<IField>();

public double Width;

public double Height;

public Thickness Margin;

public void SetFieldError()

{

string message = null;

IEnumerable<string> names = Fields.Select(x => x.Name);

IEnumerable<string> logicalNames = Fields.Select(x => x.LogicalName);

IEnumerable<string> programmingNames = Fields.Select(x => x.ProgrammingName);

for (int i = 0; i < Fields.Count; i++)

{

SetError(ref message, GetUniqueError(names, i, "Название должно быть уникальным!"));

SetError(ref message, GetUniqueError(logicalNames, i, "Логическое название должно быть уникальным!"));

SetError(ref message, GetUniqueError(programmingNames, i, "Название для языков программирования должно быть уникальным!"));

Fields[i].SetError(message);

message = null;

}

}

}

}

73. AssemblyInfo.cs.

using System.Reflection;

using System.Resources;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using System.Windows;

// Общие сведения об этой сборке предоставляются следующим набором

// набора атрибутов. Измените значения этих атрибутов, чтобы изменить сведения,

// связанные со сборкой.

[assembly: AssemblyTitle("ScriptGen")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("")]

[assembly: AssemblyProduct("ScriptGen")]

[assembly: AssemblyCopyright("Copyright © 2019")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

// Установка значения False для параметра ComVisible делает типы в этой сборке невидимыми

// для компонентов COM. Если необходимо обратиться к типу в этой сборке через

// COM, задайте атрибуту ComVisible значение TRUE для этого типа.

[assembly: ComVisible(false)]

//Чтобы начать создание локализуемых приложений, задайте

//<UICulture>CultureYouAreCodingWith</UICulture> в файле .csproj

//внутри <PropertyGroup>. Например, если используется английский США

//в своих исходных файлах установите <UICulture> в en-US. Затем отмените преобразование в комментарий

//атрибута NeutralResourceLanguage ниже. Обновите "en-US" в

//строка внизу для обеспечения соответствия настройки UICulture в файле проекта.

//[assembly: NeutralResourcesLanguage("en-US", UltimateResourceFallbackLocation.Satellite)]

[assembly: ThemeInfo(

ResourceDictionaryLocation.None, //где расположены словари ресурсов по конкретным тематикам

//(используется, если ресурс не найден на странице,

// или в словарях ресурсов приложения)

ResourceDictionaryLocation.SourceAssembly //где расположен словарь универсальных ресурсов

//(используется, если ресурс не найден на странице,

// в приложении или в каких-либо словарях ресурсов для конкретной темы)

)]

// Сведения о версии сборки состоят из следующих четырех значений:

//

// Основной номер версии

// Дополнительный номер версии

// Номер сборки

// Редакция

//

// Можно задать все значения или принять номер сборки и номер редакции по умолчанию.

// используя "\*", как показано ниже:

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyFileVersion("1.0.0.0")]

74. Resources.Designer.cs. Класс ресурса со строгой типизацией для поиска локализованных строк и т.д.

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace ScriptGen.Properties {

using System;

/// <summary>

/// Класс ресурса со строгой типизацией для поиска локализованных строк и т.д.

/// </summary>

// Этот класс создан автоматически классом StronglyTypedResourceBuilder

// с помощью такого средства, как ResGen или Visual Studio.

// Чтобы добавить или удалить член, измените файл .ResX и снова запустите ResGen

// с параметром /str или перестройте свой проект VS.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

internal class Resources {

private static global::System.Resources.ResourceManager resourceMan;

private static global::System.Globalization.CultureInfo resourceCulture;

[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]

internal Resources() {

}

/// <summary>

/// Возвращает кэшированный экземпляр ResourceManager, использованный этим классом.

/// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

internal static global::System.Resources.ResourceManager ResourceManager {

get {

if (object.ReferenceEquals(resourceMan, null)) {

global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("ScriptGen.Properties.Resources", typeof(Resources).Assembly);

resourceMan = temp;

}

return resourceMan;

}

}

/// <summary>

/// Перезаписывает свойство CurrentUICulture текущего потока для всех

/// обращений к ресурсу с помощью этого класса ресурса со строгой типизацией.

/// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

internal static global::System.Globalization.CultureInfo Culture {

get {

return resourceCulture;

}

set {

resourceCulture = value;

}

}

/// <summary>

/// Ищет локализованную строку, похожую на Aaron

/// </summary>

internal static string EN\_Name {

get {

return ResourceManager.GetString("EN\_Name", resourceCulture);

}

}

/// <summary>

/// Ищет локализованную строку, похожую на Adams

/// </summary>

internal static string EN\_Surname {

get {

return ResourceManager.GetString("EN\_Surname", resourceCulture);

}

}

/// <summary>

/// Ищет локализованную строку, похожую на a

/// </summary>

internal static string EN\_Word {

get {

return ResourceManager.GetString("EN\_Word", resourceCulture);

}

}

/// <summary>

/// Ищет локализованную строку, похожую на Авдокея

/// </summary>

internal static string RU\_Name {

get {

return ResourceManager.GetString("RU\_Name", resourceCulture);

}

}

/// <summary>

/// Ищет локализованную строку, похожую на Иванов

/// </summary>

internal static string RU\_Surname {

get {

return ResourceManager.GetString("RU\_Surname", resourceCulture);

}

}

/// <summary>

/// Ищет локализованную строку, похожую на абажур

/// </summary>

internal static string RU\_Word {

get {

return ResourceManager.GetString("RU\_Word", resourceCulture);

}

}

}

}

75. Settings.Designer.cs.

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace ScriptGen.Properties {

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "15.8.0.0")]

internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase {

private static Settings defaultInstance = ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));

public static Settings Default {

get {

return defaultInstance;

}

}

}

}

76. SettingsDialog.xaml.

<metro:MetroWindow

x:Class="ScriptGen.Settings.SettingsDialog"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:metro="http://metro.mahapps.com/winfx/xaml/controls"

xmlns:local="clr-namespace:ScriptGen.Settings.View"

mc:Ignorable="d"

ShowMaxRestoreButton="False"

ShowMinButton="False"

ResizeMode="NoResize"

Title="Параметры"

Width="860"

Height="630"

Closing="SettingsClosing">

<local:SettingsV/>

</metro:MetroWindow>

77. SettingsManager.cs. Осуществляет работу с данными настроек.

using ScriptGen.Common;

using ScriptGen.Settings.Model;

using ScriptGen.ViewModel;

namespace ScriptGen.Settings

{

/// <summary>

/// Осуществляет работу с данными настроек

/// </summary>

public static class SettingsManager

{

public static void Save(SettingsSave settings)

{

if (!FileWork.BinarySave(settings, @"Data\settings.bin", out string error))

StatusBarManagerVM.Error("Не удалось сохранить настройки!", error);

}

public static SettingsSave? Load()

{

LoadOptions();

LoadTemplates();

if (FileWork.BinaryLoad(@"Data\settings.bin", out string error, out object settingsObject)

&& settingsObject.GetType() == typeof(SettingsSave))

{

SettingsSave settings = (SettingsSave)settingsObject;

Info.ActiveOptionIndex = settings.Common.ActiveOptionIndex;

Info.ActiveTemplateSQLIndex = settings.Common.ActiveTemplateSQLIndex;

Info.ActiveTemplateProgrammingIndex = settings.Common.ActiveTemplateProgrammingIndex;

return settings;

}

return null;

}

public static void LoadOption(string path)

{

if (SettingsFileWork.LoadOption(path, out OptionData option))

if (SettingsFileWork.SaveLocalOption(path))

Info.AddOption(option);

}

public static void LoadTemplate(string path)

{

if (SettingsFileWork.LoadTemplate(path, out TemplateData template))

if (SettingsFileWork.SaveLocalTemplate(path))

Info.AddTemplate(template);

}

private static void LoadOptions()

{

if (SettingsFileWork.GetAllOptionsPath(out string[] paths))

for (int i = 0; i < paths.Length; i++)

if (SettingsFileWork.LoadOption(paths[i], out OptionData option))

Info.AddOption(option);

}

private static void LoadTemplates()

{

if (SettingsFileWork.GetAllTemplatesPath(out string[] paths))

for (int i = 0; i < paths.Length; i++)

if (SettingsFileWork.LoadTemplate(paths[i], out TemplateData template))

Info.AddTemplate(template);

}

}

}

78. ColorM.cs. Модель цвета.

using System;

using System.Windows.Media;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using MaterialDesignColors;

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель цвета

/// </summary>

public class ColorM : ViewModelBase

{

public Action<ColorM> OnSetColor;

public Swatch Swatch;

private bool isSelect;

public ColorM(Swatch swatch)

{

Swatch = swatch;

Name = $"{char.ToUpper(Swatch.Name[0])}{swatch.Name.Substring(1)}";

Color = new SolidColorBrush(Swatch.ExemplarHue.Color);

SelectColor = new SimpleCommand(() => OnSetColor?.Invoke(this));

}

public SimpleCommand SelectColor { get; }

public string Name { get; }

public SolidColorBrush Color { get; }

public bool IsSelect

{

get => isSelect;

set

{

isSelect = value;

OnPropertyChanged();

}

}

}

}

79. CommonM.cs. Модель настроек приложения.

using System;

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель настроек приложения

/// </summary>

[Serializable]

public struct CommonM

{

public int ActiveOptionIndex;

public int ActiveTemplateSQLIndex;

public int ActiveTemplateProgrammingIndex;

}

}

80. Option.cs. Модель экземпляра настроек.

using System.Linq;

using System.Text.RegularExpressions;

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель экземпляра настроек

/// </summary>

public class Option

{

public string Name;

public string[] FalseWords;

public string[] SQLTypes;

public string[] ProgrammingTypes;

public Option(OptionData optionData)

{

Name = optionData.Name;

FalseWords = GetWords(optionData.FalseWords).Select(s => s.ToLower()).ToArray();

SQLTypes = GetWords(optionData.SQLTypes);

ProgrammingTypes = GetWords(optionData.ProgrammingTypes);

}

private string[] GetWords(string data)

=> Regex.Split(data, "\\s+").Where(s => !string.IsNullOrWhiteSpace(s)).ToArray();

}

}

81. OptionData.cs. Модель настроек для загрузки.

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель настроек для загрузки

/// </summary>

public struct OptionData

{

public string Name;

public string FalseWords;

public string SQLTypes;

public string ProgrammingTypes;

}

}

82. SettingsSave.cs. Модель для сохранения настроек.

using System;

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель для сохранения настроек

/// </summary>

[Serializable]

public struct SettingsSave

{

public CommonM Common;

public bool IsDark;

public string ColorName;

public SettingsSave(CommonM common, ThemeM theme)

{

Common = common;

IsDark = theme.IsDark;

ColorName = theme.SelectColor.Name;

}

}

}

83. TemplateData.cs. Модель для загрузки шаблонов.

using ScriptGen.Common;

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель для загрузки шаблонов

/// </summary>

public struct TemplateData

{

public string Name;

public TemplateType Type;

public string Template;

}

}

84. ThemeM.cs. Модель темы.

namespace ScriptGen.Settings.Model

{

/// <summary>

/// Модель темы

/// </summary>

public struct ThemeM

{

public bool IsDark;

public ColorM SelectColor;

}

}

85. CommonV.xaml. Представление главных настроек.

<UserControl

x:Class="ScriptGen.Settings.View.CommonV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

mc:Ignorable="d" >

<StackPanel

Margin="20">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition/>

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="34"/>

<RowDefinition Height="36"/>

<RowDefinition Height="36"/>

<RowDefinition Height="36"/>

<RowDefinition Height="10"/>

</Grid.RowDefinitions>

<TextBlock

Style="{StaticResource MaterialDesignTitleTextBlock}"

VerticalAlignment="Top"

Text="Основное"/>

<TextBlock

Grid.Row="1"

VerticalAlignment="Center"

Text="Настройки: "

FontSize="14"/>

<TextBlock

Grid.Row="2"

VerticalAlignment="Center"

Text="Шаблон SQL: "

FontSize="14"/>

<TextBlock

Grid.Row="3"

VerticalAlignment="Center"

TextWrapping="Wrap"

Text="Шаблон для языков программирования: "

FontSize="14"/>

<ComboBox

ItemsSource="{Binding OptionNames, Mode=OneWay}"

SelectedIndex="{Binding ActiveOptionIndex, Mode=TwoWay}"

Grid.Row="1"

Grid.Column="1"

HorizontalAlignment="Left"

Width="150"

FontSize="14"/>

<ComboBox

ItemsSource="{Binding SqlNames, Mode=OneWay}"

SelectedIndex="{Binding ActiveTemplateSQLIndex, Mode=TwoWay}"

Grid.Row="2"

Grid.Column="1"

HorizontalAlignment="Left"

Width="150"

FontSize="14"/>

<ComboBox

ItemsSource="{Binding ProgrammingNames, Mode=OneWay}"

SelectedIndex="{Binding ActiveTemplateProgrammingIndex, Mode=TwoWay}"

Grid.Row="3"

Grid.Column="1"

HorizontalAlignment="Left"

Width="150"

FontSize="14"/>

<StackPanel

HorizontalAlignment="Right"

Orientation="Horizontal"

Grid.Column="3">

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding AddOption}"

Grid.Column="3"

HorizontalAlignment="Right"

ToolTip="Добавить настройки"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="PlaylistAdd"/>

</Button>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding AddTemplate}"

Grid.Column="3"

HorizontalAlignment="Right"

ToolTip="Добавить шаблон"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="NotePlus"/>

</Button>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding OpenInExplorer}"

Grid.Column="3"

HorizontalAlignment="Right"

ToolTip="Открыть в проводнике"

Width="32"

Padding="0">

<md:PackIcon

Width="24"

Height="24"

Kind="OpenInNew"/>

</Button>

</StackPanel>

<Border

Grid.Row="4"

Grid.ColumnSpan="3"

VerticalAlignment="Bottom"

BorderBrush="Gray"

BorderThickness="0 0 0 1"/>

</Grid>

<Grid

Margin="0 10 0 0">

<Grid.RowDefinitions>

<RowDefinition Height="34"/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBlock

Style="{StaticResource MaterialDesignTitleTextBlock}"

VerticalAlignment="Center"

Text="Плагины"/>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding AddPlugin}"

HorizontalAlignment="Right"

ToolTip="Загрузить плагин"

Width="32"

Padding="0">

<md:PackIcon

Width="26"

Height="26"

Kind="Plus"/>

</Button>

<ScrollViewer

VerticalScrollBarVisibility="Auto"

HorizontalScrollBarVisibility="Disabled"

Grid.Row="2">

<ItemsControl

ItemsSource="{Binding Plugins, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid

Grid.Row="1"

Height="50">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition/>

<ColumnDefinition Width="32"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="MouseDown">

<i:InvokeCommandAction

Command="{Binding OpenInExplorer}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

<Image

Source="{Binding ImagePath, Mode=OneTime}"

Height="40"

Grid.RowSpan="2"/>

<TextBlock

Text="{Binding Name, Mode=OneTime}"

Style="{StaticResource MaterialDesignTitleTextBlock}"

Grid.Column="1"/>

<TextBlock

Text="{Binding Description, Mode=OneTime}"

Margin="2 0 0 0"

Grid.Column="1"

Grid.Row="1"

FontSize="14"/>

<Button

Foreground="{DynamicResource MaterialDesignBodyLight}"

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemovePlaugin}"

ToolTip="Удалить"

Grid.Column="2"

Grid.RowSpan="2"

Padding="0"

Height="26"

Width="26">

<md:PackIcon

Width="18"

Height="18"

Kind="RemoveCircle"/>

</Button>

<Border

Grid.Row="1"

Grid.ColumnSpan="3"

VerticalAlignment="Bottom"

BorderBrush="Gray"

BorderThickness="0 0 0 1"/>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</ScrollViewer>

</Grid>

</StackPanel>

</UserControl>

86. SettingsV.xaml. Представление настроек.

<UserControl

x:Class="ScriptGen.Settings.View.SettingsV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:ScriptGen.Settings.View"

mc:Ignorable="d" >

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition Width="700"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="550"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<ListBox

Grid.RowSpan="2"

BorderThickness="0 0 1 0"

BorderBrush="Gray">

<ListBoxItem

IsSelected="True"

x:Name="CommonPage"

FontSize="14">

Общее

</ListBoxItem>

<ListBoxItem

x:Name="ThemePage"

FontSize="14">

Темы

</ListBoxItem>

</ListBox>

<local:CommonV

DataContext="{Binding Common, Mode=OneTime}"

Visibility="{Binding ElementName=CommonPage, Path=IsSelected, Converter={StaticResource BoolVisible}}"

Grid.Column="1"/>

<local:ThemeV

DataContext="{Binding Theme, Mode=OneTime}"

Visibility="{Binding ElementName=ThemePage, Path=IsSelected, Converter={StaticResource BoolVisible}}"

Grid.Column="1"/>

<Button

Command="{Binding Cancel}"

HorizontalAlignment="Right"

Grid.Column="1"

Grid.Row="1"

Margin="5">

Отмена

</Button>

<Button

Command="{Binding Accept}"

HorizontalAlignment="Right"

Grid.Column="1"

Grid.Row="1"

Margin="5 5 95 5">

Сохранить

</Button>

</Grid>

</UserControl>

87. ThemeV.xaml. Представление настроек тем.

<UserControl

x:Class="ScriptGen.Settings.View.ThemeV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d" >

<StackPanel

Margin="20">

<TextBlock

Style="{StaticResource MaterialDesignTitleTextBlock}"

Height="34"

Text="Персонализация"/>

<StackPanel

Orientation="Horizontal">

<TextBlock

VerticalAlignment="Center"

Text="Тема: "

FontSize="14"/>

<ComboBox

Margin="10 0 0 0"

FontSize="14">

<ComboBoxItem

x:Name="DarkMode"

IsSelected="{Binding IsDark, Mode=TwoWay}"

Content="Dark"/>

<ComboBoxItem

IsSelected="{Binding ElementName=DarkMode, Path=IsSelected, Converter={StaticResource InvBool}}"

Content="Light"/>

</ComboBox>

</StackPanel>

<TextBlock

Margin="0 4 0 0"

Text="Цвет: "

FontSize="14"/>

<ItemsControl

ItemsSource="{Binding Colors, Mode=OneTime}"

Margin="0 10 0 0">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid

Width="48"

Height="48">

<Border

Visibility="{Binding IsSelect, Mode=OneWay, Converter={StaticResource BoolVisible}}"

BorderBrush="{DynamicResource PrimaryHueMidBrush}"

BorderThickness="2"

Width="48"

Height="48"/>

<Button

Command="{Binding SelectColor}"

ToolTip="{Binding Name, Mode=OneTime}"

Background="{Binding Color, Mode=OneTime}"

BorderBrush="{x:Null}"

Margin="4 5 4 4"

Width="40"

Height="40"/>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</StackPanel>

</UserControl>

88. CommonVM.cs. Управляет основными настройками приложения.

using System.Collections.ObjectModel;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using Explorer;

using ScriptGen.API;

using ScriptGen.Common;

using ScriptGen.Settings.Model;

namespace ScriptGen.Settings.ViewModel

{

/// <summary>

/// Управляет основными настройками приложения

/// </summary>

public class CommonVM : ViewModelBase

{

private CommonM common;

private CommonM oldCommon;

private FileDialog fileDialog

= new FileDialog();

public CommonVM()

{

common.ActiveOptionIndex = Info.ActiveOptionIndex;

common.ActiveTemplateSQLIndex = Info.ActiveTemplateSQLIndex;

common.ActiveTemplateProgrammingIndex = Info.ActiveTemplateProgrammingIndex;

AddOption = new SimpleCommand(LoadOption);

AddTemplate = new SimpleCommand(LoadTemplate);

OpenInExplorer = new SimpleCommand(() => FileWork.OpenInExplorer("Data"));

AddPlugin = new SimpleCommand(LoadPlugin);

Info.OnSettingsAdd += () =>

{

OnPropertyChanged("OptionNames");

OnPropertyChanged("SqlNames");

OnPropertyChanged("ProgrammingNames");

};

}

public SimpleCommand AddOption { get; }

public SimpleCommand AddTemplate { get; }

public SimpleCommand OpenInExplorer { get; }

public SimpleCommand AddPlugin { get; }

public string[] OptionNames

=> Info.OptionNames;

public string[] SqlNames

=> Info.SqlNames;

public string[] ProgrammingNames

=> Info.ProgrammingNames;

public ObservableCollection<PluginData> Plugins

=> PluginManager.Plugins;

public int ActiveOptionIndex

{

get => common.ActiveOptionIndex;

set

{

common.ActiveOptionIndex = value;

OnPropertyChanged();

}

}

public int ActiveTemplateSQLIndex

{

get => common.ActiveTemplateSQLIndex;

set

{

common.ActiveTemplateSQLIndex = value;

OnPropertyChanged();

}

}

public int ActiveTemplateProgrammingIndex

{

get => common.ActiveTemplateProgrammingIndex;

set

{

common.ActiveTemplateProgrammingIndex = value;

OnPropertyChanged();

}

}

private void LoadOption()

{

if (fileDialog.Open(ExplorerSelectType.File, ExplorerType.Open, ".xml"))

SettingsManager.LoadOption(fileDialog.SelectPath);

}

private void LoadTemplate()

{

if (fileDialog.Open(ExplorerSelectType.File, ExplorerType.Open, ".xml"))

SettingsManager.LoadTemplate(fileDialog.SelectPath);

}

private void LoadPlugin()

{

if (fileDialog.Open(ExplorerSelectType.File, ExplorerType.Open, ".txt", ".xml"))

PluginManager.LoadPlugin(fileDialog.SelectPath);

}

public void Save()

=> oldCommon = common;

public void Accept()

{

Info.ActiveOptionIndex = ActiveOptionIndex;

Info.ActiveTemplateSQLIndex = ActiveTemplateSQLIndex;

Info.ActiveTemplateProgrammingIndex = ActiveTemplateProgrammingIndex;

}

public void Cancel()

{

ActiveOptionIndex = oldCommon.ActiveOptionIndex;

ActiveTemplateSQLIndex = oldCommon.ActiveTemplateSQLIndex;

ActiveTemplateSQLIndex = oldCommon.ActiveTemplateSQLIndex;

}

public CommonM GetCommon()

=> common;

}

}

89. SettingsVM.cs. Управляет настройками.

using System;

using BaseMVVM.Command;

using ScriptGen.Settings.Model;

namespace ScriptGen.Settings.ViewModel

{

/// <summary>

/// Управляет настройками

/// </summary>

public class SettingsVM

{

public Action Close;

public SettingsVM(bool isDark, string colorName)

{

Common = new CommonVM();

Theme = new ThemeVM(isDark, colorName);

Accept = new SimpleCommand(AcceptSettings);

Cancel = new SimpleCommand(CancelSettings);

SaveSettings();

}

public SimpleCommand Accept { get; }

public SimpleCommand Cancel { get; }

public CommonVM Common { get; }

public ThemeVM Theme { get; }

public void SaveSettings()

{

Common.Save();

Theme.Save();

}

public void AcceptSettings()

{

Common.Accept();

Theme.Accept();

SettingsManager.Save(new SettingsSave(Common.GetCommon(), Theme.GetTheme()));

Close?.Invoke();

}

public void CancelSettings()

{

Common.Cancel();

Theme.Cancel();

Close?.Invoke();

}

}

}

90. ThemeVM.cs. Управляет темами приложения.

using System.Collections.ObjectModel;

using System.Linq;

using BaseMVVM.Abstraction;

using MaterialDesignColors;

using MaterialDesignThemes.Wpf;

using ScriptGen.Settings.Model;

namespace ScriptGen.Settings.ViewModel

{

/// <summary>

/// Управляет темами приложения

/// </summary>

public class ThemeVM : ViewModelBase

{

private ThemeM theme;

private ThemeM oldTheme;

public ThemeVM(bool isDark, string colorName)

{

Swatch[] swatches = new SwatchesProvider().Swatches.ToArray();

foreach (Swatch swatch in new SwatchesProvider().Swatches)

{

ColorM color = new ColorM(swatch)

{

OnSetColor = SetColor

};

Colors.Add(color);

}

IsDark = isDark;

theme.SelectColor = Colors.FirstOrDefault(c => c.Name == colorName);

SelectColor.IsSelect = true;

Accept();

}

public ObservableCollection<ColorM> Colors { get; }

= new ObservableCollection<ColorM>();

public bool IsDark

{

get => theme.IsDark;

set

{

theme.IsDark = value;

OnPropertyChanged();

}

}

public ColorM SelectColor

{

get => theme.SelectColor;

set

{

SelectColor.IsSelect = false;

value.IsSelect = true;

theme.SelectColor = value;

OnPropertyChanged();

}

}

private void SetColor(ColorM newColor)

{

if (newColor != SelectColor)

SelectColor = newColor;

}

public void Save()

=> oldTheme = theme;

public void Accept()

{

new PaletteHelper().SetLightDark(IsDark);

new PaletteHelper().ReplacePrimaryColor(SelectColor.Swatch);

if (SelectColor.Swatch.IsAccented)

new PaletteHelper().ReplaceAccentColor(SelectColor.Swatch);

}

public void Cancel()

{

IsDark = oldTheme.IsDark;

SelectColor = oldTheme.SelectColor;

}

public ThemeM GetTheme()

=> theme;

}

}

91. FieldV.xaml. x:Name="MainControl".

<UserControl

x:Class="ScriptGen.View.FieldV"

x:Name="MainControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

mc:Ignorable="d"

ToolTip="{Binding Message, Mode=OneWay}">

<i:Interaction.Triggers>

<i:EventTrigger

EventName="Loaded">

<i:InvokeCommandAction

Command="{Binding Loaded}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

<Grid

Margin="2 -2 3 2">

<Grid.Style>

<Style TargetType="{x:Type Grid}">

<Style.Triggers>

<DataTrigger Binding="{Binding FocusedElement}" Value="Name">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=FieldName}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="LogicalName">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=FieldLogicalName}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="ProgrammingName">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=FieldProgrammingName}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="DataType">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=FieldDataType}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="ProgrammingType">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=FieldProgrammingType}"/>

</DataTrigger>

</Style.Triggers>

</Style>

</Grid.Style>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="140"/>

<ColumnDefinition Width="20"/>

<ColumnDefinition Width="20"/>

<ColumnDefinition Width="28"/>

</Grid.ColumnDefinitions>

<md:PackIcon

Visibility="{Binding IsError, Mode=OneWay, Converter={StaticResource BoolVisible}}"

Foreground="#C51162"

Kind="WarningBox"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="-34 0 0 0"

Width="24"

Height="24"/>

<TextBox

x:Name="FieldName"

Text="{Binding Name, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource PhysicalVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Название"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Up"

Command="{Binding NameUp}"/>

<KeyBinding Key="Down"

Command="{Binding NameDown}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<TextBox

x:Name="FieldLogicalName"

Text="{Binding LogicalName, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource LogicalVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Логическое название"

Grid.ColumnSpan="2"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Up"

Command="{Binding NameUp}"/>

<KeyBinding Key="Down"

Command="{Binding NameDown}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<TextBox

x:Name="FieldProgrammingName"

Text="{Binding ProgrammingName, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource ProgrammingVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Название"

AcceptsReturn="False"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Up"

Command="{Binding NameUp}"/>

<KeyBinding Key="Down"

Command="{Binding NameDown}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<ComboBox

x:Name="FieldDataType"

Text="{Binding DataType, Mode=TwoWay}"

ItemsSource="{Binding DataTypeSource, Mode=OneWay}"

Visibility="{Binding TypeModel, Converter={StaticResource PhysicalVisible}, Mode=OneWay}"

IsEnabled="{Binding Type, Converter={StaticResource DisabledFK}, Mode=OneTime}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

Grid.Column="1"

Margin="0 0 4 0"

md:HintAssist.Hint="Тип данных"

IsEditable="True"

FontSize="14">

<ComboBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding DataTypeEnter}"/>

</ComboBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</ComboBox>

<ComboBox

x:Name="FieldProgrammingType"

Text="{Binding ProgrammingType, Mode=TwoWay}"

ItemsSource="{Binding ProgrammingTypeSource, Mode=OneWay}"

Visibility="{Binding TypeModel, Converter={StaticResource ProgrammingVisible}, Mode=OneWay}"

IsEnabled="{Binding Type, Converter={StaticResource DisabledFK}, Mode=OneTime}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

Grid.Column="1"

Margin="0 0 4 0"

md:HintAssist.Hint="Тип данных"

IsEditable="True"

FontSize="14">

<ComboBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding ProgrammingTypeEnter}"/>

</ComboBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</ComboBox>

<CheckBox

IsChecked="{Binding IsNull, Mode=OneWayToSource}"

IsEnabled="{Binding Type, Converter={StaticResource DisabledPK}, Mode=OneTime}"

Style="{DynamicResource MaterialDesignCheckBox}"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Grid.Column="2"

ToolTip="IsNull"/>

<CheckBox

IsChecked="{Binding IsUnique, Mode=TwoWay}"

IsEnabled="{Binding Type, Converter={StaticResource DisabledPK}, Mode=OneTime}"

Style="{DynamicResource MaterialDesignCheckBox}"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Grid.Column="3"

ToolTip="IsUnique"/>

<Button

Command="{Binding RemoveElement}"

IsEnabled="{Binding Type, Converter={StaticResource DisabledPK}, Mode=OneTime}"

Style="{DynamicResource MaterialDesignFlatButton}"

Grid.Column="4"

Padding="0"

Width="24"

Height="24"

ToolTip="Удалить поле">

<md:PackIcon

Kind="TableRowRemove"/>

</Button>

</Grid>

</UserControl>

92. LineV.xaml. Представление соединительной линии.

<UserControl

x:Class="ScriptGen.View.LineV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

Width="200"

Height="200">

<Canvas>

<Thumb

x:Name="MainThumb"

Panel.ZIndex="20"

Width="20"

Height="20"

DragDelta="ThumbDragDelta"

DragCompleted="ThumbDragCompleted"/>

<Line

x:Name="SourceLine"

Stroke="{StaticResource PrimaryHueDarkBrush}"

X1="{Binding SourceX, Mode=TwoWay}"

Y1="{Binding SourceY, Mode=TwoWay}"

X2="{Binding ElementName=ConnectionLine, Path=X1, Mode=OneWay}"

Y2="{Binding ElementName=SourceLine, Path=Y1, Mode=OneWay}"

StrokeThickness="2"

Cursor="SizeNS"

MouseDown="SourceLineMouseDown"/>

<Line

x:Name="ConnectionLine"

Stroke="{StaticResource PrimaryHueDarkBrush}"

X1="{Binding ConnectionX, Mode=TwoWay}"

X2="{Binding ElementName=ConnectionLine, Path=X1, Mode=OneWay}"

Y1="{Binding ElementName=SourceLine, Path=Y1, Mode=OneWay}"

Y2="{Binding ElementName=TargetLine, Path=Y1, Mode=OneWay}"

StrokeThickness="2"

Cursor="SizeWE"

MouseDown="ConnectionLineMouseDown"/>

<Line

x:Name="TargetLine"

Stroke="{StaticResource PrimaryHueDarkBrush}"

X1="{Binding TargetX, Mode=TwoWay}"

Y1="{Binding TargetY, Mode=TwoWay}"

X2="{Binding ElementName=ConnectionLine, Path=X1, Mode=OneWay}"

Y2="{Binding ElementName=TargetLine, Path=Y1, Mode=OneWay}"

StrokeThickness="2"

Cursor="SizeNS"

MouseDown="TargetLineMouseDown"/>

</Canvas>

</UserControl>

93. PageV.xaml. Представление страниц.

<UserControl

x:Class="ScriptGen.View.PageV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:ScriptGen.View"

xmlns:bp="clr-namespace:ScriptGen.BindingProperty"

mc:Ignorable="d"

PreviewMouseWheel="UserControlPreviewMouseWheel"

d:DesignHeight="450" d:DesignWidth="800">

<bp:DataBinding.DataBind>

<bp:DataBindCollection>

<bp:DataBind

Source="{Binding RelativeSource={RelativeSource AncestorType={x:Type UserControl}}, Path=ActualWidth}"

Target="{Binding ActualWidth, Mode=OneWayToSource}"/>

<bp:DataBind

Source="{Binding RelativeSource={RelativeSource AncestorType={x:Type UserControl}}, Path=ActualHeight}"

Target="{Binding ActualHeight, Mode=OneWayToSource}"/>

</bp:DataBindCollection>

</bp:DataBinding.DataBind>

<md:DialogHost

CloseOnClickAway="True">

<ScrollViewer

x:Name="Scroll"

bp:ScrollViewerBinding.VerticalOffset="{Binding ScrollVertical, Mode=TwoWay}"

bp:ScrollViewerBinding.HorizontalOffset="{Binding ScrollHorizontal, Mode=TwoWay}"

HorizontalScrollBarVisibility="Auto"

VerticalScrollBarVisibility="Auto"

Grid.Row="1">

<Grid>

<ItemsControl

ItemsSource="{Binding Lines, Mode=OneWay}"

Width="2800"

Height="1575"

Panel.ZIndex="1">

<ItemsControl.ItemTemplate>

<DataTemplate>

<local:LineV/>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<Canvas/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

<ItemsControl

ItemsSource="{Binding Tables, Mode=OneWay}"

Width="2800"

Height="1575"

Panel.ZIndex="2">

<ItemsControl.ItemTemplate>

<DataTemplate>

<local:TableV/>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<Canvas/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</Grid>

</ScrollViewer>

</md:DialogHost>

</UserControl>

94. ScriptGenV.xaml. Представление главного окна.

<metro:MetroWindow

x:Class="ScriptGen.View.ScriptGenV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:metro="http://metro.mahapps.com/winfx/xaml/controls"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:locale="clr-namespace:ScriptGen.View"

xmlns:faker="clr-namespace:ScriptGen.Faker.View"

xmlns:bp="clr-namespace:ScriptGen.BindingProperty"

xmlns:comp="clr-namespace:Explorer;assembly=Explorer"

mc:Ignorable="d"

WindowState="Maximized"

Title="Script Gen"

MinHeight="500"

MinWidth="700"

Height="800"

Width="1200">

<DockPanel

LastChildFill="True">

<Border

BorderBrush="{DynamicResource PrimaryHueDarkBrush}"

BorderThickness="0 0 0 2"

DockPanel.Dock="Top">

<ToolBarTray

Background="{DynamicResource MaterialDesignPaper}"

IsLocked="True">

<ToolBar

Background="{DynamicResource MaterialDesignPaper}"

Style="{DynamicResource MaterialDesignToolBar}"

ClipToBounds="False">

<Button

Command="{Binding OpenSettings}"

ToolTip="Параметры"

Padding="6">

<md:PackIcon

Kind="Settings"/>

</Button>

<md:PopupBox

Visibility="{Binding HavePlugins, Mode=OneWay, Converter={StaticResource BoolVisible}}"

PlacementMode="RightAndAlignTopEdges"

StaysOpen="True">

<md:PopupBox.ToggleContent>

<md:PackIcon

ToolTip="Плагины"

Kind="Ballot"/>

</md:PopupBox.ToggleContent>

<ScrollViewer

HorizontalScrollBarVisibility="Disabled"

VerticalScrollBarVisibility="Auto"

Width="250"

MaxHeight="300">

<ItemsControl

ItemsSource="{Binding Plugins, Mode=OneWay}"

Margin="8 0 8 0">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Button

Content="{Binding ActionContent, Mode=OneTime}"

ToolTip="{Binding ActionToolTip, Mode=OneTime}"

Command="{Binding PluginCommand}"

Style="{StaticResource MaterialDesignFlatButton}"

Foreground="{DynamicResource MaterialDesignBodyLight}"

MaxWidth="200"

FontSize="14"/>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</ScrollViewer>

</md:PopupBox>

<Button

Command="{Binding OpenPage}"

ToolTip="Открыть"

Padding="6"

Click="ClearFocus">

<md:PackIcon

Kind="Upload"/>

</Button>

<Button

Command="{Binding SavePage}"

ToolTip="Сохранить"

Padding="6"

Click="ClearFocus">

<md:PackIcon

Kind="Download"/>

</Button>

</ToolBar>

<ToolBar

Background="{DynamicResource MaterialDesignPaper}"

Style="{DynamicResource MaterialDesignToolBar}"

ClipToBounds="False">

<Button

Command="{Binding AddPage}"

ToolTip="Добавить модель"

Padding="6">

<md:PackIcon

Kind="DatabasePlus"/>

</Button>

<Button

Command="{Binding AddTable}"

ToolTip="Создать таблицу"

Padding="6">

<md:PackIcon

Kind="TablePlus"/>

</Button>

<Button

Command="{Binding CheckError}"

ToolTip="Проверить"

Padding="6"

Click="ClearFocus">

<md:PackIcon

Kind="BugCheck"/>

</Button>

<Button

Command="{Binding GenerateScript}"

ToolTip="Генерировать скрипт"

Padding="6"

Click="ClearFocus">

<md:PackIcon

Kind="ProgressDownload"/>

</Button>

<ComboBox

SelectedIndex="{Binding TypeModel, Mode=TwoWay, Converter={StaticResource ModelTypeToInt}}"

FontSize="14">

<ComboBoxItem

IsSelected="True"

Content="Physical"/>

<ComboBoxItem

Content="Logical"/>

<ComboBoxItem

Content="Programming"/>

</ComboBox>

</ToolBar>

</ToolBarTray>

</Border>

<StatusBar

DataContext="{Binding StatusBarManager}"

DockPanel.Dock="Bottom"

Height="30">

<StackPanel

Orientation="Horizontal"

Margin="10 0">

<md:PackIcon

Kind="{Binding Kind, Mode=OneWay}"/>

<TextBlock

Text="{Binding Data, Mode=OneWay}"

Margin="5 0 0 0"/>

</StackPanel>

</StatusBar>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="4\*" MinHeight="1"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*" MinHeight="1"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" MinWidth="1"/>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="5\*" MinWidth="1"/>

</Grid.ColumnDefinitions>

<Grid

Background="{DynamicResource MaterialDesignPaper}">

<TreeView

ItemsSource="{Binding Pages}">

<TreeView.ItemContainerStyle>

<Style

BasedOn="{StaticResource NoSelectTreeViewItem}"

TargetType="{x:Type TreeViewItem}">

<Setter Property="Padding" Value="0"/>

</Style>

</TreeView.ItemContainerStyle>

<TreeView.ItemTemplate>

<HierarchicalDataTemplate

ItemsSource="{Binding Tables}">

<StackPanel

Orientation="Horizontal">

<md:PackIcon

Kind="Database"

VerticalAlignment="Center"

Margin="0 0 4 0"/>

<TextBox

Text="{Binding Name, Mode=TwoWay}"

md:HintAssist.Hint="Модель"

MinWidth="100"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

</StackPanel>

<HierarchicalDataTemplate.ItemTemplate>

<HierarchicalDataTemplate

ItemsSource="{Binding Fields}">

<StackPanel

Orientation="Horizontal">

<md:PackIcon

Kind="Table"

VerticalAlignment="Center"

Margin="0 0 4 0"/>

<TextBox

Text="{Binding Name, Mode=TwoWay}"

md:HintAssist.Hint="Таблица"

MinWidth="100"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

</StackPanel>

<HierarchicalDataTemplate.ItemTemplate>

<DataTemplate>

<StackPanel

Orientation="Horizontal">

<md:PackIcon

Kind="{Binding Type, Converter={StaticResource Icon}, Mode=OneWay}"

VerticalAlignment="Center"

Margin="0 0 4 0"/>

<TextBox

Text="{Binding Name, Mode=TwoWay}"

md:HintAssist.Hint="Поле"

MinWidth="100"

FontSize="14"

KeyDown="LostFocusOnEnter"/>

</StackPanel>

</DataTemplate>

</HierarchicalDataTemplate.ItemTemplate>

</HierarchicalDataTemplate>

</HierarchicalDataTemplate.ItemTemplate>

</HierarchicalDataTemplate>

</TreeView.ItemTemplate>

</TreeView>

</Grid>

<GridSplitter

HorizontalAlignment="Center"

VerticalAlignment="Stretch"

ShowsPreview="False"

Grid.Column="1"

Width="3"/>

<Grid

Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition Height="32"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<StackPanel

Orientation="Horizontal">

<ItemsControl

ItemsSource="{Binding Pages, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<RadioButton

Style="{StaticResource MaterialDesignTabRadioButton}"

BorderBrush="{DynamicResource PrimaryHueMidBrush}"

IsChecked="{Binding IsSelect, Mode=TwoWay}"

Command="{Binding SelectPage}"

GroupName="Pages"

Padding="8 0 0 0"

MaxWidth="200">

<StackPanel

Orientation="Horizontal">

<comp:EditableTextBlock

Text="{Binding Name, Mode=TwoWay}"

MaxWidth="158"/>

<Button

Foreground="{DynamicResource MaterialDesignBodyLight}"

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemovePage}"

ToolTip="Закрыть модель"

VerticalAlignment="Bottom"

Padding="0"

Height="22"

Margin="4 0 0 0"

Width="22">

<md:PackIcon

Kind="DatabaseMinus"/>

</Button>

</StackPanel>

</RadioButton>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel

Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</StackPanel>

<md:PopupBox

Visibility="{Binding HavePages, Mode=OneWay, Converter={StaticResource BoolVisible}}"

Grid.Column="1"

PlacementMode="LeftAndAlignTopEdges"

StaysOpen="True">

<md:PopupBox.ToggleContent>

<md:PackIcon

Margin="8 8 0 0"

Width="22"

Height="22"

Kind="MenuDown"/>

</md:PopupBox.ToggleContent>

<ScrollViewer

HorizontalScrollBarVisibility="Disabled"

VerticalScrollBarVisibility="Auto"

Width="250"

MaxHeight="300">

<ItemsControl

ItemsSource="{Binding Pages, Mode=OneWay}"

Margin="16 8 16 8">

<ItemsControl.ItemTemplate>

<DataTemplate>

<RadioButton

Style="{StaticResource MaterialDesignTabRadioButton}"

BorderBrush="{DynamicResource PrimaryHueMidBrush}"

IsChecked="{Binding IsSelect, Mode=TwoWay}"

Command="{Binding SelectPage}"

HorizontalContentAlignment="Left"

GroupName="Pages"

Padding="0"

Margin="0"

Width="200">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="178"/>

<ColumnDefinition Width="22"/>

</Grid.ColumnDefinitions>

<TextBlock

Text="{Binding Name, Mode=OneWay}"/>

<Button

Grid.Column="1"

Foreground="{DynamicResource MaterialDesignBodyLight}"

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemovePage}"

ToolTip="Закрыть модель"

VerticalAlignment="Bottom"

Padding="0"

Height="22"

Width="22">

<md:PackIcon

Kind="DatabaseMinus"/>

</Button>

</Grid>

</RadioButton>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</ScrollViewer>

</md:PopupBox>

<Grid

Grid.Row="1"

Grid.ColumnSpan="2">

<ItemsControl

ItemsSource="{Binding Pages, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<locale:PageV

Visibility="{Binding IsSelect, Mode=OneWay, Converter={StaticResource BoolVisible}}"/>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<Grid/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

<bp:DataBinding.DataBind>

<bp:DataBindCollection>

<bp:DataBind

Source="{Binding RelativeSource={RelativeSource AncestorType={x:Type Grid}}, Path=ActualWidth}"

Target="{Binding ActualWidth, Mode=OneWayToSource}"/>

<bp:DataBind

Source="{Binding RelativeSource={RelativeSource AncestorType={x:Type Grid}}, Path=ActualHeight}"

Target="{Binding ActualHeight, Mode=OneWayToSource}"/>

</bp:DataBindCollection>

</bp:DataBinding.DataBind>

</Grid>

</Grid>

</Grid>

<GridSplitter

HorizontalAlignment="Stretch"

VerticalAlignment="Center"

ShowsPreview="False"

Grid.Row="1"

Height="3"/>

<Grid

Background="{DynamicResource MaterialDesignPaper}"

Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" MinWidth="1"/>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="0.5\*" MinWidth="1"/>

</Grid.ColumnDefinitions>

<faker:FakerV

DataContext="{Binding FakerContext}"/>

<GridSplitter

HorizontalAlignment="Center"

VerticalAlignment="Stretch"

ShowsPreview="False"

Grid.Column="1"

Width="3"/>

<Grid

DataContext="{Binding StatusBarManager}"

Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Border

BorderBrush="{DynamicResource PrimaryHueDarkBrush}"

BorderThickness="0 0 0 2"

VerticalAlignment="Bottom"/>

<StackPanel

Orientation="Horizontal"

Margin="10 0 0 0">

<TextBlock

Style="{StaticResource MaterialDesignSubheadingTextBlock}"

VerticalAlignment="Center"

Text="История"/>

</StackPanel>

<StackPanel

Orientation="Horizontal"

HorizontalAlignment="Right"

Margin="0 0 10 0">

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding ClearHistory}"

ToolTip="Очистить историю"

Padding="0"

Height="30"

Width="30">

<md:PackIcon

Kind="Backspace"/>

</Button>

</StackPanel>

<ScrollViewer

Grid.Row="1"

HorizontalScrollBarVisibility="Auto"

VerticalScrollBarVisibility="Auto"

Padding="5">

<ItemsControl

ItemsSource="{Binding History, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid

Margin="0 0 0 4"

MinHeight="26">

<md:PackIcon

Foreground="{Binding Kind, Mode=OneTime, Converter={StaticResource SwitchMessageColor}}"

Kind="{Binding Kind, Mode=OneTime}"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Width="20"

Height="20"/>

<TextBlock

Text="{Binding Data, Mode=OneTime}"

VerticalAlignment="Center"

HorizontalAlignment="Left"

TextWrapping="Wrap"

Margin="25 0 0 0"

FontSize="14"/>

<Button

Visibility="{Binding IsVisible, Mode=OneTime, Converter={StaticResource BoolVisible}}"

Content="{Binding ActionTitle, Mode=OneTime}"

Command="{Binding ActionCommand, Mode=OneTime}"

Style="{StaticResource MaterialDesignRaisedButton}"

VerticalContentAlignment="Center"

VerticalAlignment="Center"

HorizontalAlignment="Right"

FontSize="12"

Padding="6 0"

Margin="10 0"

Height="26"/>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</ScrollViewer>

</Grid>

</Grid>

</Grid>

</DockPanel>

</metro:MetroWindow>

95. TableV.xaml. Представление таблицы.

<UserControl

x:Class="ScriptGen.View.TableV"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

xmlns:local="clr-namespace:ScriptGen.View"

mc:Ignorable="d"

Background="{DynamicResource MaterialDesignPaper}"

BorderBrush="{DynamicResource PrimaryHueDarkBrush}"

Width="{Binding Width, Mode=TwoWay}"

Margin="{Binding Margin, Mode=TwoWay}"

ToolTip="{Binding Message, Mode=OneWay}"

md:ShadowAssist.ShadowDepth="Depth2"

BorderThickness="2">

<i:Interaction.Triggers>

<i:EventTrigger

EventName="Loaded">

<i:InvokeCommandAction

Command="{Binding Loaded}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="3"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="5"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid

Visibility="{Binding IsSelect, Mode=OneWay, Converter={StaticResource BoolVisible}}"

Panel.ZIndex="10"

Background="Black"

Opacity="0.5"

Grid.ColumnSpan="2"

Grid.RowSpan="2">

<i:Interaction.Triggers>

<i:EventTrigger

EventName="MouseDown">

<i:InvokeCommandAction

Command="{Binding SelectElement}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</Grid>

<Thumb

Grid.ColumnSpan="2"

Cursor="SizeAll"

DragDelta="Move\_DragDelta">

</Thumb>

<Thumb

Grid.Column="1"

Grid.Row="1"

Cursor="SizeWE"

DragDelta="Resize\_DragDelta"/>

<StackPanel

Grid.Row="1"

Margin="2 2 0 2">

<Grid>

<Grid.Style>

<Style TargetType="{x:Type Grid}">

<Style.Triggers>

<DataTrigger Binding="{Binding FocusedElement}" Value="Name">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=TableName}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="LogicalName">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=TableLogicalName}"/>

</DataTrigger>

<DataTrigger Binding="{Binding FocusedElement}" Value="ProgrammingName">

<Setter Property="FocusManager.FocusedElement" Value="{Binding ElementName=TableProgrammingName}"/>

</DataTrigger>

</Style.Triggers>

</Style>

</Grid.Style>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="24"/>

<ColumnDefinition/>

<ColumnDefinition Width="24"/>

</Grid.ColumnDefinitions>

<md:PackIcon

Visibility="{Binding IsError, Mode=OneWay, Converter={StaticResource BoolVisible}}"

Foreground="#C51162"

Kind="WarningBox"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="-32 0 0 0"

Width="24"

Height="24"/>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding SelectElement}"

ToolTip="Связать таблицу"

Margin="-2 -4 0 0"

VerticalAlignment="Top"

HorizontalAlignment="Left"

Padding="0"

Width="21"

Height="21">

<md:PackIcon

Width="18"

Height="18"

Kind="ArrangeBringForward"/>

</Button>

<TextBox

x:Name="TableName"

Text="{Binding Name, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource PhysicalVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Название таблицы"

Grid.Column="1"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Down"

Command="{Binding NameEnter}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<TextBox

x:Name="TableLogicalName"

Text="{Binding LogicalName, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource LogicalVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Логическое название таблицы"

Grid.Column="1"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Down"

Command="{Binding NameEnter}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<TextBox

x:Name="TableProgrammingName"

Text="{Binding ProgrammingName, Mode=TwoWay}"

Visibility="{Binding TypeModel, Converter={StaticResource ProgrammingVisible}, Mode=OneWay}"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

ScrollViewer.ScrollChanged="StopScrollChanged"

md:HintAssist.Hint="Название таблицы"

Grid.Column="1"

FontSize="14">

<TextBox.InputBindings>

<KeyBinding Key="Enter"

Command="{Binding NameEnter}"/>

<KeyBinding Key="Down"

Command="{Binding NameEnter}"/>

</TextBox.InputBindings>

<i:Interaction.Triggers>

<i:EventTrigger

EventName="LostFocus">

<i:InvokeCommandAction

Command="{Binding LostFocus}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBox>

<Button

Style="{StaticResource MaterialDesignFlatButton}"

Command="{Binding RemoveElement}"

Grid.Column="2"

Padding="0"

Width="24"

Height="24"

ToolTip="Удалить таблицу">

<md:PackIcon

Kind="TableRemove"/>

</Button>

</Grid>

<ItemsControl

ItemsSource="{Binding Fields, Mode=OneWay}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<local:FieldV/>

</DataTemplate>

</ItemsControl.ItemTemplate>

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</StackPanel>

</Grid>

</UserControl>

96. FieldVM.cs. Работа с полями таблицы.

using BaseMVVM.Command;

using ScriptGen.Abstraction;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGenPlugin.Model;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с полями таблицы

/// </summary>

public class FieldVM : FieldBase

{

/// <summary>

/// Создаёт новое поле таблицы

/// </summary>

public FieldVM()

{

Type = FieldType.Any;

Loaded = new SimpleCommand(SetFocus);

Info.OnSettingsChange += () =>

{

OnPropertyChanged("DataTypeSource");

OnPropertyChanged("ProgrammingSource");

};

}

/// <summary>

/// Загружает поле таблицы

/// </summary>

/// <param name="fieldSave">Сохранённые данные</param>

public FieldVM(FieldSave fieldSave) : this()

{

Load(fieldSave);

DataType = fieldSave.DataType;

ProgrammingType = fieldSave.ProgrammingType;

}

/// <summary>

/// Список SQL типов данных

/// </summary>

public override string[] DataTypeSource

=> Info.SQLTypes;

/// <summary>

/// Список типов данных для языков программирования

/// </summary>

public override string[] ProgrammingTypeSource

=> Info.ProgrammingTypes;

}

}

97. ForeignFieldLineVM.cs. Работа с соединительной линией.

using System.Windows;

using ScriptGen.Abstraction;

using ScriptGen.Interface;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с соединительной линией

/// </summary>

public class ForeignFieldLineVM : LineBase

{

/// <summary>

/// Таблица, являющаяся источником

/// </summary>

private ITable source;

/// <summary>

/// Таблица, являющаяся целью

/// </summary>

private ITable target;

/// <summary>

/// Поле FK, с которым связанна линия

/// </summary>

private readonly IField field;

/// <summary>

/// Создаёт новую линию связи

/// </summary>

/// <param name="source">Таблица, являющаяся источником</param>

/// <param name="target">Таблица, являющаяся целью</param>

/// <param name="field">Поле FK, с которым связанна линия</param>

public ForeignFieldLineVM(ITable source, ITable target, IField field) :

base(source, target)

{

this.source = source;

this.target = target;

this.field = field;

source.OnMarginChanged += OnSourceMove;

target.OnMarginChanged += OnTargetMove;

source.OnRemoveTable += OnRemoveSource;

target.OnRemoveTable += OnRemoveTarget;

field.OnRemoveField += OnRemoveField;

SourceX = source.Margin.Left + 150;

SourceY = source.Margin.Top + 10;

TargetX = target.Margin.Left + 150;

TargetY = target.Margin.Top + 10;

ConnectionX = source.Margin.Left + source.Width + (target.Margin.Left - source.Margin.Left - source.Width) / 2;

}

// <summary>

/// Поле FK, с которым связанна линия

/// </summary>

public override IField Field

=> field;

/// <summary>

/// Координата Y для начальной точки линии

/// </summary>

public override double SourceY

{

get => base.SourceY;

set

{

if (value > Source.Margin.Top + 5 && value < Source.Margin.Top + Source.Height - 5)

base.SourceY = value;

}

}

/// <summary>

/// Координата Y для конечной точки линии

/// </summary>

public override double TargetY

{

get => base.TargetY;

set

{

if (value > Target.Margin.Top + 5 && value < Target.Margin.Top + Target.Height - 5)

base.TargetY = value;

}

}

/// <summary>

/// Происходит при удалении поля

/// </summary>

/// <param name="obj">Поле</param>

private void OnRemoveField(IField obj)

{

target.OnRemoveTable -= OnRemoveTarget;

source.OnRemoveTable -= OnRemoveSource;

RemoveLine(this);

}

/// <summary>

/// Происходит при удаления цели

/// </summary>

/// <param name="obj">Цель</param>

private void OnRemoveTarget(ITable obj)

{

source.OnRemoveTable -= OnRemoveSource;

RemoveLine(this);

}

/// <summary>

/// Происходит при удаления источника

/// </summary>

/// <param name="obj">Источник</param>

private void OnRemoveSource(ITable obj)

{

target.OnRemoveTable -= OnRemoveTarget;

target.RemoveField(field);

RemoveLine(this);

}

/// <summary>

/// Происходит при перемещении источника

/// </summary>

/// <param name="margin">Отступы источника</param>

/// <param name="obj">Источник</param>

private void OnSourceMove(Thickness margin, IMove obj)

{

SourceX = margin.Left + obj.Width / 2;

SourceY += margin.Top - obj.Margin.Top;

}

/// <summary>

/// Происходит при перемещении цели

/// </summary>

/// <param name="margin">Отступы цели</param>

/// <param name="obj">Цель</param>

private void OnTargetMove(Thickness margin, IMove obj)

{

TargetX = margin.Left + obj.Width / 2;

TargetY += margin.Top - obj.Margin.Top;

}

}

}

98. ForeignFieldVM.cs. Работа с внешним ключом.

using System.ComponentModel;

using BaseMVVM.Command;

using ScriptGen.Abstraction;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с внешним ключом

/// </summary>

public class ForeignFieldVM : FieldBase

{

/// <summary>

/// Создаёт новый внешний ключ таблицы

/// </summary>

public ForeignFieldVM()

{

Type = FieldType.FK;

NameEnter = new SimpleCommand(() => NextField?.Invoke());

Loaded = new SimpleCommand(SetFocus);

}

/// <summary>

/// Загружает внешний ключ таблицы

/// </summary>

/// <param name="fieldSave">Сохранённые данные</param>

public ForeignFieldVM(FieldSave fieldSave) : this()

=> Load(fieldSave);

/// <summary>

/// Связывает поле с таблице (возвращает объект линии связи)

/// </summary>

/// <param name="source">Источник</param>

/// <param name="target">Цель</param>

/// <returns>Линия связи</returns>

public ILine AddRefTable(ITable source, ITable target)

{

RefTable = source;

DataType = $"FK({RefTable.Name})";

ProgrammingType = $"FK({RefTable.Name})";

RefTable.PropertyChanged += RefTablePropertyChanged;

return new ForeignFieldLineVM(source, target, this);

}

/// <summary>

/// Загружает связь поля с таблице (возвращает объект линии связи)

/// </summary>

/// <param name="source">Источник</param>

/// <param name="target">Цель</param>

/// <param name="lineSave">Сохраненные данные</param>

/// <returns>Линия связи</returns>

public ILine AddRefTable(ITable source, ITable target, LineSave lineSave)

{

ILine line = AddRefTable(source, target);

line.SourceX = lineSave.SourceX1;

line.SourceY = lineSave.SourceY1;

line.TargetX = lineSave.TargetX1;

line.TargetY = lineSave.TargetY1;

line.ConnectionX = lineSave.ConnectionX1;

return line;

}

/// <summary>

/// Отслеживание изменения имени цели

/// </summary>

private void RefTablePropertyChanged(object sender, PropertyChangedEventArgs e)

{

if (e.PropertyName == "Name")

{

DataType = $"FK({RefTable.Name})";

ProgrammingType = $"FK({RefTable.Name})";

}

}

}

}

99. PageVM.cs. Работа с страницей приложения.

using System.Linq;

using System.Windows;

using ScriptGen.Abstraction;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с страницей приложения

/// </summary>

public class PageVM : PageBase

{

/// <summary>

/// Тип модели

/// </summary>

private ModelType modelType;

/// <summary>

/// Создаёт новую страницу

/// </summary>

public PageVM()

=> Info.OnSettingsChange += () =>

{

OnPropertyChanged("Width");

OnPropertyChanged("Height");

};

/// <summary>

/// Выделенная таблица

/// </summary>

public override ITable SelectTable

{

get => base.SelectTable;

set

{

if (SelectTable != null && SelectTable.Equals(value))

value = null;

if (SelectTable == null || value == null)

{

if (value != null)

value.IsSelect = true;

else if (SelectTable != null)

SelectTable.IsSelect = false;

base.SelectTable = value;

}

else if (value != null)

AddReference(value);

}

}

/// <summary>

/// Тип модели

/// </summary>

public override ModelType TypeModel

{

get => modelType;

set

{

modelType = value;

for (int i = 0; i < Tables.Count; i++)

Tables[i].TypeModel = value;

}

}

/// <summary>

/// Добавляет таблицу

/// </summary>

/// <param name="table">Таблица</param>

public override void AddTable(ITable table)

{

if (table != null)

{

table.Margin = new Thickness(ScrollHorizontal + 50, ScrollVertical + 50, 0, 0);

table.TypeModel = TypeModel;

table.GetNewField = () => new FieldVM() { TypeModel = TypeModel };

SetTableMove(table);

base.AddTable(table);

}

}

/// <summary>

/// Загружает таблицу

/// </summary>

/// <param name="table">Таблица</param>

private void LoadTable(ITable table)

{

table.TypeModel = TypeModel;

table.GetNewField = () => new FieldVM() { TypeModel = TypeModel };

SetTableMove(table);

base.AddTable(table);

}

/// <summary>

/// Загружает данные страницы

/// </summary>

/// <param name="dataBaseSave">Сохранённая информация</param>

public override void Load(DataBaseSave dataBaseSave)

{

base.Load(dataBaseSave);

for (int i = 0; i < dataBaseSave.Tables.Length; i++)

LoadTable(new TableVM(dataBaseSave.Tables[i]));

for (int i = 0; i < dataBaseSave.Lines.Length; i++)

AddReference(dataBaseSave.Lines[i]);

}

/// <summary>

/// Устанавливает ограничения перемещения таблицы

/// </summary>

/// <param name="table">Таблица</param>

private void SetTableMove(ITable table)

{

table.CanWidthChange = (w, \_) => w >= 300 && w <= 500;

table.CanHorizontalChange = (l, o) => l >= 10 && l + o.Width <= 2790;

table.CanVerticalChange = (t, o) => t >= 10 && t + o.Height <= 1565;

table.OnMarginChanged += OnTableMove;

}

/// <summary>

/// Происходит при движении таблицы

/// </summary>

/// <param name="margin">Отступы таблицы</param>

/// <param name="obj">Таблица</param>

private void OnTableMove(Thickness margin, IMove obj)

{

if (margin.Left + obj.Width > ScrollHorizontal + ActualWidth)

ScrollHorizontal += 5;

else if (margin.Left < ScrollHorizontal)

ScrollHorizontal -= 5;

if (margin.Top + obj.Height > ScrollVertical + ActualHeight)

ScrollVertical += 5;

else if (margin.Top < ScrollVertical)

ScrollVertical -= 5;

}

/// <summary>

/// Связывает таблицу с другой

/// </summary>

/// <param name="target">Цель</param>

private void AddReference(ITable target)

{

ForeignFieldVM field = new ForeignFieldVM()

{

TypeModel = TypeModel

};

AddLine(field.AddRefTable(SelectTable, target));

target.AddField(field);

SelectTable = null;

}

/// <summary>

/// Загружает связь

/// </summary>

/// <param name="lineSave">Сохранённые данные</param>

private void AddReference(LineSave lineSave)

{

ForeignFieldVM field = new ForeignFieldVM(lineSave.Field);

ITable source = Tables.FirstOrDefault(t => t.Id == lineSave.SourceId);

ITable target = Tables.FirstOrDefault(t => t.Id == lineSave.TargetId);

AddLine(field.AddRefTable(source, target, lineSave));

target.AddField(field);

}

/// <summary>

/// Добавляет линию

/// </summary>

/// <param name="line">Линия</param>

private void AddLine(ILine line)

{

line.OnRemoveLine += (l) => Lines.Remove(l);

Lines.Add(line);

}

}

}

100. PrimaryFieldVM.cs. Работа с первичным ключом.

using ScriptGen.Abstraction;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGenPlugin.Model;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с первичным ключом

/// </summary>

public class PrimaryFieldVM : FieldBase

{

/// <summary>

/// Создаёт первичный ключ таблицы

/// </summary>

public PrimaryFieldVM()

{

Type = FieldType.PK;

IsUnique = true;

}

/// <summary>

/// Загружает первичный ключ таблицы

/// </summary>

/// <param name="fieldSave">Сохранённые данные</param>

public PrimaryFieldVM(FieldSave fieldSave) : this()

{

Load(fieldSave);

DataType = fieldSave.DataType;

ProgrammingType = fieldSave.ProgrammingType;

}

/// <summary>

/// Список SQL типов данных

/// </summary>

public override string[] DataTypeSource

=> Info.SQLTypes;

/// <summary>

/// Список типов данных для языков программирования

/// </summary>

public override string[] ProgrammingTypeSource

=> Info.ProgrammingTypes;

}

}

101. ScriptGenVM.cs. Работа с приложение.

using System.Collections.ObjectModel;

using BaseMVVM.Abstraction;

using BaseMVVM.Command;

using Explorer;

using ScriptGen.API;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Faker.ViewModel;

using ScriptGen.Generator;

using ScriptGen.Interface;

using ScriptGen.Model;

using ScriptGen.Settings;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с приложение

/// </summary>

public class ScriptGenVM : ViewModelBase

{

private bool havePages;

private bool havePlugins;

private ScriptGenM scriptGen = new ScriptGenM();

private readonly FileDialog explorer = new FileDialog();

private readonly SettingsDialog settings = new SettingsDialog();

public ScriptGenVM()

{

OpenSettings = new SimpleCommand(settings.Open);

OpenPage = new SimpleCommand(LoadPage);

SavePage = new SimpleCommand(SaveSelectPage, () => SelectPage != null);

AddPage = new SimpleCommand(NewPage);

AddTable = new SimpleCommand(NewTable, () => SelectPage != null);

CheckError = new SimpleCommand(() => IsError(), () => SelectPage != null && SelectPage.Tables.Count > 0);

GenerateScript = new SimpleCommand(Generate, () => SelectPage != null && SelectPage.Tables.Count > 0);

FakerContext = new FakerVM();

PluginManager.CommonAPI.GetError = () => SelectPage != null && SelectPage.Tables.Count > 0

? IsError()

: false;

PluginManager.CommonAPI.GetTables = () => SelectPage != null && SelectPage.Tables.Count > 0

? DataConverter.IPageToInfo(SelectPage)

: null;

PluginManager.OnPluginsUpdate = () => HavePlugins = Plugins.Count > 0;

HavePlugins = Plugins.Count > 0;

FileDialog.OnInfo = StatusBarManagerVM.ExplorerInfo;

FileDialog.OnError = StatusBarManagerVM.ExplorerError;

}

public SimpleCommand OpenSettings { get; }

public SimpleCommand OpenPage { get; }

public SimpleCommand SavePage { get; }

public SimpleCommand AddPage { get; }

public SimpleCommand AddTable { get; }

public SimpleCommand CheckError { get; }

public SimpleCommand GenerateScript { get; }

public FakerVM FakerContext { get; }

public ObservableCollection<IPage> Pages

=> scriptGen.Pages;

public ObservableCollection<PluginData> Plugins

=> PluginManager.Plugins;

public StatusBarManagerVM StatusBarManager { get; }

= new StatusBarManagerVM();

public double ActualWidth { get; set; }

public double ActualHeight { get; set; }

public bool HavePages

{

get => havePages;

set

{

havePages = value;

OnPropertyChanged();

}

}

public bool HavePlugins

{

get => havePlugins;

set

{

havePlugins = value;

OnPropertyChanged();

}

}

public IPage SelectPage

{

get => scriptGen.SelectPage;

set

{

if (scriptGen.SelectPage == null || !scriptGen.SelectPage.Equals(value))

{

scriptGen.SelectPage = value;

if (value != null)

{

SelectPage.ActualWidth = ActualWidth;

SelectPage.ActualHeight = ActualHeight;

}

OnPropertyChanged();

}

}

}

public ModelType TypeModel

{

get => scriptGen.ModelType;

set

{

scriptGen.ModelType = value;

for (int i = 0; i < Pages.Count; i++)

Pages[i].TypeModel = value;

}

}

private void NewTable()

=> SelectPage?.AddTable(new TableVM(new PrimaryFieldVM()));

private void NewPage()

{

IPage page = new PageVM()

{

Name = "Новая страница",

TypeModel = TypeModel,

ActualWidth = ActualWidth,

ActualHeight = ActualHeight

};

page.OnRemovePage += RemovePage;

page.OnSelectPage += (p) => SelectPage = p;

Pages.Add(page);

HavePages = true;

SelectLast();

}

private void RemovePage(IPage page)

{

if (SelectPage != null && SelectPage.Equals(page))

SelectPage = null;

Pages.Remove(page);

HavePages = Pages.Count > 0;

if (SelectPage == null && HavePages)

SelectLast();

}

private void SelectLast()

{

int i;

for (i = 0; i < Pages.Count - 1; i++)

Pages[i].IsSelect = false;

SelectPage = Pages[i];

SelectPage.IsSelect = true;

}

private void Select(IPage page)

{

if (page == null)

return;

for (int i = 0; i < Pages.Count; i++)

Pages[i].IsSelect = false;

SelectPage = page;

SelectPage.IsSelect = true;

}

private bool IsError()

{

SelectPage.SetError();

if (SelectPage.IsError)

{

IPage page = SelectPage;

SelectPage.OnRemovePage += (t) => page = null;

StatusBarManagerVM.Error($"На схеме \"{SelectPage.Name}\" найдены ошибки(-а)!", "перейти",

new SimpleCommand(() => Select(page), () => page != null));

return true;

}

StatusBarManagerVM.Message("Ошибок не обнаружено!");

return false;

}

private void SaveSelectPage()

{

DataBaseSave dataBaseSave = DataConverter.IPageToSave(SelectPage);

if (explorer.Open(ExplorerSelectType.All, ExplorerType.Save, ".xml", ".txt", ""))

DataBaseFileWork.Save(dataBaseSave, explorer.SelectPath);

}

private void LoadPage()

{

if (explorer.Open(ExplorerSelectType.File, ExplorerType.Open, ".xml")

&& DataBaseFileWork.Load(explorer.SelectPath, out DataBaseSave dataBase))

{

NewPage();

SelectPage.Load(dataBase);

}

}

private void Generate()

{

if (IsError())

return;

if (explorer.Open(ExplorerSelectType.Folder, ExplorerType.Save))

ScriptGenerator.Generate(DataConverter.IPageToInfo(SelectPage), explorer.SelectPath);

}

}

}

102. StatusBarManagerVM.cs. Осуществляет вывод сообщений.

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using BaseMVVM.Command;

using Explorer;

using MaterialDesignThemes.Wpf;

using ScriptGen.Common;

using ScriptGen.Model;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Осуществляет вывод сообщений

/// </summary>

public class StatusBarManagerVM

{

public static PackIconKind ErrorIcon

= PackIconKind.CloseBox;

public static PackIconKind InfoIcon

= PackIconKind.WarningBox;

public static PackIconKind MessageIcon

= PackIconKind.CheckboxMarked;

private static string data;

private static PackIconKind kind;

public static SimpleCommand ClearHistory { get; }

= new SimpleCommand(Clear, () => History.Count > 0);

public static ObservableCollection<MessageM> History { get; }

= new ObservableCollection<MessageM>();

public static string Data

{

get => data;

private set

{

data = value;

OnStaticPropertyChanged();

}

}

public static PackIconKind Kind

{

get => kind;

private set

{

kind = value;

OnStaticPropertyChanged();

}

}

private static void SetOption(string message, string option, PackIconKind kind)

{

Data = message;

Kind = kind;

message = Verification.ConcatString(message, option);

History.Add(new MessageM(message, kind));

}

private static void SetOption(string message, string option, PackIconKind kind, string actionTitle, SimpleCommand action)

{

Data = message;

Kind = kind;

message = Verification.ConcatString(message, option);

History.Add(new MessageM(message, kind, actionTitle, action));

}

private static void SetOption(string message, PackIconKind kind)

=> SetOption(message, null, kind);

private static void SetOption(string message, PackIconKind kind, string actionTitle, SimpleCommand action)

=> SetOption(message, null, kind, actionTitle, action);

public static void Error(string message)

=> SetOption(message, ErrorIcon);

public static void Error(string message, string option)

=> SetOption(message, option, ErrorIcon);

public static void Error(string message, string actionTitle, SimpleCommand action)

=> SetOption(message, ErrorIcon, actionTitle, action);

public static void Error(string message, string option, string actionTitle, SimpleCommand action)

=> SetOption(message, option, ErrorIcon, actionTitle, action);

public static void Info(string message)

=> SetOption(message, InfoIcon);

public static void Info(string message, string option)

=> SetOption(message, option, InfoIcon);

public static void Info(string message, string actionTitle, SimpleCommand action)

=> SetOption(message, InfoIcon, actionTitle, action);

public static void Info(string message, string option, string actionTitle, SimpleCommand action)

=> SetOption(message, option, InfoIcon, actionTitle, action);

public static void Message(string message)

=> SetOption(message, MessageIcon);

public static void Message(string message, string option)

=> SetOption(message, option, MessageIcon);

public static void Message(string message, string actionTitle, SimpleCommand action)

=> SetOption(message, MessageIcon, actionTitle, action);

public static void Message(string message, string option, string actionTitle, SimpleCommand action)

=> SetOption(message, option, MessageIcon, actionTitle, action);

public static void ExplorerError(ExplorerException e)

{

if (!string.IsNullOrEmpty(e.Message) && !string.IsNullOrEmpty(e.Option) && e.Action != null)

Error(e.Message, e.Option, e.ActionTitle, new SimpleCommand(e.Action));

else if (!string.IsNullOrEmpty(e.Message) && !string.IsNullOrEmpty(e.Option))

Error(e.Message, e.Option);

}

public static void ExplorerInfo(string message, string option)

{

if (!string.IsNullOrEmpty(message) && !string.IsNullOrEmpty(option))

Info(message, option);

else if (!string.IsNullOrEmpty(message))

Info(message);

}

private static void Clear()

{

History.Clear();

Message("История очищена!");

}

public static event EventHandler<PropertyChangedEventArgs> StaticPropertyChanged;

private static void OnStaticPropertyChanged([CallerMemberName] string name = "")

=> StaticPropertyChanged?.Invoke(null, new PropertyChangedEventArgs(name));

}

}

103. TableVM.cs. Работа с таблицей.

using System.Linq;

using BaseMVVM.Command;

using ScriptGen.Abstraction;

using ScriptGen.Common;

using ScriptGen.Common.SaveModel;

using ScriptGen.Interface;

using ScriptGenPlugin.Model;

namespace ScriptGen.ViewModel

{

/// <summary>

/// Работа с таблицей

/// </summary>

public class TableVM : TableBase

{

/// <summary>

/// Создаёт новую таблицу

/// </summary>

/// <param name="startField">Начальное поле</param>

public TableVM(IField startField)

{

startField.PrevField = SetFocus;

AddField(startField);

NameDown = NameEnter = new SimpleCommand(Fields[0].SetFocus);

}

// <summary>

/// Загружает таблицу

/// </summary>

/// <param name="startField">Сохранённые данные</param>

public TableVM(TableSave tableSave) : this(new PrimaryFieldVM(tableSave.Fields[0]))

{

Load(tableSave);

FieldSave[] fields = tableSave.Fields.Where((f) => f.Type == FieldType.Any).ToArray();

for (int i = 0; i < fields.Length; i++)

AddField(new FieldVM(fields[i]));

}

/// <summary>

/// Тип модели (физическая/логическая/программирование)

/// </summary>

public override ModelType TypeModel

{

get => base.TypeModel;

set

{

base.TypeModel = value;

for (int i = 0; i < Fields.Count; i++)

Fields[i].TypeModel = value;

}

}

/// <summary>

/// Обновляет команды полей

/// </summary>

private void RefreshFieldCommand()

{

int i;

for (i = 0; i < Fields.Count - 1; i++)

{

Fields[i].NextField = Fields[i + 1].SetFocus;

Fields[i + 1].PrevField = Fields[i].SetFocus;

}

Fields[i].NextField = () => AddField(GetNewField?.Invoke());

Height = 50 + 42 \* Fields.Count;

}

/// <summary>

/// Сортирует поля таблицы

/// </summary>

private void LastFieldSort()

{

int index = Fields.Count(f => f.Type != FieldType.Any) - 1;

if (index < Fields.Count && index > 0)

Fields.Move(Fields.Count - 1, index);

}

/// <summary>

/// Добавляет поле

/// </summary>

/// <param name="field">Поле</param>

public override void AddField(IField field)

{

if (field != null)

{

base.AddField(field);

if (field.Type != FieldType.Any)

LastFieldSort();

RefreshFieldCommand();

}

}

/// <summary>

/// Удаляет поле

/// </summary>

/// <param name="field">Поле</param>

public override void RemoveField(IField field)

{

if (field != null)

{

base.RemoveField(field);

RefreshFieldCommand();

}

}

}

}

ПРИЛОЖЕНИЕ Г. Руководство пользователя

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

АННОТАЦИЯ

В приложении «Руководство пользователя» приведена инструкция эксплуатации программы пользователем.

Приложение состоит из нескольких разделов.

В разделе «Установка системы» приведена инструкция по установки программного обеспечения «ScriptGen».

В разделе «Работа системы» приведена инструкция по использовании программного продукта.

В разделе «Создание шаблона для генерации скриптов» приведена инструкция по написанию своего шаблона для генерации скриптов по модели.

В разделе «Управление шаблонами генератора случайных значений» приведена инструкция для создания маски случайных значений и создания основного шаблона генерации.

В разделе «Деинсталляция системы» приведена инструкция по удалению программного обеспечения «ScriptGen».

В разделе «Сообщения» показаны возможные сообщения системы.

УСТАНОВКА СИСТЕМЫ

Для установки программного обеспечения «ScriptGen» необходимо запустить установщик и следовать инструкции на экране:

1. Выбрать папку для установки программы.

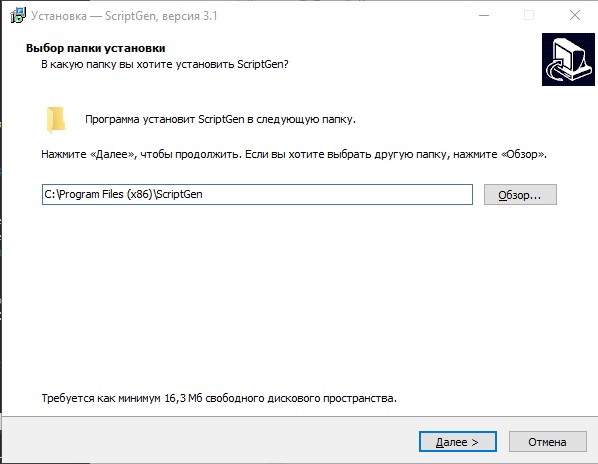


Рисунок 1 – Выбор папки для установки

1. Указать необходимо ли создать ярлык на рабочем столе.

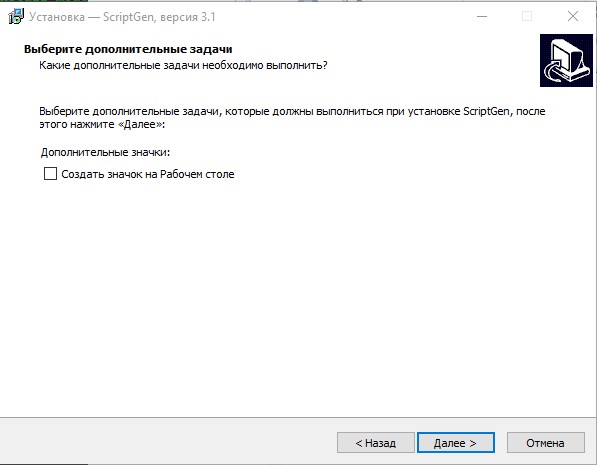


Рисунок 2 – Создание ярлыка программы

1. Подтвердить установку.

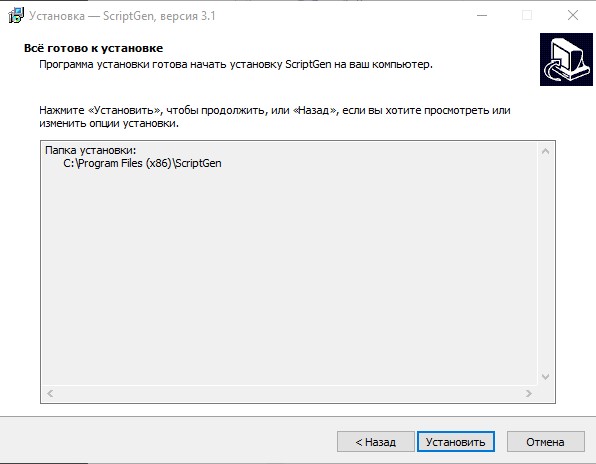


Рисунок 3 – Подтверждение установки

1. Дождаться завершения установки и нажать «Завершить».

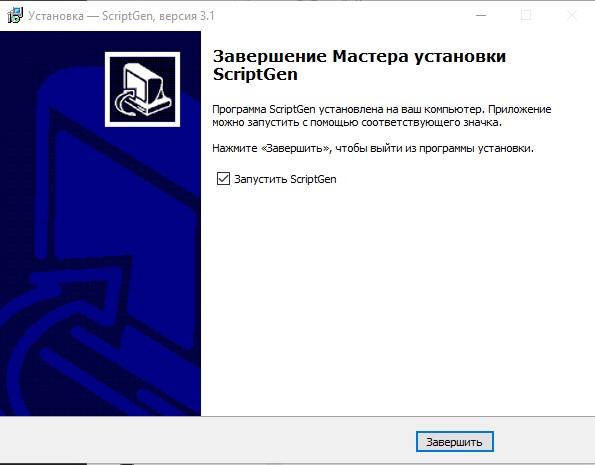


Рисунок 4 - Завершение установки

РАБОТА СИСТЕМЫ

После запуска приложения пользователь может создать модель при помощи команды меню.

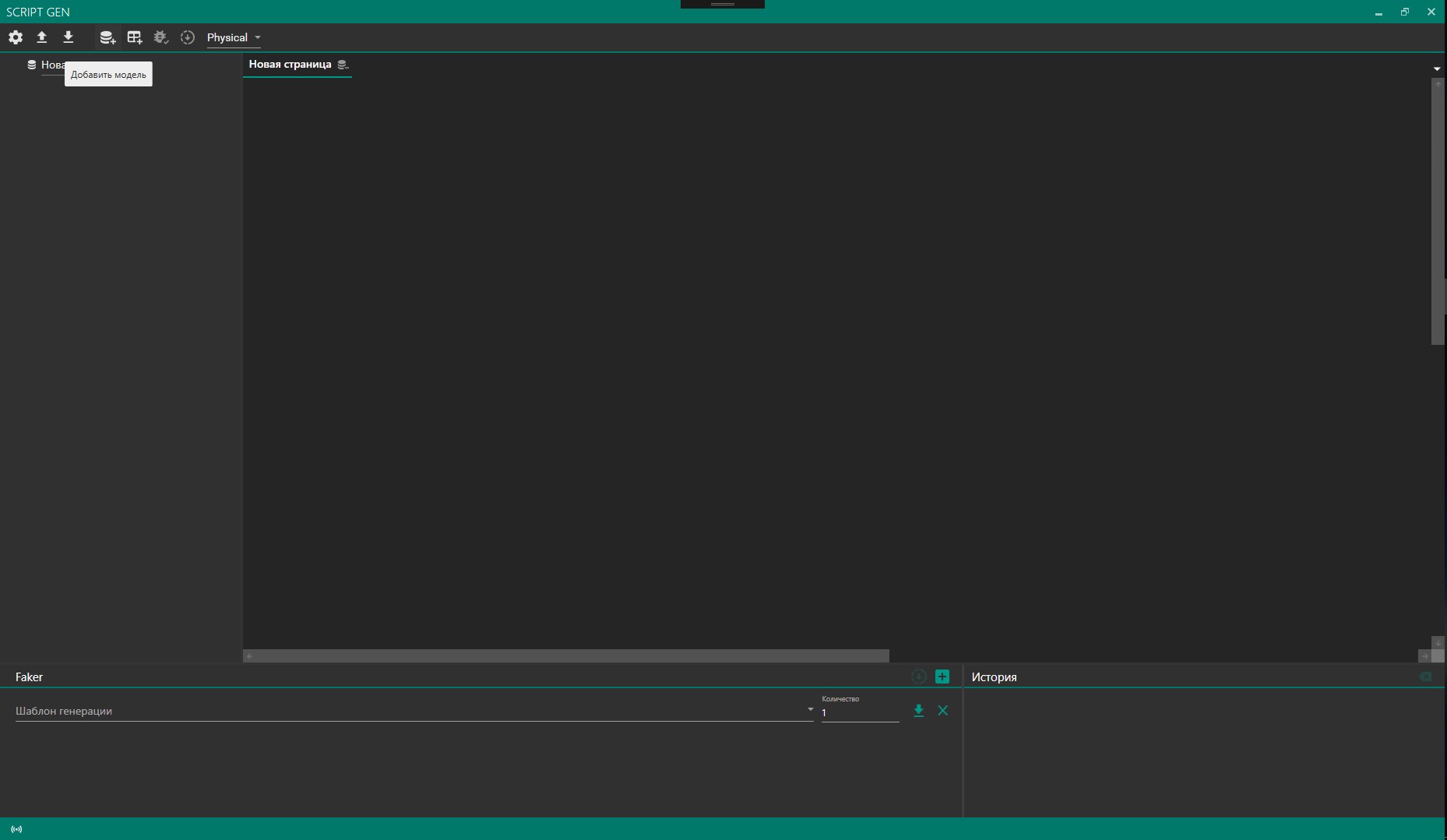


Рисунок 5 – Создание модели

Создание сущностей происходит при помощи команды меню, после используя клавиатуру можно заполнить данные сущности, для создание нового атрибута сущности необходимо нажать «Enter» в поле ввода названия или типа данных последнего атрибута.

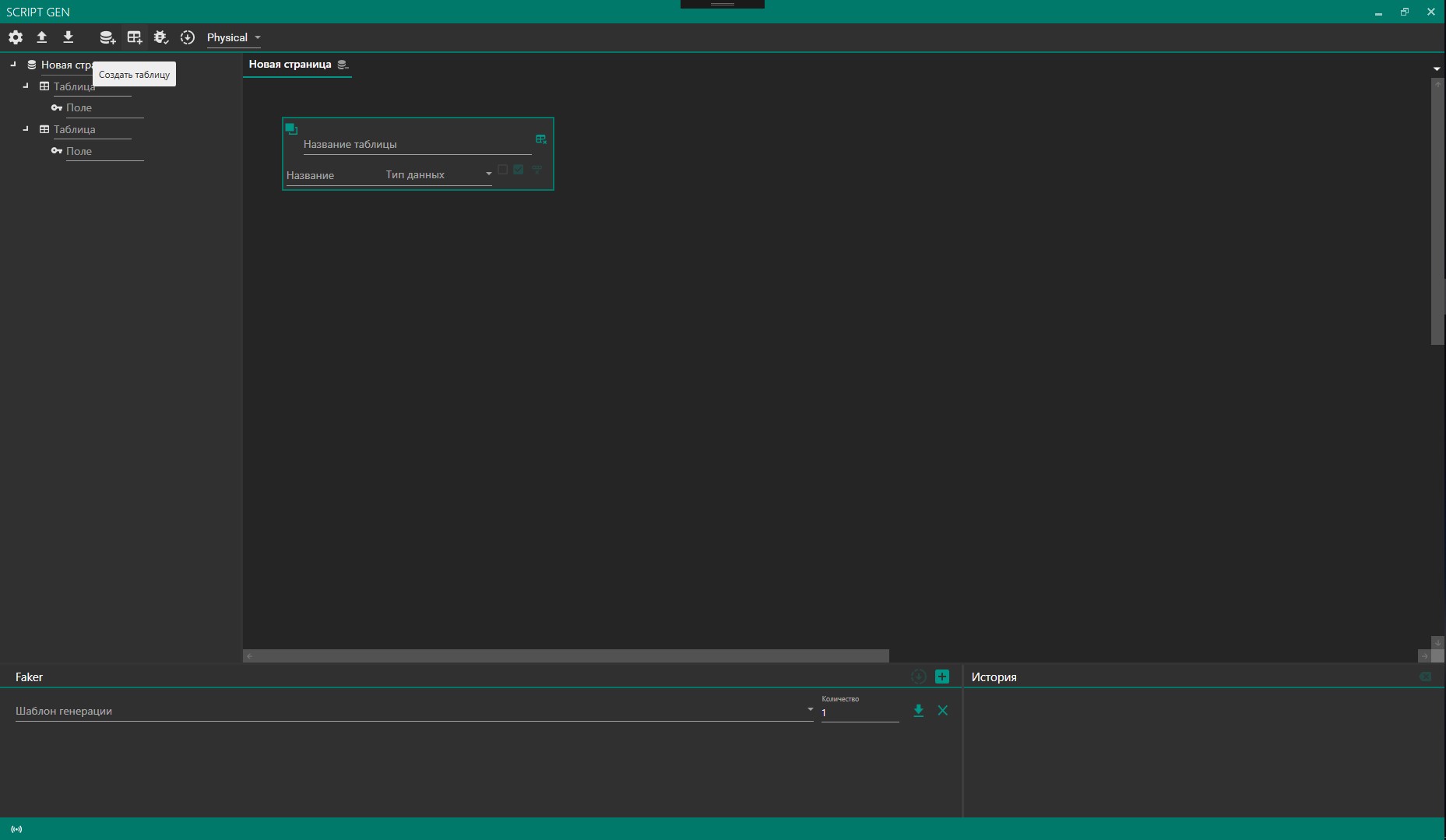


Рисунок 6 – Создание сущности (таблицы)

Для проверки ошибок на модели необходимо нажать на соответствующую кнопку меню. Также проверка ошибок происходит перед началом генерации скриптов. Результат проверки отобразится в поле для сообщений, а при наличии ошибок они обозначатся в самой модели.

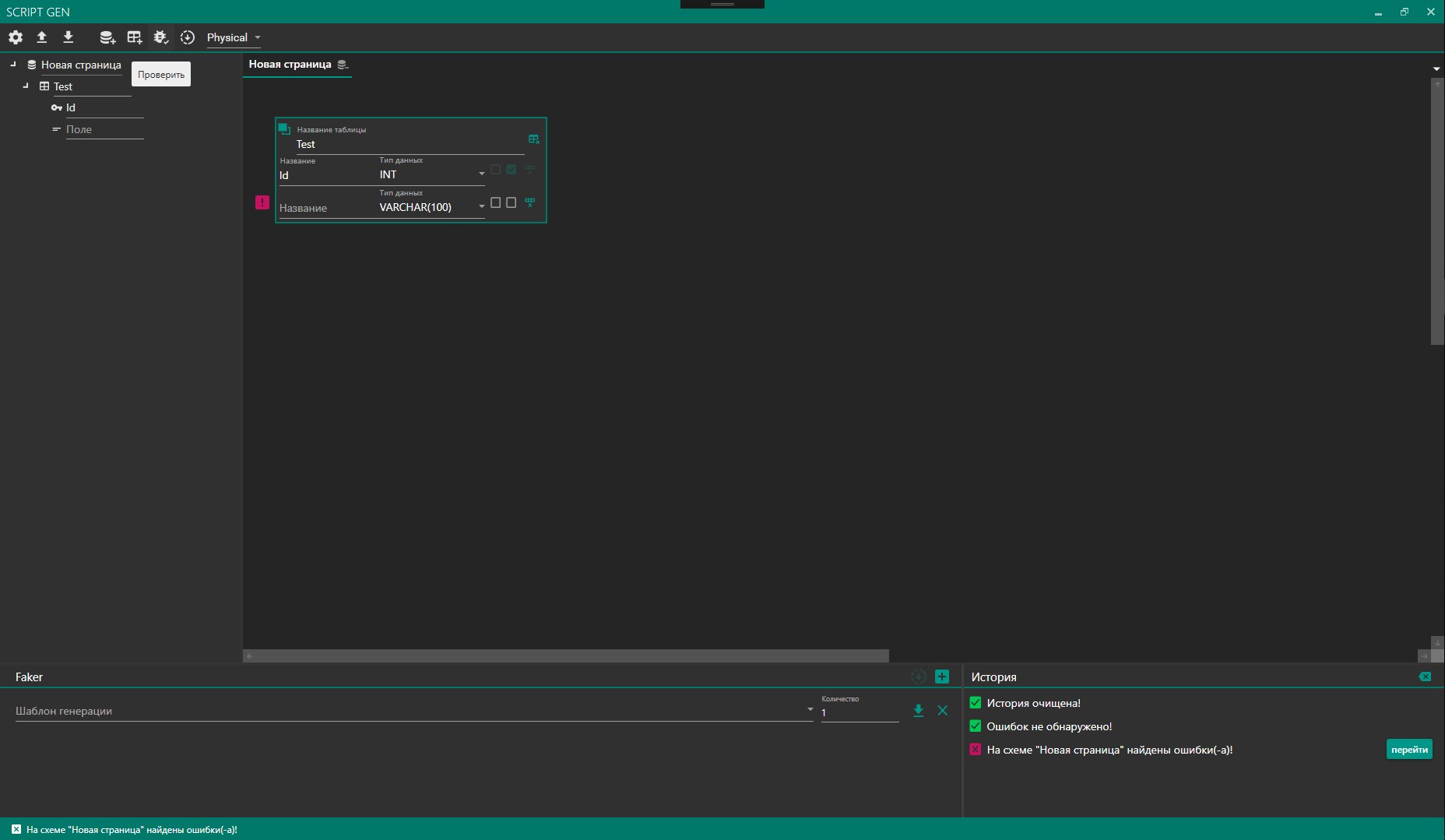


Рисунок 7 – Проверка ошибок модели

Для генерации скриптов необходимо воспользоваться соответствующей кнопкой меню. В случае наличия ошибок в модели генерация не произойдёт, ошибки обозначатся в модели. В случае отсутствия ошибок программа предложит выбрать путь для сохранения, после чего сгенерирует скрипты.

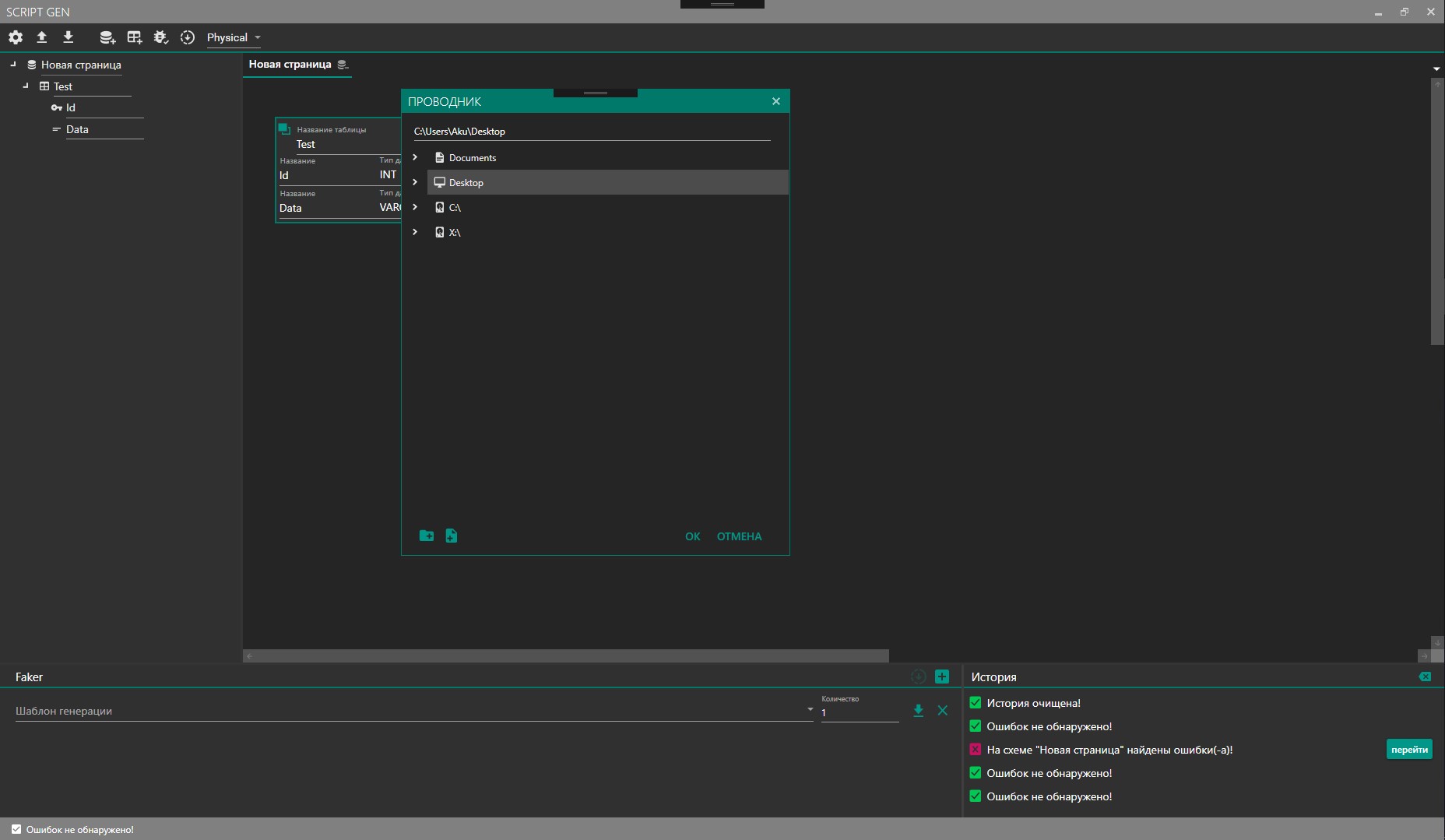


Рисунок 8 – Генерация скриптов

Для переключения типа модели (логическая, физическая, дополнительная) можно использовать выпадающий список вверху экрана.

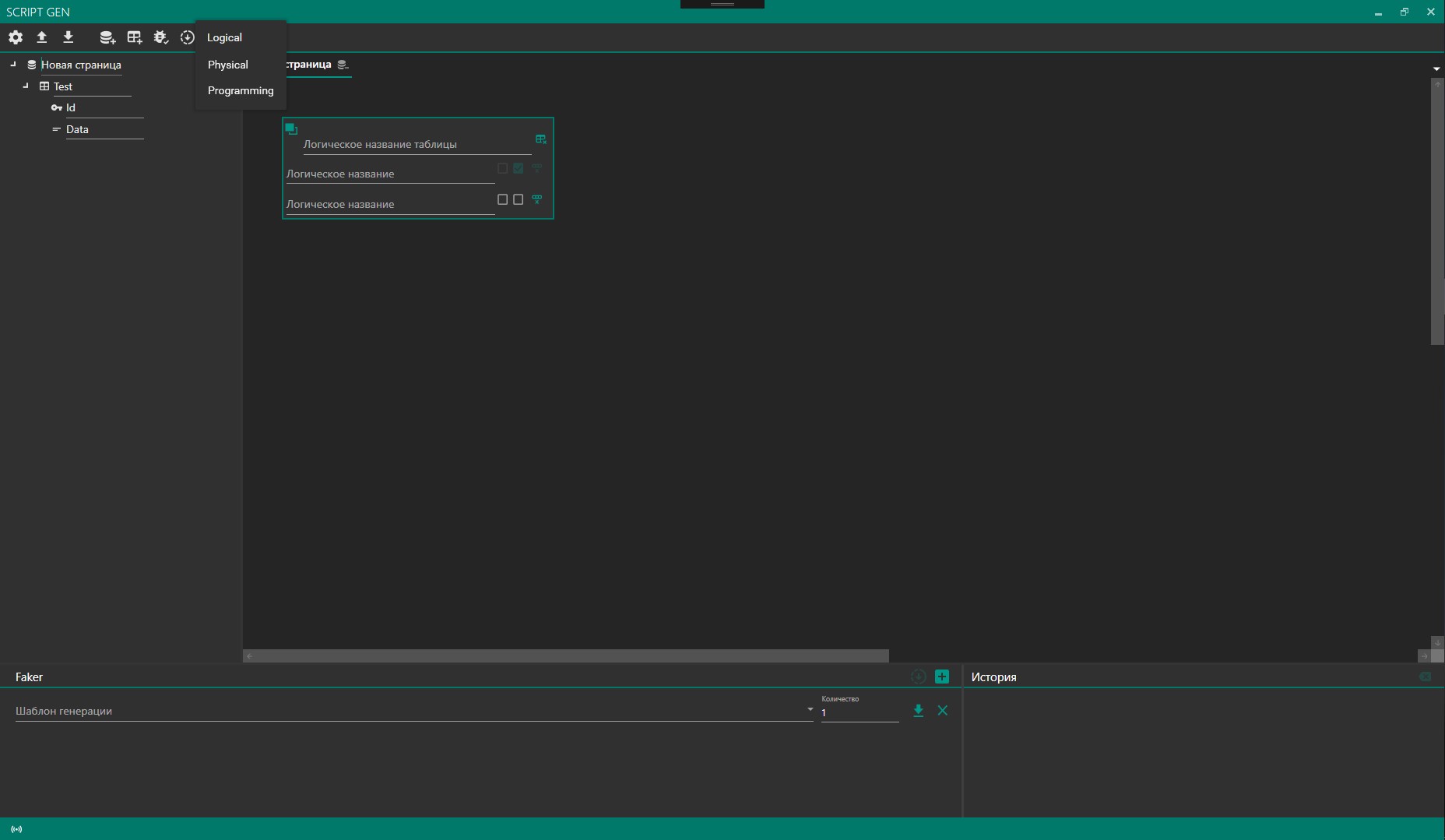


Рисунок 9 – Переключение типа модели

В левой части меню находятся кнопки сохранения и загрузки модели, которые выводят диалог с выбором пути, после чего осуществляют необходимые действия. Рядом находится кнопка, при нажатии на которую появляется список плагинов приложения для запуска их действий.

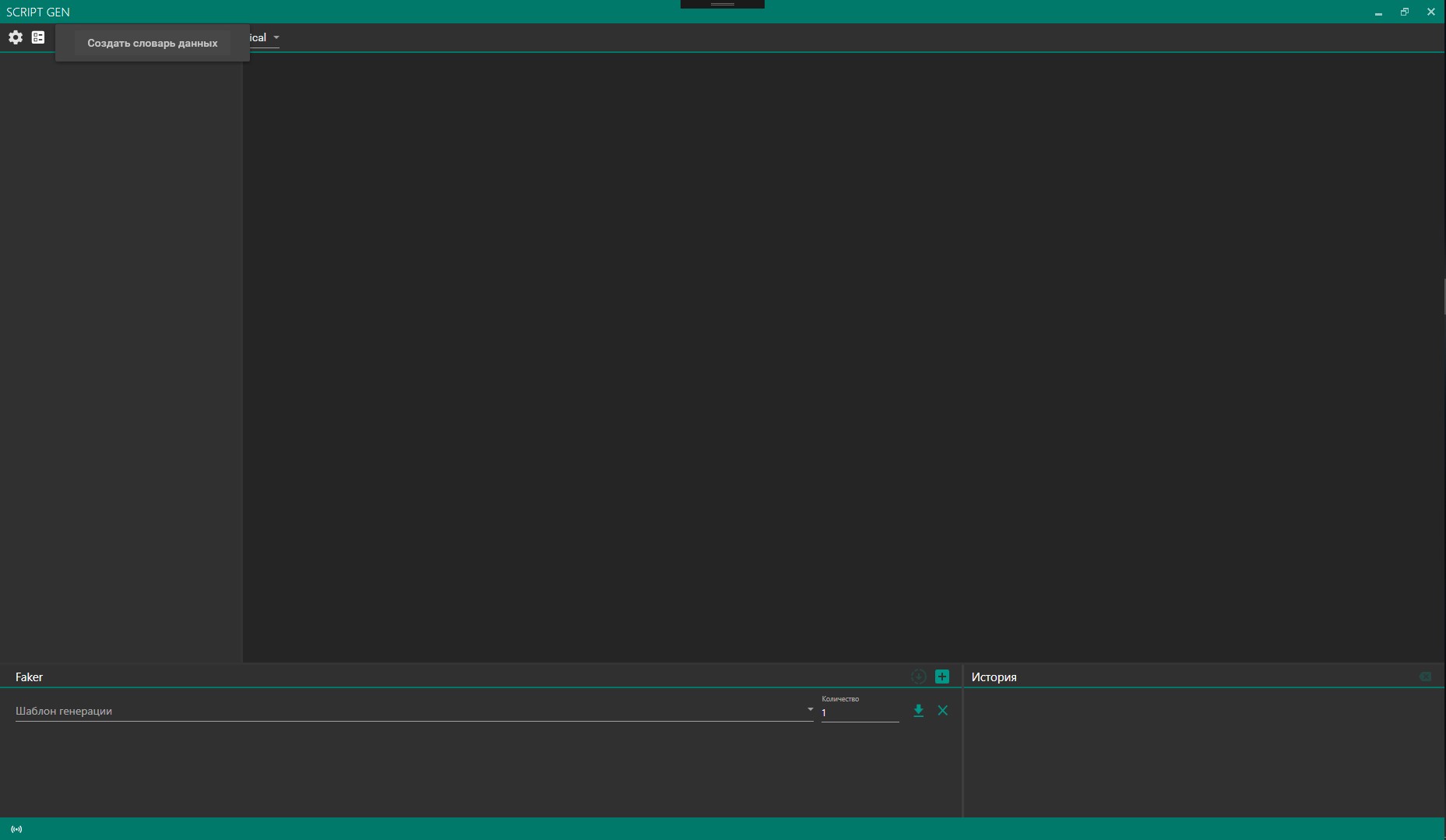


Рисунок 10 – Список плагинов

При нажатии на кнопку вызова настроек появится новое окно с настройками приложения. В общих настройках, с помощью трёх выпадающих списков производится выбор шаблонов и экземпляров пользовательских настроек приложения. А также добавлять плагины при помощи соответственной кнопки, для этого необходимо указать путь к файлу конфигурации плагина в соответствующем меню.

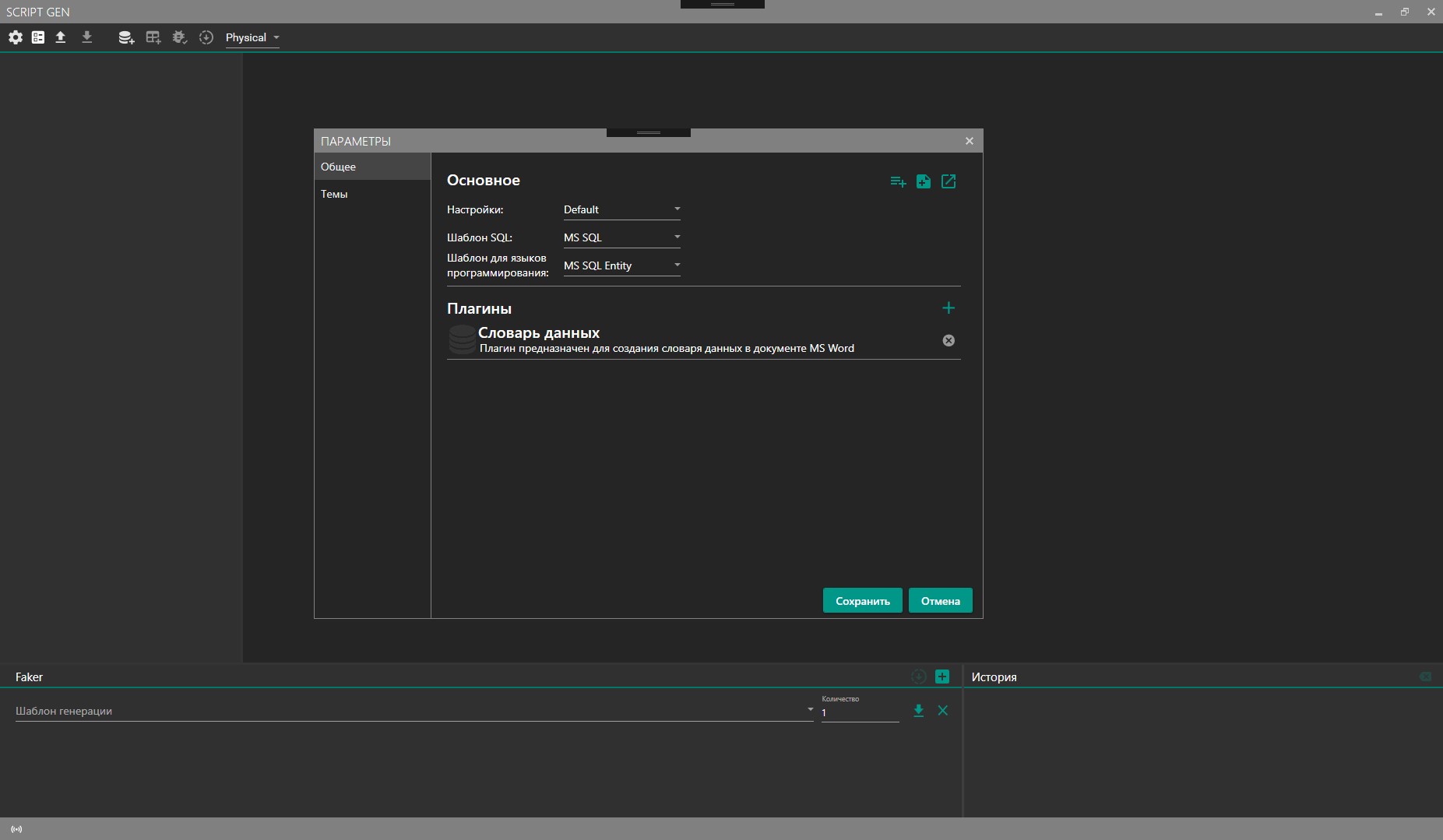


Рисунок 11 - Общие настройки

В списке слева можно перейти на вкладку «Темы» для настройки темы приложения. С помощью выпадающего списка можно переключить тёмную и светлую темы, а наживая на цвета – палитру цветов приложения.

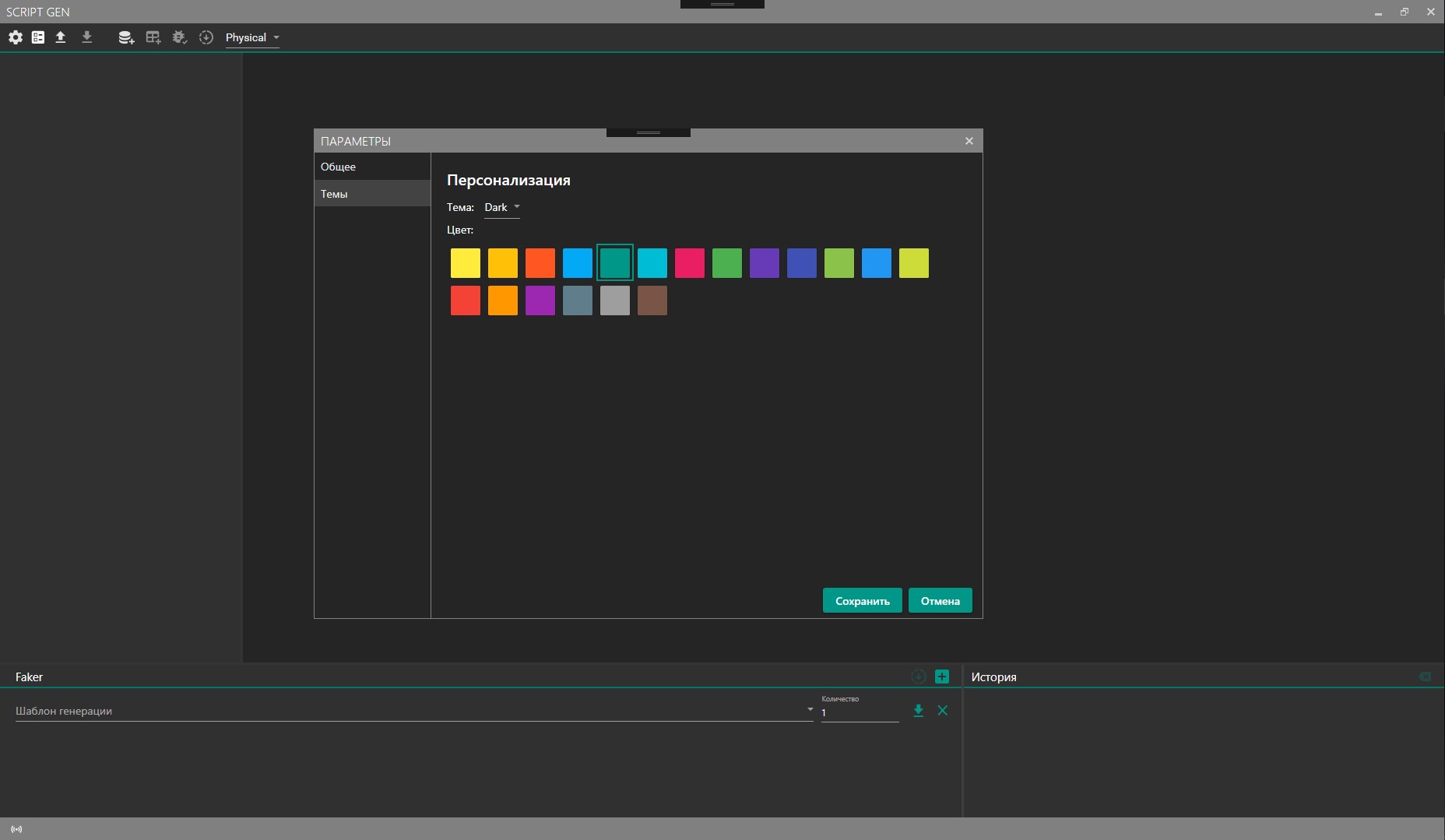


Рисунок 12 – Настройка темы приложения

Изменение настроек вступит в силу при нажатии на кнопку «Сохранить» и сбросится в предыдущее состояние при нажатии на кнопку «Отмена».

Для генерации случайных данных необходимо ввести шаблон генерации и необходимое количество шаблонов случайных параметров. Для добавления шаблонов случайных параметров используется специальная кнопка в правом верхнем углу. Шаблон можно ввести вручную или выбрать из существующих при помощи выпадающего меню. После необходимо указать количество результатов и запустить генерацию при помощи соответствующей кнопки, при нажатии на которую программа предложит указать файл для сохранения результата.

Принцип написания шаблонов случайных значений приведен в пункте «Управление шаблонами генератора случайных значений».

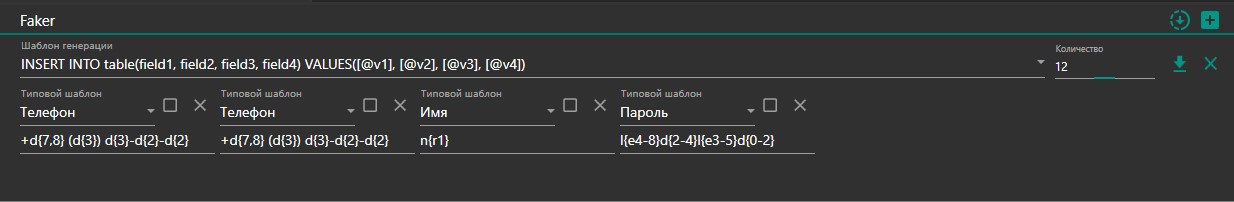


Рисунок 13 – Генератор случайных значений

СОЗДАНИЕ ШАБЛОНА ДЛЯ ГЕНЕРАЦИИ СКРИПТОВ

Для создания шаблона необходимо сначала заполнить общую информацию, указав тип шаблона и его название.

<TemplateData>

<Name>Название шаблона</Name>

<Type>Тип шаблона (SQL или Programming)</Type>

<Template>Шаблон</Template>

</TemplateData>

Далее необходимо создать сам шаблон, который представлен текстом со вставками команд и констант для получения данных о модели.

Команды должны находится в специальном блоке, который начинается с «{{» и заканчивается на «}}».

«write()» - выводит текст или значения констант.

К константам относятся два объекта: поле (field) и таблица (table), которые содержат ряд дополнительных значений, обращение к которым происходит через точку, а также nl{n} – выводит n новых строк и tab{n} – выводит n\*4 пробелов.

Возможные дополнительные значения для поля:

1. «name» – имя поля;
2. «fullname» - полное имя поля;
3. «logicalname» - логическое имя поля;
4. «programmingname» - дополнительное имя поля;
5. «datatype» - тип данных поля;
6. «programmingtype» - дополнительный тип данных поля;
7. «reftable» - объект таблицы, на которое ссылается поле (только для внешнего ключа).

Возможные дополнительные значения для таблицы:

1. «name» – имя поля;
2. «logicalname» - логическое имя поля;
3. «programmingname» - дополнительное имя поля;
4. «id» - объект поля, которое является первичным ключом для данной таблицы.

Например, для вывода имени первичного ключа таблицы можно написать: {{ write(table.id.name) }}

Также при выводе значений можно предоставить альтернативный вариант, который будет использован в случае пустого основного значения. Задаётся альтернативный вариант с помощью знака вопроса. Например, для вывода названия поля, если логическое название не заполнено можно написать: {{ write(field.logicalname?field.name) }}

«if()» - обрабатывает блок кода если выполняется условие.

В качестве параметров может выступать одна из следующих констант:

1. «field.isfk» - является ли поле внешним ключом;
2. «field.ispk» - является ли поле первичным ключом;
3. «field.isany» - является ли поле обычным (не ключом);
4. «field.isnull» - может ли поле быть равно null;
5. «field.isunique» - является ли поле уникальным.

Все константы можно инвертировать, поставив перед ними восклицательный знак.

После блока «if» можно использовать команду «else» или «elseif()» для обработки блока кода, если условие не выполняется.

Например, {{ if (field.isnull) { write(NULL) } else { write(NOT NULL) } }}

«for()» - обрабатывает блок кода для всех значений параметра.

В качестве параметров может выступать одна из следующих констант:

1. «table.fields» - все поля таблицы;
2. «table.reftables» - все таблицы, которые имеют внешний ключ на текущую;
3. «tables» - все таблицы модели.

«rec()» - определяет блок кода, который может быть рекурсивно запущен для обработки модели.

В качестве параметров может передаваться слово «start», объявляющее начало блока рекурсии, либо константа «field.reftables» для выполнения блока для связанной таблицы.

«removelast(n)» - удаляет n последних символов.

«replace(v1,v2)» - заменяет все вхождения v1 на v2.

«once» - выполняет блок кода только однажды для всей модели.

«set(key,value)» - сохраняет значение value с именем key. Для обращения к сохраннённому значению в команде «write()» следует использовать команду «get(key)». Например, {{ set(test,test) write(get[test]) }}

УПРАВЛЕНИЕ ШАБЛОНАМИ ГЕНЕРАТОРА СЛУЧАЙНЫХ ЗНАЧЕНИЙ

Для генерации случайных значений необходимо создать общий шаблон генерации, который может быть представлен любыми символами. Для обозначения места для вставки случайного параметра используется следующая конструкция: @v[index], где index – номер параметра с отсчётов от единицы (например, @v1).

После этого с помощью кнопки можно добавить шаблоны параметров, которые будут подставлены в основной шаблон.

Для написания шаблона параметра можно использовать любые символы, которые будут вставлены без изменений, а также специальные управляющие символы, которые имеют следующую конструкцию [управляющий символ] {[параметры]}.

В программе поддерживаются следующие управляющие символы:

1. «d» - отвечает за генерацию цифр;
2. «l» - отвечает за генерацию букв;
3. «w» - отвечает за генерацию слов;
4. «n» - отвечает за генерацию имён;
5. «s» - отвечает за генерацию фамилий.

Параметры всех управляющих символов, кроме цифр, могут начинаться с одной из букв «e» или «r» для указания языка генерации (русский или английский).

Далее у всех управляющих символов могут быть три вида параметров:

1. Одиночное число – указывает количество генерируемых значений. Например, «d{3}» сгенерирует три случайных цифры.
2. Диапазон – задаётся двумя числами через «-», количество значений будет находится в этом диапазоне.

Например, «l{e3-6}» сгенерирует от трёх до шести английских букв.

1. Список значений – перечисление значений через запятую.

Например, «l{q,w,e,r}» сгенерирует одну из четырёх букв.

ДЕИНСТАЛЛЯЦИЯ СИСТЕМЫ

Для деинсталляции программного обеспечения «ScriptGen» необходимо:

1. Запустить «Панель управления» - «Программы и компоненты» и найти в списке приложение «ScriptGen».

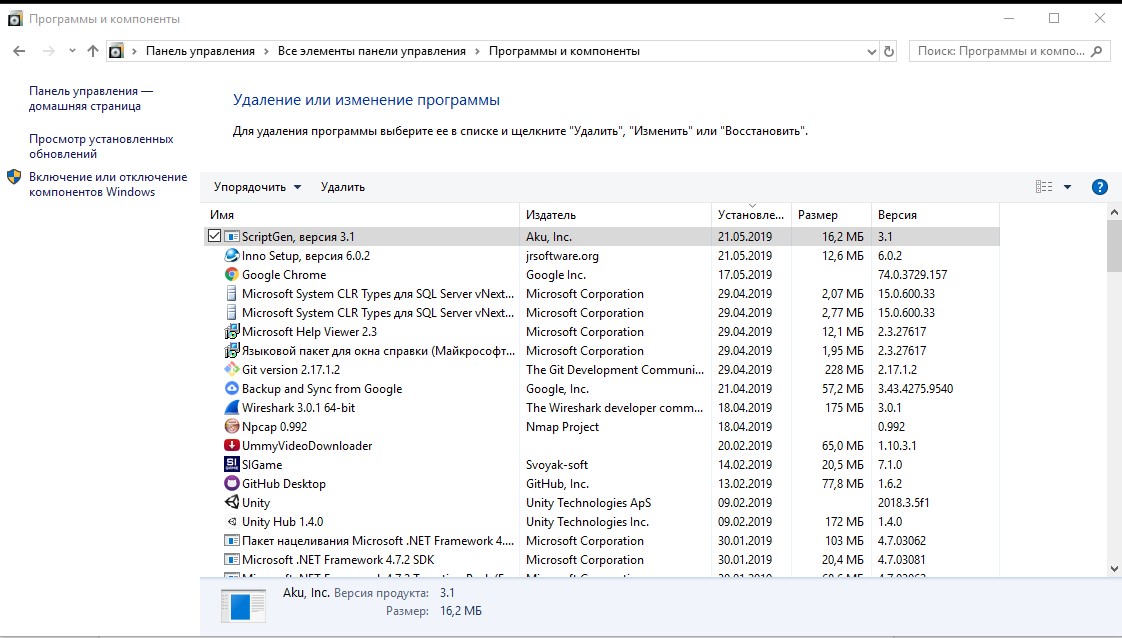


Рисунок 14 – Начало деинсталяции

1. Подтвердить удаление, нажав «Да» в открывшемся диалоге.

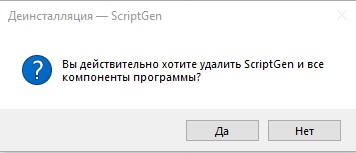


Рисунок 15 – Подтверждение деинсталляции

1. Дождаться полного удаления программного обеспечения. Нажать «Завершить».

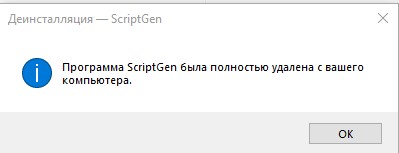


Рисунок 16 – Завершение деинсталляции

СООБЩЕНИЯ

В приложении присутствует система уведомлений, с тремя типами сообщений:

1. Сообщения об ошибке;

Происходят при появлении ошибок в модели, при ошибках в плагинах или непредвиденных исключениях программы.

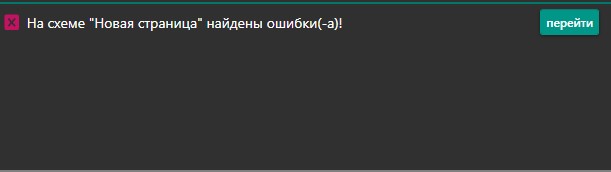


Рисунок 17 – Сообщения об ошибке

1. Предупреждения;

Происходят при некритических ошибках, например, при невозможности создания новых уникальных значений в генераторе.



Рисунок 18 - Предупреждения

1. Информационные сообщения.

Происходят при успешном завершении операций, например, при отсутствии ошибок в модели или успешном сохранении скриптов.

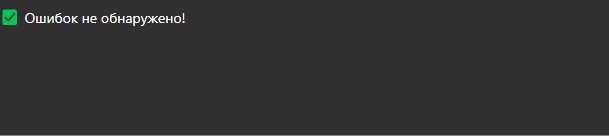


Рисунок 19 – Информационные сообщения

.

ПРИЛОЖЕНИЕ Д. Результаты проверки на уникальность

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Курсовой проект

ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

МДК 01.02. «Прикладное программирование»

Специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения проектирования, документирования и генерации запросов баз данных»

|  |  |
| --- | --- |
|  | Руководитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Шимбирёв |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |
|  | Исполнитель |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.С. Абрамкин |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 год |
|  |  |

2019

АННОТАЦИЯ

В приложении «Результаты проверки на уникальность» предоставлен результат проверки на уникальность пояснительной записки для курсового проекта «Разработка приложения проектирования, документирования и генерации запросов баз данных».

РЕЗУЛЬТАТЫ ПРОВЕРКИ НА УНИКАЛЬНОСТЬ

На рисунке 1 показан результат проверки на уникальность пояснительной записке, полученный с помощью интернет-проекта, программно-аппаратного комплекса для проверки текстовых документов на наличие заимствований из открытых источников в сети Интернет и других источников Антиплагиат (Антиплагиат.ру).

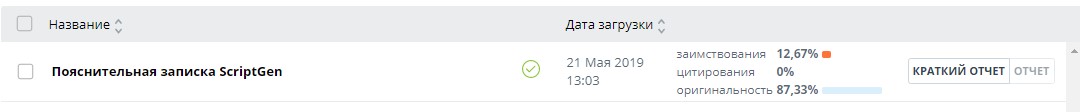


Рисунок 1 – Результат проверки на уникальность