МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Приложение пользователя для системы автозаполнения баз данных

Студент _____ Бинус Полина Игоревна
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__»_____2020 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент_____Бинус Полина Игоревна_____гр._16ВИ1__факультета__ВТ____
<Фамилия, имя, отчество полностью>

2. Тема работы___Приложение пользователя для системы автозаполнения баз данных_____

Тема утверждена приказом ПГУ №_____от "_____"_____20_____

2. Руководитель работы_____к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С.В.

3. Задание на работу (назначение разработки, исходные данные и т.п.)_____

Назначение: *приложение пользователя для системы автозаполнения баз данных предназначено для организации общения между пользователем и программными средствами взаимодействия с системами управления баз данных, для создания правил генерации данных и контроля процесса автозаполнения баз данных*

Средства выполнения: Операционная система Windows (версия 10 pro)
Средства разработки: *Visual Studio 2019, Framework 4.8, .NET*
Язык программирования: C#
Технологии разработки: ООП, UML, MVC

4. Перечень подлежащих разработке вопросов

Анализ предметной области и постановка задачи на разработку
Планирование разработки и расчет бюджета
Разработка требований к приложению пользователя для системы автозаполнения баз данных
Разработка приложения пользователя для системы автозаполнения баз данных
Представление руководства пользователя для системы автозаполнения баз данных
Тестирование приложения пользователя для системы автозаполнения баз данных

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи на разработку	10	13.04.2020 – 19.04.2020	
2	Разработка требований к приложению пользователя для системы автозаполнения баз данных	20	20.04.2020 – 01.05.2020	
3	Разработка приложения пользователя для системы автозаполнения баз данных	45	02.05.2020 – 29.05.2020	
4	Тестирование приложения пользователя для системы автозаполнения баз данных	15	30.06.2020 – 08.06.2020	
5	Оформление пояснительной записки	10	9.06.2020 – 13.06.2020	

Дата выдачи задания " 30 "__декабря_____2019__

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 100 листов, 11 таблиц, 44 рисунка, 28 использованных источников и 2 приложения.

АВТОЗАПОЛНЕНИЕ, ГЕНЕРАЦИЯ, БД, СУБД, СПРАВОЧНИКИ, КЛАССИФИКАТОРЫ, SQL, MVC, UML, C#.

Объект исследования: приложение пользователя для системы автозаполнения баз данных.

Цель работы: реализовать приложения пользователя для прототипа системы автозаполнения баз данных.

Технологии разработки: ООП, MVC, C#, Visual Studio, .Net Framework.

Задание: спроектировать и реализовать интерфейс пользовательского приложения, логику подключения к программными средствам взаимодействия с системами управления базами данных и бизнес-логику приложения.

Результаты работы: спроектировано и разработано приложение пользователя для прототипа системы автозаполнения баз данных.

СОДЕРЖАНИЕ

Введение 8

1 Анализ предметной области и постановка задачи на разработку 10

1.1 Автоматизированное заполнение баз данных информационных систем 10

1.1.1 Автозаполнение баз данных в жизненном цикле информационных систем 10

1.1.2 Современное состояние исследований проблемы автозаполнения баз данных 11

1.1.3 Актуальность разработки системы автозаполнения баз данных 12

1.2 Сравнительный анализ современных средств автоматизированного заполнения баз данных 13

1.2.1 Цели и задачи сравнительного анализа 13

1.2.2 Выбор средств автоматизированной генерации данных 13

1.2.3 Обзор средств для автоматизированной генерации данных 15

1.2.4 Результаты сравнительного анализа 23

1.3 Постановка задачи на разработку 24

1.3.1 Постановка задачи на разработку системы автозаполнения баз данных 24

1.3.1.1 Цели разработки и назначение системы автозаполнения баз данных 24

1.3.1.2 Пользователи системы автозаполнения баз данных 25

1.3.1.3 Автоматизируемые функции системы автозаполнения баз данных .. 25

1.3.1.4 Аппаратное и программное окружение системы автозаполнения баз данных 26

1.3.2 Постановка задачи на разработку приложения пользователя для системы автозаполнения баз данных 26

1.3.2.1 Цели разработки и назначение приложения пользователя для системы автозаполнения баз данных 26

1.3.2.2 Основные функции приложения пользователя для системы автозаполнения баз данных 27

1.3.2.3 Требования к пользовательскому интерфейсу 27

1.4 Планирование разработки и расчет бюджета 28

2 Разработка требований к приложению пользователя для системы автозаполнения баз данных 32

2.1 Функциональные требования 32

2.2 Требования к программной архитектуре 37

2.2.1 Требования к обобщенной архитектуре системы 37

2.2.2 Требования к архитектуре приложения пользователя для системы автозаполнения баз данных 39

2.3 Требования к аппаратно-программному окружению приложения пользователя для системы автозаполнения баз данных 41

2.4 Обоснование выбора поддерживаемых СУБД 42

2.5 Обоснование выбора средств разработки приложения пользователя для системы автозаполнения баз данных 44

2.5.1 Обоснование выбора языковых средств разработки 44

2.5.2 Обоснование выбора инструментальных средств разработки 45

3 Разработка приложения пользователя для системы автозаполнения баз данных 47

3.1 Разработка интерфейса пользователя 47

3.2 Разработка структуры приложения 50

3.3 Спецификация методов приложения пользователя 51

4 Руководство пользователя 55
4.1 Описание главного меню 55
4.2 Работа с вкладками и дополнительными окнами 58
4.3 Справочная информация 65
5 Тестирование приложения пользователя для системы автозаполнения баз данных 67
5.1 Выбор методологии тестирования 67
5.2 Проведение тестирования приложения 68
5.3 Метрики Хостледа 72
Заключение 75
Список использованных источников 76
Приложение А. Глоссарий 79
Приложение Б. Листинг программы 82

Введение

В современном мире появляется все больше информационных систем, и в процессе их разработки возникает необходимость начального заполнения баз данных. Это требуется в следующих случаях:

– при переносе данных из предыдущей системы баз данных (БД);

– при тестировании нагрузки системы, когда требуется большое количество записей;

– при загрузке данных из справочников и классификаторов.

Актуальность выбранной темы связана с тем, что вручную внесение больших объемов тестовых данных в базы данных затруднительно. Решить задачу повышения качества и сокращения расходов на нагрузочное тестирование можно за счет привлечения эффективных средств автоматизации разработки тестовых данных.

Главной целью разработки является создание системы, которая будет автоматизировать процесс наполнения БД тестовыми данными. При этом должна учитываться поддержка различных СУБД реляционного типа.

Для достижения поставленной цели необходимо решить следующие задачи:

– выполнить анализ предметной области;

– провести сравнительный анализ существующих систем-аналогов;

– на основе анализа выполнить постановку задачи на разработку системы автозаполнения баз данных в целом и приложения пользователя, в частности;

– определить функционал приложения пользователя для системы автозаполнения баз данных с использованием языка UML;

– разработать приложение пользователя для системы автозаполнения баз данных в соответствии с поставленной задачей;

– провести тестирование ПО.

Существующие системы-аналоги были изучены в статье «Сравнительный анализ средств автоматического заполнения баз данных»,

представленной в 2019г. на XVI Международной научно-технической конференции [1].

Основные требования к реализуемой системе и ее функционал были выделены в статье «Функциональные возможности системы автоматического заполнения баз данных», подготовленной для VII ежегодной всероссийской межвузовской научно-практической конференции – 2020 [2].

В результате выполнения работы была реализована система автозаполнения баз данных, которая позволяет:

– автоматизировать заполнение базы данных;

– автоматически генерировать случайные данные;

– загружать данные из справочников и классификаторов;

– перемещать данные между различными базами данных, в том числе под управлением разных СУБД;

– выполнять резервное копирование и восстановление данных.

1 Анализ предметной области и постановка задачи на разработку

1.1 Автоматизированное заполнение баз данных информационных систем

1.1.1 Автозаполнение баз данных в жизненном цикле информационных систем

Базы данных – это важнейший компонент ИС, поэтому жизненные циклы (ЖЦ) базы данных и информационной системы взаимосвязаны. ЖЦ ИС – это комплекс этапов развития ИС от принятия решения ее реализации до прекращения эксплуатации.

ЖЦ БД включает в себя следующие этапы:

– планирование разработки БД;
– определение требований к системе;
– сбор и анализ требований пользователей;
– этап проектирования (моделирования) БД;
– выбор целевой СУБД;
– разработка приложений;
– создание БД;
– конвертирование и загрузка данных из старой системы;
– тестирование БД;
– эксплуатация и сопровождение [3].

Средства автоматического заполнения баз данных применяются:

– при начальной загрузке международных, общероссийских, отраслевых справочников и классификаторов на этапе создания БД, эти данные необходимы для реального функционирования ИС;

– на этапе конвертирования и загрузки данных из старой системы, когда из предыдущих версий ИС или резервных копий выбираются данные, которые также необходимы для реального функционирования ИС;

– на этапе тестирования БД, когда осуществляется загрузка тестовых данных для оценки корректности и эффективности запросов к БД, функционального и комплексного тестирования приложений ИС.

1.1.2 Современное состояние исследований проблемы автозаполнения баз данных

Чтобы понять, насколько актуальной является разработка системы автозаполнения БД, необходимо сначала определить современное состояние исследований в данной области.

В статье А.В. Демакова «Генерация тестовых данных сложной структуры с учетом контекстных ограничений» освещается проблема оптимизации процесса генерации тестовых данных. Автор предлагает технологию тонкой настройки генерации данных, которая приводит к уменьшению объемов генерируемых данных и повышению их качества. Это достигается применением техники абстрактных моделей. Ограничения определяются с помощью итераторов – объектов, имеющих несколько последовательных состояний. В целом, генерация данных сложной структуры сводится к генерации атрибутированных деревьев с наличием горизонтальных связей, что позволяет учитывать контекст элементов дерева. Ключевым моментом в схеме генерации является то, что для каждого типа вершин дерева имеется свой генератор, который появляется по требованию и зависит от контекста [4].

Е.А. Костычев в своей публикации «Нацеленная генерация данных для тестирования приложений над базами данных» сообщает о недостатках существующих инструментов. В случае генерации на основе реализации, отсутствует гарантия покрытия всех требуемых функциональных ветвей. Во многих современных инструментах такая гарантия возможна только при полном переборе комбинаций значений полей. Это приводит к временным затратам и при генерации, и при анализе полученных тестов. В качестве решения этих проблем, автор предлагает подход на основе формализации и

анализа требований к тестируемой системе, который поддерживается генератором данных сложной структуры Pinery [5].

Подробнее о генераторе Pinery говорится в работе А.В. Демакова «Генератор сложных данных Pinery: реализация новых возможностей UniTESK». Данный инструмент предназначен для практического использования подхода к автоматической генерации тестовых данных сложной структуры, основанного на формальном описании тестовых данных в виде грамматик. В статье предложен метод разработки подобных генераторов, который подразумевает генерацию небольших объемов данных для тестирования заданных функций программного обеспечения (ПО), а также позволяет повторно использовать отдельные модули за счет обеспечения модульности настройки генератора [6].

Изучение современного состояния исследований по выбранной теме показало, что в настоящее время предпочтение отдается генерации небольшого количества тестовых данных. Такой подход должен обеспечивать экономию временных затрат на процесс тестирования и повышение его качества. В большинстве существующих инструментов отсутствует подобная оптимизация.

1.1.3 Актуальность разработки системы автозаполнения баз данных

Актуальность разработки системы автоматизированного заполнения баз данных обусловлена тем, что аналогичных инструментов существует не так много, и в большинстве из них отсутствует возможность управления генерацией данных, в том числе данных сложной структуры. В рамках разрабатываемой системы под данными сложной структуры понимаются логически непротиворечивые данные.

Система автоматизированного заполнения баз данных актуальна при поддержке этапов реализации, начальной загрузки данных и тестирования в ЖЦ приложений БД.

На этапе реализации происходит загрузка международных, общероссийских, отраслевых справочников и классификаторов.

На этапе начальной загрузки данных в новую систему переносятся данные из предыдущих систем или резервных копий.

На этапе тестирования загружаются реальные тестовые, которые не сохраняются в БД после проведения проверок.

Внедрение разработанной системы позволит более тонко настраивать процесс генерации, облегчить миграцию данных из предыдущих версий БД, что обеспечит экономию времени и повысит качество тестирования. В дальнейшем к системе будет возможно подключать необходимые СУБД и тем самым расширять ее применение.

1.2 Сравнительный анализ современных средств автоматизированного заполнения баз данных

1.2.1 Цели и задачи сравнительного анализа

Сравнительный анализ современных средств автоматизированной генерации данных позволит определить наиболее используемые системы, а также получить представление о том, насколько они распространены и актуальны.

Можно выделить следующие цели проведения анализа систем:

– выявление достоинств;

– выявление недостатков;

– выявление характерных базовых функций.

По итогам анализа главной задачей будет определение обязательных функций, а также уникальных возможностей, которые расширят функционал реализуемой системы и приведут к ее высокой конкурентоспособности.

1.2.2 Выбор средств автоматизированной генерации данных

На сегодняшний день существует множество инструментальных средств, которые позволяют выполнять полноценное заполнение БД. Далее приведены лишь некоторые из них:

- Databene Benerator;
- DATATECT;
- DTM DATA GENERATOR;
- EMS DATA GENERATOR;
- RUBY-PLSQL-SPEC;
- SELITE;
- DBeaver;
- Red Gate SQL Data Generator;
- HTDGen;
- DBFORGE STUDIO;
- DBMonster;
- Turbo Data;
- AGENDA.

Как правило, эти средства ориентированы на генерацию тестовых данных. Для проведения сравнительного анализа были выбраны следующие системы:

- Databene Benerator;
- DTM DATA GENERATOR;
- EMS DATA GENERATOR;
- DBeaver;
- Red Gate SQL Data Generator;
- DBFORGE STUDIO.

Среди представленных средств встречаются как бесплатные, так и платные, у которых возможно наличие on-line версий. Эта информация подробно представлена в таблице 1

Таблица 1 – Информация о системах

Система	Доступность	Наличие онлайн-версии	Фирма-производитель
Databene Benerator	Бесплатная	+	Volker Bergmann
DTM DATA GENERATOR	Платная, есть бесплатные версии	+	DTM soft

Продолжение таблицы 1

Система	Доступность	Наличие онлайн-версии	Фирма-производитель
EMS DATA GENERATOR	Платная	+	EMS
DBeaver	Бесплатная	-	Open-source community
Red Gate SQL Data Generator	Платная. Демо-версия на две недели	+	Red Gate Software Ltd
DBFORGE STUDIO	Платная. Демо-версия на 30 дней, присутствует версия с неполным функционалом без временного ограничения	+	Devart

Таким образом, для дальнейшего анализа были выбраны наиболее используемые инструменты автоматической генерации данных.

1.2.3 Обзор средств для автоматизированной генерации данных

Databene Benerator – это платформа для создания реалистичных тестовых данных большого объема, используемых для тестирования нагрузки и производительности, а также для настройки витрины. Данные генерируются из легко конфигурируемой модели метаданных и экспортируются в базы данных, XML, CSV или плоские файлы, не имеющие внутреннюю иерархию [7]. На рисунке 1 показан интерфейс этого приложения.

EMS Data Generator – это мощная программа для генерации тестовых данных для таблиц баз данных с возможностью сохранения и редактирования сценариев. С помощью этого приложения возможно моделировать среду рабочей базы данных, заполнять одновременно несколько таблиц тестовыми данными, задавать таблицы для генерации данных, устанавливать диапазоны значений, создавать текстовые поля по маске, задавать списки значений вручную или выбирать их из запросов SQL, настраивать различные

параметры генерации тестовых данных для каждого типа поля и многое другое.

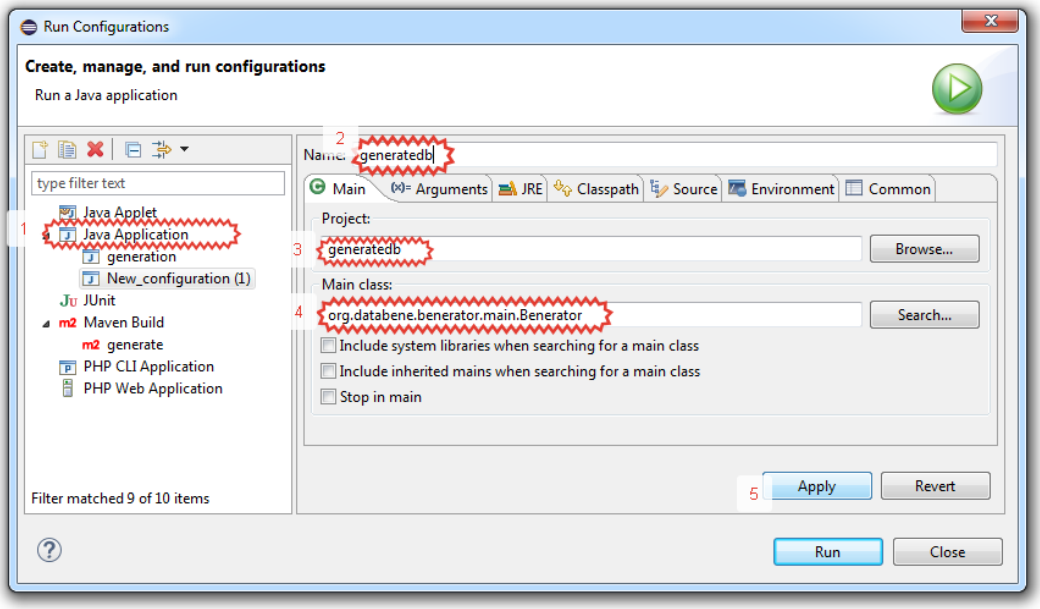


Рисунок 1 – Интерфейс Databene Benerator

Эта программа предоставляет все необходимые инструменты для быстрой генерации тестовых данных и дополнительно включает консольную утилиту для генерации тестовых данных с помощью шаблонов генерации [8]. EMS Data Generator работает с несколькими СУБД, на рисунке 2 представлена первая странница интерфейса приложения для SQL Server, организованного в виде мастера.

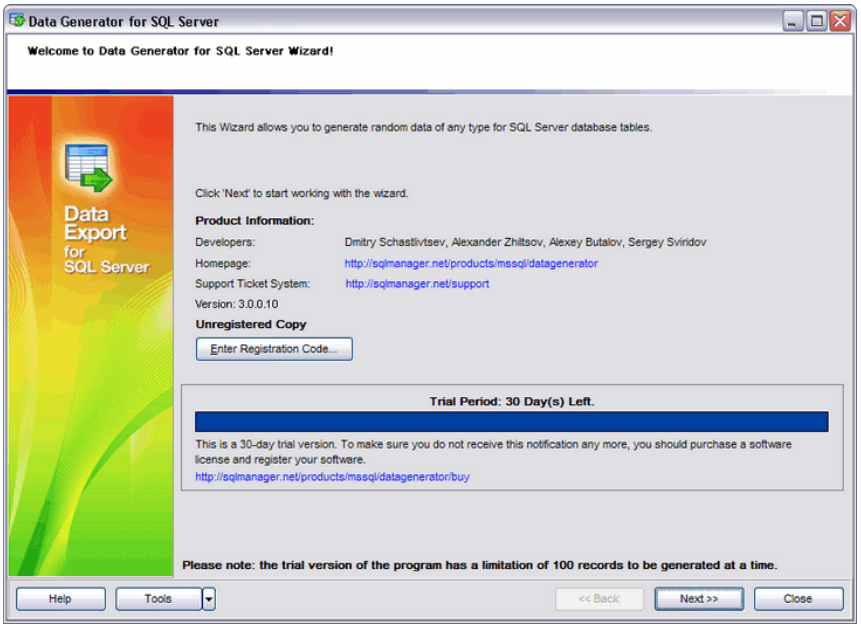
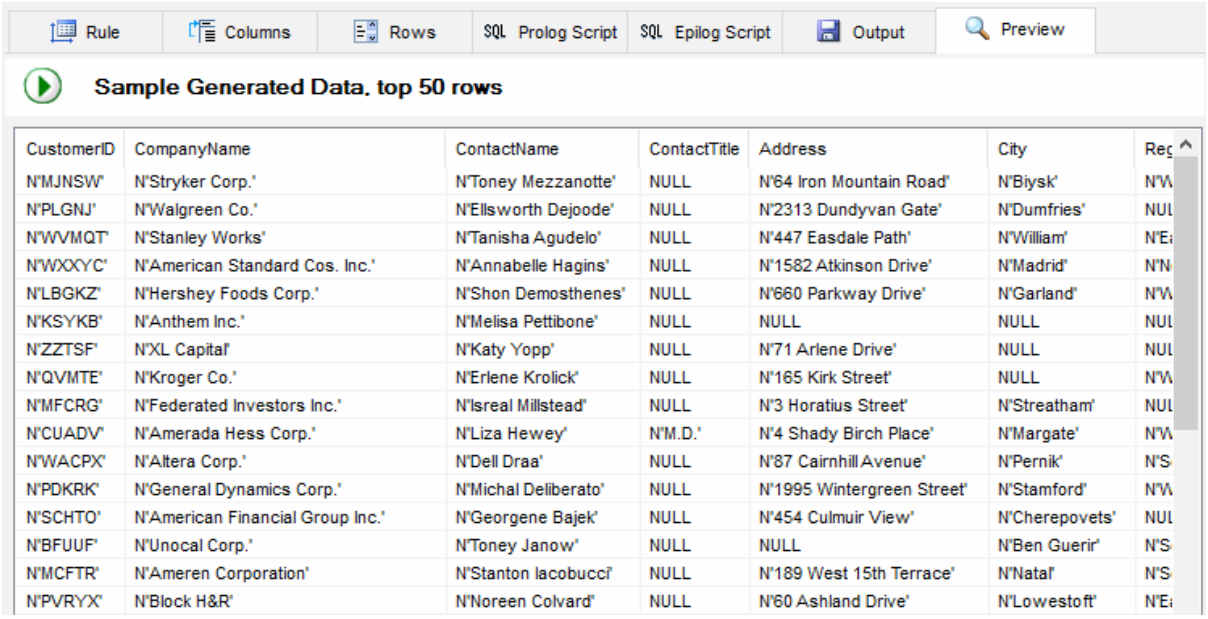


Рисунок 2 – Интерфейс EMS Data Generator for SQL Server

DTM Data Generator – это программный продукт, который создает строки данных и объекты схемы для целей тестирования: тестирование базы данных, анализ производительности, тестирование QA или выполнение тестов загрузки. Генератор был разработан для обеспечения разработчиков и инженеров по обеспечению качества высококачественными и реалистичными тестовыми массивами. Он автоматически создает значения данных и необязательные объекты схемы (таблицы, представления, процедуры, триггеры) [9]. На рисунке 3 представлен интерфейс системы.



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Reg
N'MJNSW'	N'Stryker Corp.'	N'Toney Mezzanotte'	NULL	N'64 Iron Mountain Road'	N'Biysk'	N'W
N'PLGNJ'	N'Walgreen Co.'	N'Ellsworth Dejoode'	NULL	N'2313 Dundyvan Gate'	N'Dumfries'	NUL
N'WVMQT'	N'Stanley Works'	N'Tanisha Agudelo'	NULL	N'447 Easdale Path'	N'William'	N'E
N'WXXYC'	N'American Standard Cos. Inc.'	N'Annabelle Hagins'	NULL	N'1582 Atkinson Drive'	N'Madrid'	N'N
N'LBGKZ'	N'Hershey Foods Corp.'	N'Shon Demosthenes'	NULL	N'660 Parkway Drive'	N'Garland'	N'W
N'KSYKB'	N'Anthem Inc.'	N'Melisa Pettibone'	NULL	NULL	NULL	NUL
N'ZZTSF'	N'XL Capital'	N'Katy Yopp'	NULL	N'71 Arlene Drive'	NULL	NUL
N'QVMTE'	N'Kroger Co.'	N'Erlene Krolick'	NULL	N'165 Kirk Street'	NULL	N'W
N'MFCRG'	N'Federated Investors Inc.'	N'Isreal Millstead'	NULL	N'3 Horatius Street'	N'Streatham'	NUL
N'CUADV'	N'Amerada Hess Corp.'	N'Liza Hewey'	N'M.D.'	N'4 Shady Birch Place'	N'Margate'	N'W
N'WACPX'	N'Altera Corp.'	N'Dell Draa'	NULL	N'87 Cairnhill Avenue'	N'Pernik'	N'S
N'PDKRK'	N'General Dynamics Corp.'	N'Michal Deliberato'	NULL	N'1995 Wintergreen Street'	N'Stamford'	N'W
N'SCHTO'	N'American Financial Group Inc.'	N'Georgene Bajek'	NULL	N'454 Culmuir View'	N'Cherepovets'	NUL
N'BFUUF'	N'Unocal Corp.'	N'Toney Janow'	NULL	NULL	N'Ben Guerir'	N'S
N'MCFTR'	N'Ameren Corporation'	N'Stanton Iacobucci'	NULL	N'189 West 15th Terrace'	N'Natal'	N'S
N'PVRYX'	N'Block H&R'	N'Noreen Colvard'	NULL	N'60 Ashland Drive'	N'Lowestoft'	N'E

Рисунок 3 – Интерфейс DTM Data Generator

DBeaver – бесплатный мультиплатформенный инструмент для работы с БД для разработчиков, администраторов баз данных, аналитиков и всех людей, которым необходимо работать с базами данных. Поддерживает все популярные базы данных: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata и другие. DBeaver написан на фреймворке с открытым исходным кодом, что позволяет дописывать различные расширения (плагины) [10]. Основной интерфейс системы приложения представлен на рисунке 4.

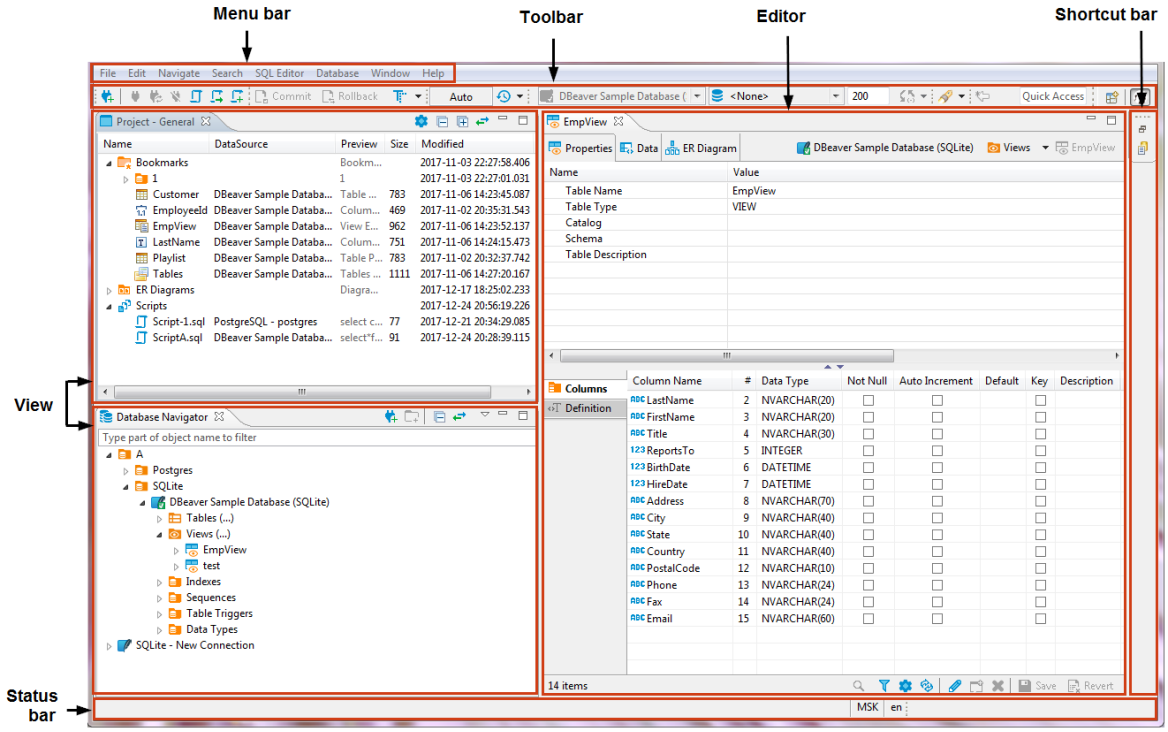


Рисунок 4 – Интерфейс DBeaver

Red Gate SQL Data Generator – мощная программа, позволяющая генерировать тестовые данные для таблиц баз данных SQL Server. Использование приложения дает возможность моделировать среду рабочей базы данных, заполнять одновременно несколько таблиц SQL Server тестовыми данными, задавать формат таблиц и настраивать параметры генерации данных SQL Server для каждого типа поля. Red Gate SQL Data Generator предоставляет необходимые инструменты для быстрой и качественной генерации тестовых данных, настройки диапазонов и списков значений. Программа предусматривает автоматическое заполнение таблиц и возможность редактирования накопленных баз данных, позволяет не перегружать базу данных дополнительной информацией [11]. Интерфейс данного средства представлен на рисунке 5.

DBFORGE STUDIO – это полнофункциональная среда разработки для управления, администрирования, представления и анализа данных. Этот инструмент будет полезен для разработчиков и администраторов баз данных при выполнении сложных задач, таких как проектирование баз данных,

написание SQL кода, сравнение баз данных, синхронизация схем и данных, генерации реальных тестовых данных и т.п.

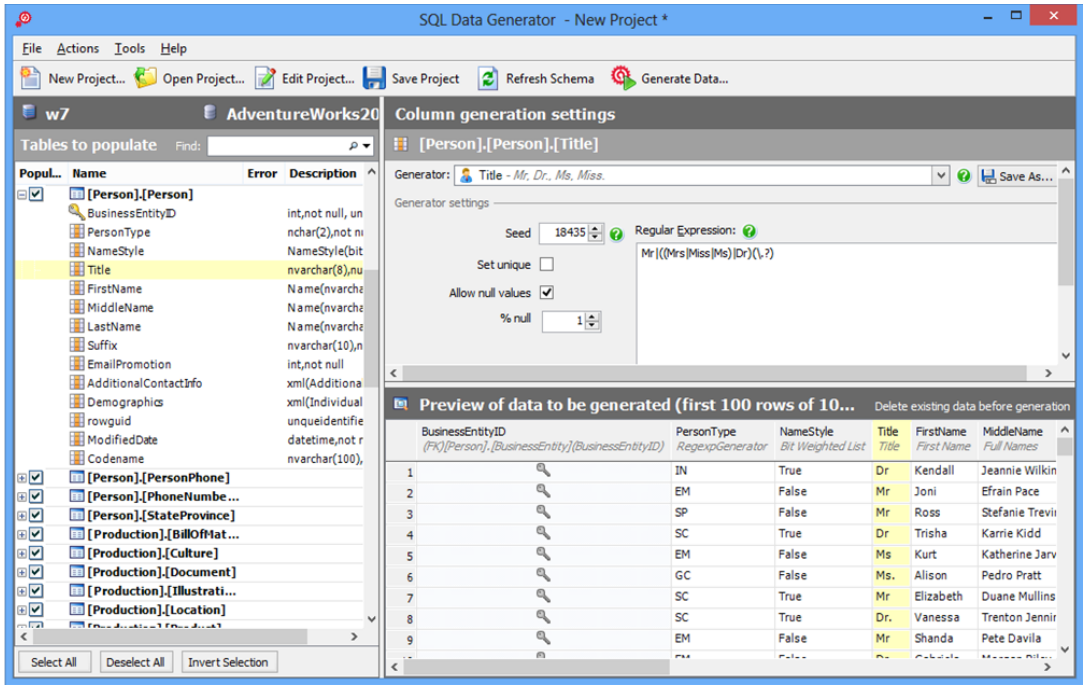


Рисунок 5 – Интерфейс Red Gate SQL Data Generator

У производителя DBFORGE STUDIO есть несколько отдельных приложений для разный систем управления базами данных [12]. На рисунке 6 представлен интерфейс генерации данных в DBFORGE STUDIO для SQL Server.

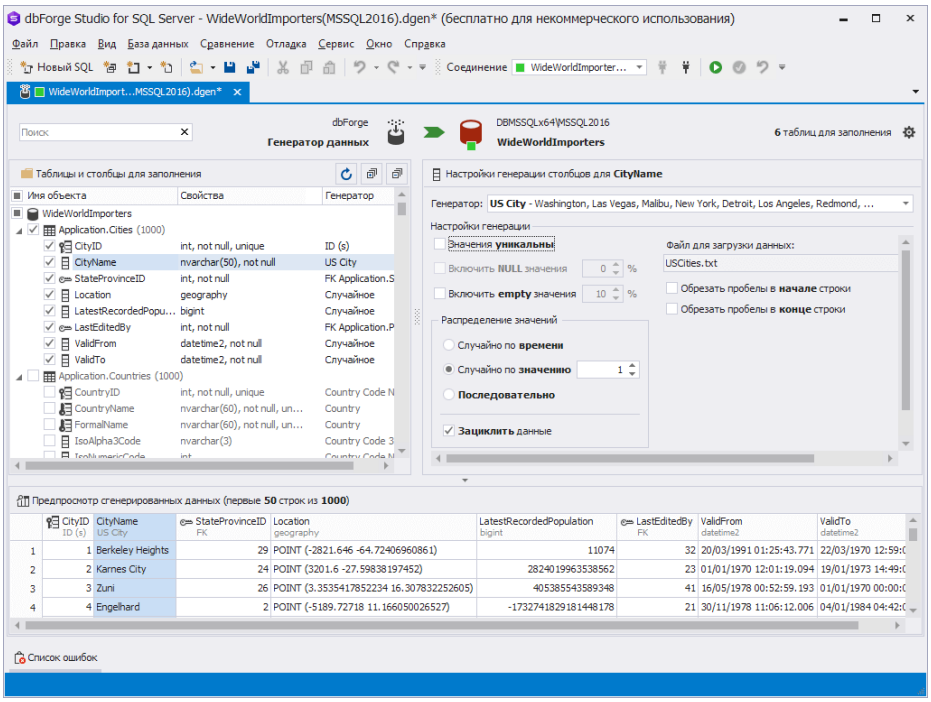


Рисунок 5 – Интерфейс DBFORGE STUDIO for SQL Server

Для оценки, анализа и сравнения инструментальных средств были выделены следующие критерии:

– загрузка из справочников – наличие функции загрузки тестовых данных из встроенных библиотек или подключение дополнительных справочников;

– импорт – возможность загрузки данных из сторонних БД;

– экспорт – возможность выгрузки в сторонние БД;

– поддерживаемые БД – перечень БД, взаимодействие с которыми поддерживает система;

– платформа – операционные системы, которые поддерживают рассматриваемые инструментальные средства;

– форматы файлов – допустимые форматы файлов;

– анализ структуры БД – поддержка связи между таблицами и ссылочной целостности;

– предварительный просмотр – наличие функции вывода промежуточных результатов.

Результаты анализа выделенных инструментальных средств автоматической генерации данных представлены в таблице 2.

Таблица 2 – Результирующая таблица анализа систем по критериям

Функции	Системы					
	Databene Benerator	DTM DATA GENERATOR	EMS DATA GENERATOR	DBeaver	Red Gate SQL Data Generator	DBFORGE STUDIO
Загрузка из справочников	+	+	-	+	+	+
Импорт	+	-	-	+	+	+
Экспорт	+	-	+	+	-	+
Поддерживаемые СУБД	Oracle, DB2, MS SQL Server, MySQL, PostgreSQL, HSQL, H2, Derby, Firebird	MS SQL Server, Oracle, DB2, MySQL, PostgreSQL, Informix, Sybase, SQLite and Interbase/Firebird	MySQL, Oracle, Db2, Microsoft SQL Server, PostgreSQL, Interbase/ Firebird	MySQL, Oracle, PostgreSQL, IBM DB2, Microsoft SQL Server, Sybase, ODBC, Java DB (Derby), Firebird (Interbase), HSQLDB, SQLite, Mimer, IBM Informix, SAP MAX DB, Cache, Ingres, Linter, Любой JDBC совместимый источник	Microsoft SQL Server	MySQL, Oracle, PostgreSQL, Microsoft SQL Server

Продолжение таблицы 2

Платформа	Linux, Windows и Mac OS	Имеет 32- и 64-разрядные версии и поддерживает все современные версии Windows : XP, Vista, Windows 7, Windows 8/10 (для настольных ПК), Windows Server 2003 и новее. Unix / Linux, Mac OS	Многоплатформенность	Linux, Windows и MacOS, Solaris, AIX, HPUX	Windows	Linux, Windows и MacOS
Форматы файлов	Сценарий SQL, текст / CSV, XML, JSON .	CSV / текстовые файлы, электронные таблицы Excel, документы XML, файлы JSON, файлы Access, веб-ресурсы, пользовательские сценарии и т. д.	Текстовые файлы, электронные таблицы Excel, документы XML, веб-ресурсы, пользовательские сценарии, CSV и т. д.	Текстовые файлы, электронные таблицы Excel, документы XML, файлы JSON, веб-ресурсы, пользовательские сценарии и т. д.	CSV, электронные таблицы Excel	Текстовые файлы, электронные таблицы Excel, документы XML, файлы JSON, веб-ресурсы, пользовательские сценарии и т. д.
Анализ структуры БД	+	+	+	+	+	+
Предварительный просмотр	-	+	+	-	-	-

На основе результирующей таблицы наглядно определены сильные и слабые стороны систем автоматической генерации данных.

1.2.4 Результаты сравнительного анализа

В результате анализа было выявлено, что функция загрузки из справочников присутствует во всех системах, кроме EMS DATA GENERATOR. Импорт из сторонних БД обеспечивают Databene Benerator, DBeaver, Red Gate SQL Data Generator и DBFORGE STUDIO. Системы DTM DATA GENERATOR и Red Gate SQL Data Generator не предоставляют возможности экспорта в сторонние БД. Все рассмотренные средства поддерживают достаточное количество различных БД, и только Red Gate SQL Data Generator ориентирована исключительно на Microsoft SQL Server. Каждая из выделенных систем совместима со всеми основными ОС, за исключением Red Gate SQL Data Generator, которая работает только на Windows. Доступна работа с наиболее популярными форматами файлов. Функция анализа структуры БД осуществляется во всех представленных средствах. Системы DTM DATA GENERATOR и EMS DATA GENERATOR имеют дополнительную опцию предварительного просмотра промежуточных результатов для удобства работы.

Таким образом, среди платных систем наиболее полным функционалом обладает DBFORGE STUDIO от Devart, но использование свободно распространяемых систем обеспечит существенную экономию.

Среди бесплатных систем Databene Benerator от Volker Bergmann и DBeaver от Open-source community сложно отдать приоритет какой-либо. Обе имеют одинаковый рассматриваемый функционал, за исключением списка поддерживаемых баз данных, который больше у DBeaver. Дальнейший выбор инструментальных средств генерации зависит от удобства интерфейса и предпочтений конкретного пользователя [1].

В большинстве рассмотренных средствах автоматической генерации данных присутствуют стандартные функции, которые необходимо поддержать и в реализуемой системе. К ним относятся:

– загрузка из справочников;
– поддержка различных СУБД;
– работа с множеством форматов файлов;
– анализ структуры БД.

Кроме обязательных функций, в разрабатываемую систему необходимо включить ряд дополнительных возможностей, обеспечивающих ее уникальность и конкурентоспособность. В качестве таких особенностей были определены следующие функции:

– расширение числа поддерживаемых СУБД;
– задание и модификация схемы БД в процессе наполнения;
– подключение новых справочников и классификаторов в процессе функционирования системы;
– резервное копирование и восстановление данных.

1.3 Постановка задачи на разработку

1.3.1 Постановка задачи на разработку системы автозаполнения баз данных

1.3.1.1 Цели разработки и назначение системы автозаполнения баз данных

Целью разработки является создание универсальной системы для автоматической генерации больших объемов данных. Система должна обеспечивать быстрое подключение к нескольким СУБД для импорта и экспорта данных, легкой генерации случайных данных, а также генерации данных на основе справочников и классификаторов. Средство автоматической генерации данных будет реализовано на языке C# на базе MS Visusal Studio. Доступ к различным СУБД и поддержка форматов файлов будут реализованы через интерфейсы plug-in, взаимодействующих на разработанных библиотеках DLL. Система направлена на оптимизацию процесса тестирования баз данных.

1.3.1.2 Пользователи системы автозаполнения баз данных

Пользователей ИС можно разделить на две категории: внутренние и конечные. К внутренним относятся пользователи, которые разрабатывают ИС и поддерживают ее функционирование. Конечные пользователи используют ИС непосредственно по назначению.

В рамках реализуемой системы можно говорить об отсутствии четкого разграничения, так как она ориентирована на пользователей, деятельность которых связана с разработкой и поддержанием ИС. К ним относятся:

– администратор системы – это специалист, который руководит всеми работами по созданию БД и программного окружения, принимает потребности пользователей, отвечает за эксплуатацию системы и обеспечивает ее работоспособность;

– прикладной программист – это разработчик программ для решения прикладных задач, который реализует запросы к базе данных;

– системный программист – специалист, который осуществляет контроль за функционированием СУБД в процессе работы, а также разрабатывает программы, расширяющие возможности СУБД.

– тестировщик БД – специалист, задачей которого является выявление сбоев и ошибок в работе БД.

1.3.1.3 Автоматизируемые функции системы автозаполнения баз данных

Система автоматической генерации данных автоматизирует следующие функции:

– генерация больших объемов случайных данных;

– генерация данных по заранее заданным правилам;

– наполнение БД сгенерированными данными;

– перемещение данных между разными БД;

– перенос в БД информации из справочников и классификаторов;

– создание и изменение схемы базы данных.

С помощью интерфейса система задаёт правила генерации данных, соответствие этих правил атрибутам таблицы БД и контролирует процесс заполнения БД в процессе генерации.

1.3.1.4 Аппаратное и программное окружение системы автозаполнения баз данных

Для работы системы автоматической генерации данных на ПК пользователя необходимы следующие минимальные требования к программно-аппаратным средствам:

a) операционная система: Microsoft Windows 7, 8.1, 10;

b) наличие необходимой СУБД: Microsoft SQL Server (2014, 2016, 2017), PostgreSQL (9.0.1 или выше), MySQL(5.7.11 или выше), Oracle(10g XE или выше);

c) параметры компьютера:

- процессор 32-разрядный или 64-разрядный с тактовой частотой 1 ГГц или выше;
- оперативная память не менее 2 ГБ;
- свободное место не менее 10 ГБ на жестком диске;
- графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

Необходимым параметром является возможность подключения к серверу с помощью внешней сети.

1.3.2 Постановка задачи на разработку приложения пользователя для системы автозаполнения баз данных

1.3.2.1 Цели разработки и назначение приложения пользователя для системы автозаполнения баз данных

Пользовательское приложение разрабатывается с целью удобного, понятного и простого общения пользователя с программными средствами взаимодействия СУБД для автозаполнения БД. Приложение пользователя предназначено для создания правил генерации данных, указания

соответствий этих правил атрибутам таблиц БД и контроля процесса автозаполнения БД.

Приложение должно иметь привлекательный, удобный и дружелюбный (интуитивно понятный) интерфейс и быть удобным для установки и модернизации.

Приложение должно иметь документацию: руководство пользователя, руководство администратора.

1.3.2.2 Основные функции приложения пользователя для системы автозаполнения баз данных

Приложение пользователя для системы автозаполнения баз данных включает в себя следующие функции:

- работа с проектом;
- выбор присоединяемых серверов БД;
- выбор присоединяемой БД;
- работа с таблицами БД;
- работа с полями таблиц БД;
- настройка правил генерации данных;
- подключение справочников и классификаторов;
- управление процессом автозаполнения.

Также одной из основных функций приложения пользователя является предоставление справочной информации о системе.

1.3.2.3 Требования к пользовательскому интерфейсу

Пользовательский интерфейс обеспечивает взаимодействие пользователя и системы автозаполнения БД. Пользовательский интерфейс обеспечивает безопасность взаимодействия. Безопасность определяется тем, что каждый пользователь имеет безопасный доступ к системе путём регистрации с указанием логина и безопасного пароля, который задаётся не менее восьмью символами буквенно-цифровыми. Безопасность доступа к

данным обеспечивается указанием пароля администратора при подключении к БД.

Для взаимодействия с системой должен использоваться графический оконный интерфейс, разработанный в среде Microsoft Visual Studio.

Система должна иметь понятный набор меню и визуальных графических инструментов, поддерживающих ввод с помощью клавиатуры и мыши.

Интерфейс должен быть привлекательным, удобным практичным, разумно и понятно скомпонованным и соответствовать нормам эргономики.

Система должна быть удобна в сопровождении и легко переносимой на другие платформы.

1.4 Планирование разработки и расчет бюджета

Планирование процесса разработки системы – одна из основных задач управления при создании нового проекта. Одной из особенностей процесса планирования в проектном виде деятельности является постоянная коррекция и поддержка актуальности планов сначала и до стадии завершения.

Для планирования разработки приложения пользователя для системы автозаполнения БД была выбрана диаграмма Ганта. Данная диаграмма является одним из наиболее популярных, простых в исполнении и легко осваиваемых инструментов планирования процесса, управления задачами, оценки ресурсных составов и взаимосвязей работ [21].

Диаграмма Ганта представляет собой столбчатую диаграмму, где в столбцах указаны даты, а горизонтальные линии наглядно отображают длительность задач, так же в ней указана их взаимосвязь и ответственные за конкретную задачу.

Для представления планирования разработки был выбран Microsoft Project 2016. Он даёт возможность заполнения отдельных задач и ресурсов. Заполненный лист ресурсов представлен на рисунке 7.

	Название ресурса	Тип	Единицы измерения материало	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочны	Затраты на использ.	Начисление	Базовый календарь
1	Аналитик	Трудовой		А		100%	110,00р./ч	200,00р./ч	0,00р.	Пропорциональн	Стандартный
2	Программист приложения	Трудовой		П1		100%	160,00р./ч	300,00р./ч	0,00р.	Пропорциональн	Стандартный
3	Тестировщик	Трудовой		Т		100%	135,00р./ч	250,00р./ч	0,00р.	Пропорциональн	Стандартный
4	Компьютер	Материальны		К			35 000,00р.		0,00р.	Пропорциональн	
5	Печать	Затраты		П						Пропорциональн	
6	Интернет	Затраты		И						Пропорциональн	
7	Программист ПС	Трудовой		П2		100%	160,00р./ч	300,00р./ч	0,00р.	Пропорциональн	Стандартный

Рисунок 7 – Ресурсы проекта

В процессе планирования необходимо выделить основные задачи, подзадачи, длительность их выполнения, а так же распределить ресурсы по задачам. В Microsoft Project существует таблица с общим представлением данных. Данная таблица представлена на рисунке 8.

ЛИСТ ЗАДАЧ

	Название задачи	Длительно	Начало	Окончание	Предше	Названия ресурсов
1	Начало реализации проекта	0 дней	Пн 02.03.20	Пн 02.03.20		Компьютер[0]
2	**Анализ предметной области и постановка задач н**	**11 дней**	**Вт 03.03.20**	**Вт 17.03.20**		
3	Анализ актуальности разработки	2 дней	Вт 03.03.20	Ср 04.03.20	1	Аналитик;Интернет
4	Сравнительный анализ систем-аналогов	6 дней	Вт 03.03.20	Вт 10.03.20	1	Аналитик[50%];Программист приложения[50%];Интернет
5	Постановка задачи на разработку системы	3 дней	Ср 11.03.20	Пт 13.03.20	4;3	Аналитик;Интернет
6	Постановка задачи на разработку приложения пс	2 дней	Пн 16.03.20	Вт 17.03.20	5	Аналитик;Интернет
7	Анализ предметной области и постановка задач	0 дней	Вт 17.03.20	Вт 17.03.20	6	
8	**Разработка требований к системе**	**12 дней**	**Ср 18.03.20**	**Чт 02.04.20**	**2**	
9	Разработка требований к архитектуре системы	2 дней	Ср 18.03.20	Чт 19.03.20	7	Программист приложения;Интернет
10	**Разработка требований к приложению пользова**	**10 дней**	**Пт 20.03.20**	**Чт 02.04.20**	**2**	
11	Функциональные требования	4 дней	Пт 20.03.20	Ср 25.03.20	9	Аналитик;Интернет;Программист приложения
12	Требования к архитектуре	3 дней	Чт 26.03.20	Пн 30.03.20	11	Интернет;Программист приложения
13	Трбования к аппаратно-программному окружен	1 день	Вт 31.03.20	Вт 31.03.20	12	Интернет;Программист приложения
14	Выбо поддерживаемых СУБД	1 день	Ср 01.04.20	Ср 01.04.20	13	Интернет;Программист приложения
15	Выбор средств разработки системы	1 день	Чт 02.04.20	Чт 02.04.20	14	Интернет;Программист приложения
16	Разработка требований к системе завершена	0 дней	Чт 02.04.20	Чт 02.04.20	15	
17	**Разработка приложения пользователя для систем**	**35 дней**	**Пт 03.04.20**	**Чт 21.05.20**	**10;8**	
18	Разработка интерфейса пользователя	17 дней	Пт 03.04.20	Пн 27.04.20	16	Программист приложения
19	Разработка бизнес-логики	15 дней	Вт 28.04.20	Пн 18.05.20	18	Программист приложения
20	Разработка интерфейса взаимодействия	3 дней	Вт 19.05.20	Чт 21.05.20	19	Программист приложения
21	Разработка приложения завершена	0 дней	Чт 21.05.20	Чт 21.05.20	20	
22	**Тестирование приложения пользователя**	**5 дней**	**Пт 22.05.20**	**Чт 28.05.20**	**17**	
23	Выбор метода тестирования	1 день	Пт 22.05.20	Пт 22.05.20	21	Тестировщик
24	Проведение тестирования	4 дней	Пн 25.05.20	Чт 28.05.20	23	Программист приложения[20%];Тестировщик
25	Тестирование приложения пользователя заверш	0 дней	Чт 28.05.20	Чт 28.05.20	24	
26	Интеграция с ПС взаимодействия с СУБД	1 день	Пт 29.05.20	Пт 29.05.20	22	Программист приложения[50%];Программист ПС[50%]
27	Тестирование собранной системы	3 дней	Пн 01.06.20	Ср 03.06.20	26	Тестировщик
28	**Завершение проекта**	**16 дней**	**Чт 04.06.20**	**Чт 25.06.20**	**27**	
29	Составление документации	14 дней	Чт 04.06.20	Вт 23.06.20	27	Аналитик;Печать;Программист приложения;Интернет;Программист ПС
30	Ввод в эксплуатацию	2 дней	Ср 24.06.20	Чт 25.06.20	29	Программист приложения;Программист ПС
31	Проект завершен	0 дней	Чт 25.06.20	Чт 25.06.20	30	
32	Конец проекта	0 дней	Чт 25.06.20	Чт 25.06.20	31	

Рисунок 8 – Перечень задач и ресурсов проекта

После описания задач и ресурсов Microsoft Project создаёт графическое представление календарного плана проекта. На рисунке 9 представлена диаграмма Ганта для разработанных задач.

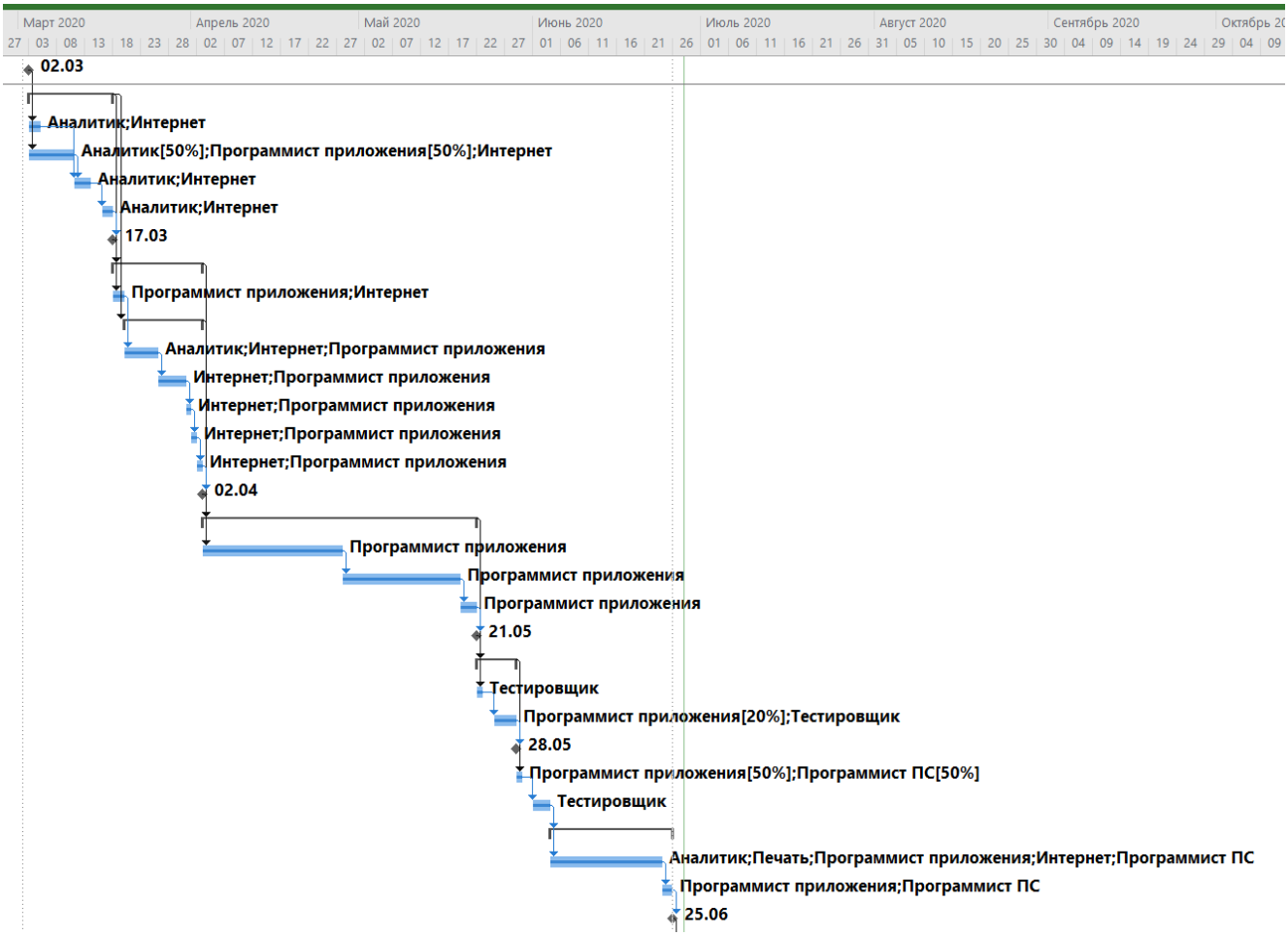
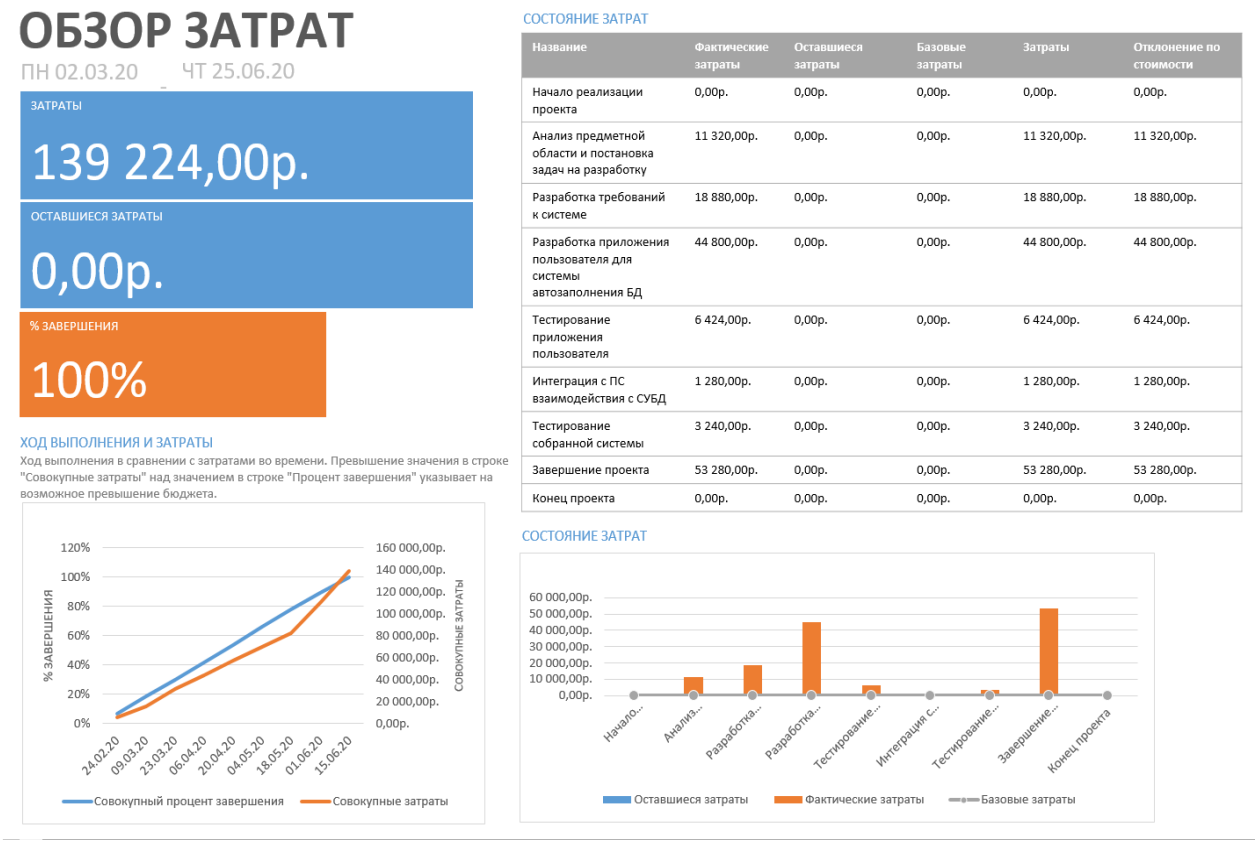


Рисунок 9 – Диаграмма Ганта

Также Microsoft Project 2016 включает набор готовых отчетов и панелей мониторинга, который помогает контролировать общее состояние проекта и проверять соответствие его компонентов графику [22].

Текущее состояние проектных затрат, информация о запланированных, оставшихся, фактических, совокупных, базовых затратах и процент завершения представлены на вкладке «Обзор затрат» в панели мониторинга. Данный отчет подробно представлен на рисунке 10.

По результатам отчета видно, что фактические затраты реализации приложения пользователя для системы автозаполнения БД составили 139 224,00 р., длительность проекта, с учетом праздников и выходных дней составила 115 дней.



СОСТОЯНИЕ ЗАТРАТ

Название	Фактические затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало реализации проекта	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.
Анализ предметной области и постановка задач на разработку	11 320,00р.	0,00р.	0,00р.	11 320,00р.	11 320,00р.
Разработка требований к системе	18 880,00р.	0,00р.	0,00р.	18 880,00р.	18 880,00р.
Разработка приложения пользователя для системы автозаполнения БД	44 800,00р.	0,00р.	0,00р.	44 800,00р.	44 800,00р.
Тестирование приложения пользователя	6 424,00р.	0,00р.	0,00р.	6 424,00р.	6 424,00р.
Интеграция с ПС взаимодействия с СУБД	1 280,00р.	0,00р.	0,00р.	1 280,00р.	1 280,00р.
Тестирование собранной системы	3 240,00р.	0,00р.	0,00р.	3 240,00р.	3 240,00р.
Завершение проекта	53 280,00р.	0,00р.	0,00р.	53 280,00р.	53 280,00р.
Конец проекта	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.

Рисунок 10 – Отчет «Обзор затрат»

Исходя из данных о сроках создания и бюджете, необходимых для разработки системы, можно сделать вывод, что создание приложения пользователя для системы автозаполнения БД является экономически выгодным.

2 Разработка требований к приложению пользователя для системы автозаполнения баз данных

2.1 Функциональные требования

Приложение пользователя должно обеспечивать удобную и комфортную работу пользователя и управлять подключением к серверам и базам данных, работой с БД, работой с правилами заполнения и процессом генерации данных. Так же приложение пользователя должно осуществлять удобство использования программных средств взаимодействия с СУБД.

Для наглядного представления функций систем использованы диаграммы общецелевого языка визуального моделирования UML. Функциональные возможности приложения пользователя для системы автозаполнения БД представлены UML диаграммой вариантов использования. Диаграмма приведена на рисунке 11.

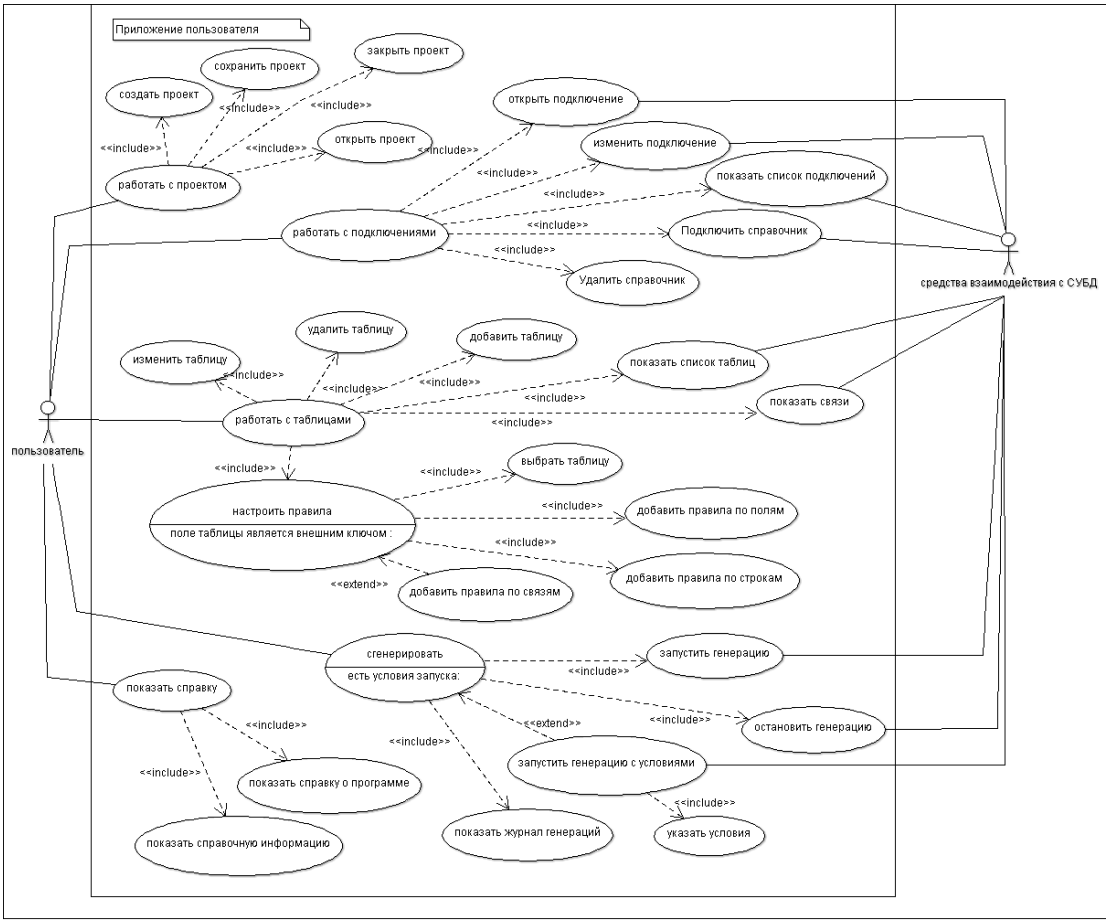


Рисунок 11 – Диаграмма вариантов использования. Функциональные возможности приложения

Краткое описание каждого варианта использования представлено в таблице 3.

Таблица 3 – Краткое описание вариантов использования

Цель	Краткое описание
1	2
Работать с проектом	Приложение пользователя обеспечивает работу с проектом, его открытие сохранение и закрытие
Работать с подключениями	Обеспечивает работу с подключениями, открывает, изменяет, показывает доступные, а также осуществляет подключение справочников
Работать с таблицами	Обеспечивает работу с таблицами БД, их просмотр, модификацию и настройку правил
Сгенерировать	Обеспечивает управление генерацией: запуск, остановку и просмотр журнала
Показать справку	Даёт пользователю возможность просмотра информации о работе в системе и о самой программе
Создать проект	Создает новый проект для работы в системе
Открыть проект	Открывает уже созданный проект
Сохранить проект	Сохраняет проект с будущей возможностью просмотра и загрузки
Закрыть	Закрывает проект и программу
Показать список подключений	Отправляет запрос к средствам взаимодействия для демонстрации существующих соединений
Открыть подключение	Отправляет запрос к средствам взаимодействия для работы для установки соединения
Изменить подключение	Отправляет запрос к средствам взаимодействия для переключения к другому серверу или БД

Продолжение таблицы 3

1	2
Добавить таблицу	Создаёт новую таблицу для генерации данных
Изменить таблицу	Редактирует таблицу в списке генерации
Удалить таблицу	Удаляет таблицу в списке генерации
Показать список таблиц	Отправляет запрос к средствам взаимодействия для просмотра списка таблиц БД
Показать связи	Отправляет запрос к средствам взаимодействия для просмотра связей между таблицами БД
Настроить правила	Обеспечивает настройку правил генерации для выбранной таблицы или поля
Выбрать таблицу	Организует работу с отдельной таблицей
Добавить правила по полям	Настраивает правила генерации для выбранного поля выбранной таблицы
Добавить правила по строкам	Настраивает правила генерации строк в каждой таблице
Добавить правила по связям	Настраивает правила генерации при наличии связей между таблицами
Запустить генерацию	Отправляет запрос к средствам взаимодействия для генерации данных
Запустить генерацию с условиями	Отправляет запрос к средствам взаимодействия для генерации данных
Остановить генерацию	Отправляет запрос к средствам взаимодействия для прерывания генерации при желании пользователя
Показать журнал генераций	Обеспечивает просмотр истории генерации
Показать справку о программе	Даёт пользователю возможность просмотра информации о программе
Показать справочную информацию	Даёт пользователю возможность просмотра информации о работе в системе

В таблице 4 приведены сценарий и полное описание варианта использования «настроить правила».

Таблица 4 – Сценарий и полное описание варианта использования «Настроить правила»

Название пункта	Описание
Наименование	Настроить правила
Краткое описание	Обеспечивает написание правил генерации
Основные действующие лица	Пользователь
Второстепенные действующие лица	нет
Предусловие	Пользователь подключил основную базу данных и видит список всех таблиц
Основной сценарий	1. Пользователь выбирает таблицу 2. Пользователь нажимает «Правила по полям» 3. Приложение показывает информацию о полях выбранной таблицы 4. Пользователь выбирает необходимое правило заполнения для поля 5. Пользователь выбирает необходимые настройки 6. Пользователь нажимает «Сохранить» 7. Приложение сохраняет выбранные настройки для поля 8. Пользователь выбирает «правила по строкам» 9. Приложение отображает форму для настройки правила по строкам 10. Пользователь вводит необходимые данные 11. Пользователь нажимает «Сохранить» 12. Приложение сохраняет настройку по строкам 13. Приложение отправляет запрос программным средствам взаимодействия с СУБД для проверки наличия связей Точка расширения: поле таблицы является внешним ключом 14. Приложение получает отрицательный ответ

Продолжение таблицы 4

Наименование	Описание
Основной сценарий (продолжение)	15. Приложение позволяет выйти из настройки правил 16. Пользователь выходит в меню «таблица»
Постусловие	Приложение сохранило правила генерации по строкам и по полям
Альтернативные сценарии	14.1 Приложение получает положительный ответ от программных средств взаимодействия с СБУД 14.2 Приложение уведомляет о наличии в поле таблицы внешнего ключа таблицы 2 14.3 Приложение проверяет наличие содержимого в таблице 2 14.4 Приложение открывает форму настройки правил по связям 14.5 Пользователь заполняет правила по связям 14.6 Пользователь нажимает «Сохранить» 14.7 Приложение сохраняет правила по связям 14.8 Пользователь выходит в меню «таблица»

Пользователь, может повторить действия при необходимости изменить или удалить правила.

Настроенные правила должны сохраняться на стороне приложения пользователя, а передача правил заполнения в программные средства (ПС) взаимодействия с СУБД должна происходить при запуске генерации.

Для наглядного представления работы системы при запуске генерации была создана диаграмма последовательности. Диаграмма последовательности – диаграмма UML, которая применяется для визуализации процесса взаимодействия объектов (выделенных частей системы) и описывается для более детальной логики сценариев диаграммы прецедентов [24].

Взаимодействие пользователя с приложением пользователя и ПС взаимодействия с СУБД в рамках прецедента «Запустить генерацию» представлено на рисунке 12.

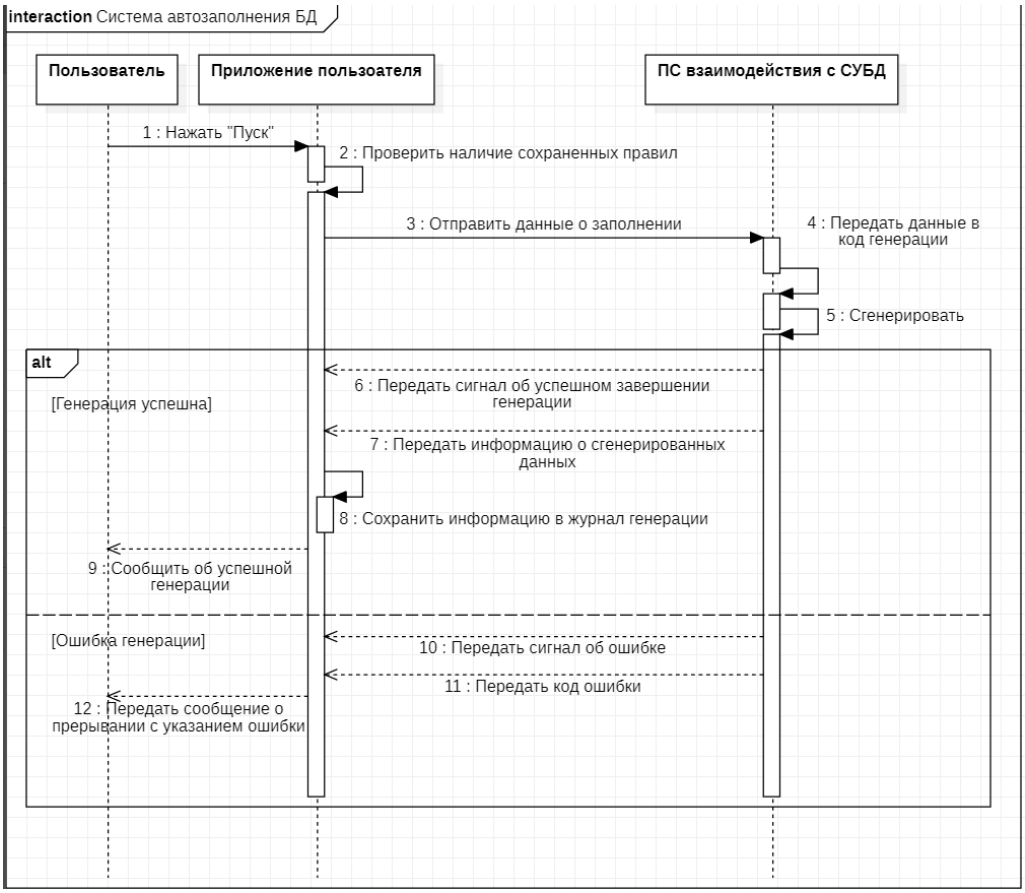


Рисунок 12 – Взаимодействие частей системы прецедента «Запустить генерацию» (диаграмма последовательности)

Генерация данных происходит на стороне ПС взаимодействия с СУБД на основе правил генерации, внесенных пользователем и сохраненных на стороне приложения пользователя.

2.2 Требования к программной архитектуре

2.2.1 Требования к обобщенной архитектуре системы

Для реализации средства автоматизированного заполнения баз данных была выбрана клиент-серверная архитектура [23].

Клиент-сервер – это архитектура, в которой участвуют две стороны: клиент – заказчик услуг и сервер – поставщик услуг. Клиент отправляет запрос на сервер с целью выполнения задач или получения информации,

сервер в свою очередь обрабатывает полученный запрос и возвращает клиенту результат.

Выделяют следующие преимущества данной архитектуры:

– разделение программного кода клиентского и серверного приложений;

– низкие требования к клиентским ЭВМ;

– основная часть вычислительных операций выполняется на сервере;

– клиентское приложение не требует администрирования;

– разделение задач разработки и, следовательно, более быстрое ее выполнение;

– гибкость и масштабируемость;

– высокая степень защиты локальной сети.

В системе автоматизированного заполнения БД роль клиента играет приложение пользователя. Задачами этого уровня являются генерация случайных данных, сбор и отправка на второй уровень введенных пользователем данных, формирование пользовательского запроса и вывод на экран пользователя информации в удобном виде. На стороне клиента содержатся все инструменты для взаимодействия с БД: создания таблиц, добавления и удаления записей.

В качестве сервера выступает СУБД, которая предоставляет доступ к таблицам и метаданным БД. СУБД принимает запрос от приложения пользователя и выполняет необходимые действия с данными, после чего возвращает результат.

Детально архитектура разрабатываемой системы автоматизированного заполнения БД показана в виде диаграммы развертывания на рисунке 13.

На диаграмме представлена система автозаполнения БД, которая состоит из приложения пользователя и программных средств взаимодействия с СУБД, включающих набор динамически подключаемых библиотек для взаимодействия с конкретным классом СУБД.

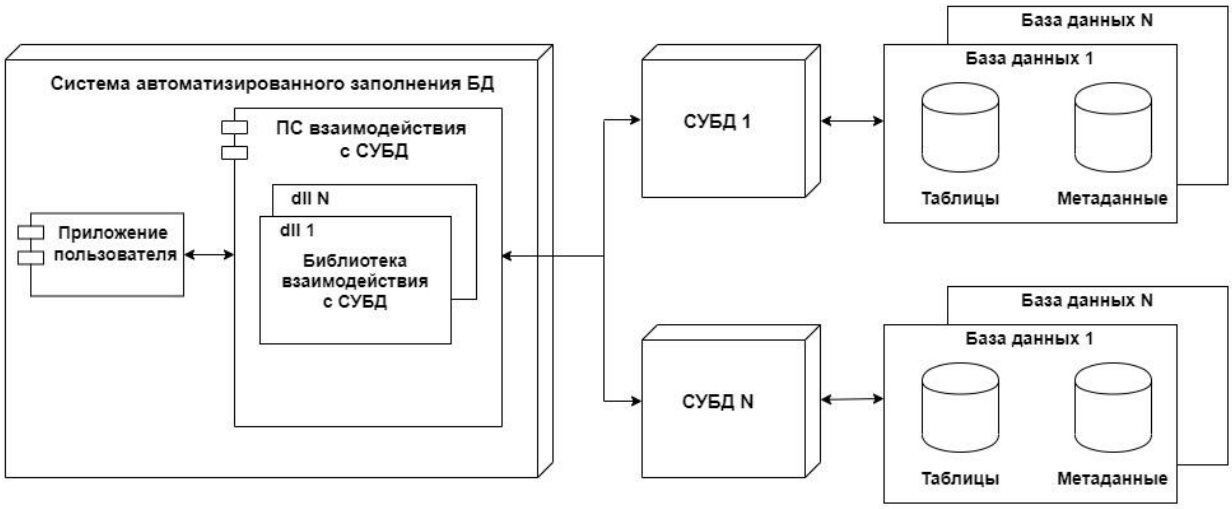


Рисунок 13 – Архитектура системы (диаграмма развертывания)

Каждая СУБД имеет доступ данным и метаданным нескольких БД. Число доступных СУБД будет увеличиваться по мере подключения к приложению пользователя новых dll-библиотек. Такая архитектура обеспечит быстрое расширение возможностей системы без необходимости постоянно вносить изменения в приложение пользователя.

2.2.2 Требования к архитектуре приложения пользователя для системы автозаполнения баз данных

На сегодняшний день существует несколько архитектурных шаблонов систем, помогающих организовать удобные для разработки, объединения и тестирования приложения. Рассмотрим самые популярные из них.

Структура Model-View-Controller (MVC, «Модель-Представление-Контроллер») – схема, разделяющая приложение на три части:

– модель (Model) – отвечает на команды контроллера, предоставляет и обновляет данные;

– представление (View) – после изменения модели представление отображает пользователю данные из модели;

– контроллер (Controller) – реагирует на действия пользователя и сообщает модели о предстоящем изменении [25].

Графическое отображение структуры представлено на рисунке 14.

Рисунок 14 – Структура MVC

Идея данного паттерна состоит в полном разделении между бизнес-логикой (работой приложения) и логикой отображения (внешним представлением приложения). Такая структура уменьшает объем кода и позволяет вести разработку частей независимо друг от друга. Также при использовании различных контроллеров, можно в ответ на каждое действие пользователя получать разное поведение. Существование модели в структуре даёт возможность создания нескольких вариантов отображения.

Структура Model-View-Presenter (MVP, «Модель-Представление-Презентер») – одна из модификаций паттерна MVC. MVP также состоит из трёх частей, но «Контроллер» заменяется «Презентером»:

– модель (Model) – содержит данные для отображения

– представление (View) – отображает данные из модели, обращается к презентеру за обновлениями, перенаправляет события от пользователя в презентер;

– презентер (Presenter) – осуществляет взаимодействие между моделью и представлением, хранит в себе всю бизнес-логику; при необходимости получает данные из хранилища и преобразует для отображения во представление [26].

Наглядное представление структуры шаблона MVP представлено на рисунке 15.

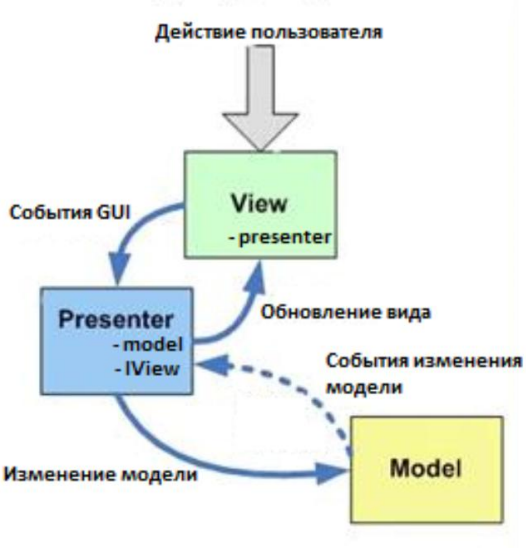


Рисунок 15 – Структура MVP

В данном случае создаётся абстрактный интерфейс для представления и презентер работает уже только с абстракцией, а не с самим представлением. Это решает проблему отделения отображения от бизнес-логики и трудности при тестировании.

Приложения пользователя для системы автозаполнения БД необходимо создать на основе структуры MVC.

2.3 Требования к аппаратно-программному окружению приложения пользователя для системы автозаполнения баз данных

Для разработки приложения пользователя для системы автоматической генерации данных необходимы следующие минимальные требования к программно-аппаратным средствам:

a) операционная система: Microsoft Windows 7, 8.1, 10;

b) Microsoft Visual Studio v.16 или выше

c) Microsoft Framework .NET v.4.8 или выше;

d) введение проекта в git для возможности соединения приложения пользователя и программных средств взаимодействия с СУБД

e) параметры компьютера:

– процессор 32-разрядный или 64-разрядный с тактовой частотой 2,5 ГГц или выше;

– оперативная память не менее 4 ГБ;

– свободное место не менее 20 ГБ на жестком диске;

– графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

Для разработки необходимо подключение к внешним ресурсам глобальной сети Internet.

2.4 Обоснование выбора поддерживаемых СУБД

Разработанная система автоматического заполнения БД направлена на работу с СУБД, которые отвечают следующим общим требованиям:

– реляционный тип;

– надежность;

– эффективность;

– популярность.

В 2019 году компания Gartner выпустила отчет по очередному исследованию в области систем управления базами данных [15]. Результаты исследования показали, какие СУБД являются наиболее эффективными и востребованными для реализации систем оперативного учета. На рисунке 16 представлен так называемый магический квадрант, на котором распределены ИТ-вендоры, поставляющие СУБД. Лидирующую позицию занимает компания Microsoft. В области лидеров также присутствуют Oracle, Amazon Web Services, SAP, Google и InterSystems.

В системе предполагается использование следующих популярных СУБД:

– MS SQL Server;

– Oracle;

– PostgreSQL;

– MySQL.

Рисунок 16 – Магический квадрант для операционных СУБД по версии Gartner

В таблице 5 приведены некоторые характеристики выделенных СУБД.

Таблица 5 – Характеристики современных СУБД

Характеристика	MS SQL Server	Oracle	PostgreSQL	MySQL
Интерфейс	GUI, SQL, другое	GUI, SQL	GUI, SQL	SQL
Поддерживаемые языки	Python, Ruby, Java, .Net, VB, PHP	C, C++, Ruby, Java, Objective-C и другие	Java, C, C++, Delphi, .Net , Python, PHP, Perl, Tcl	C, C++, Java, Ruby, Objective-C и другие
Методы доступа	ADO.NET, ODBC, JDBC, OLE DB, табличный поток данных (TDS)	Oracle (OCI), ODBC, JDBC, ODP.NET	ADO.NET, ODBC, JDBC, потоковое API для больших объектов, нативная C библиотека	ADO.NET, API, ODBC, JDBC
Ограничения размера БД	524 272 ТБ	8 ЭБ	Нет	64 ТБ на таблицу

На текущий момент система поддерживает только MS SQL Server. В процессе развития будет возможен доступ к любым другим СУБД.

2.5 Обоснование выбора средств разработки приложения пользователя для системы автозаполнения баз данных

2.5.1 Обоснование выбора языковых средств разработки

Язык программирования – формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением [17].

На сегодняшний день рынок языковых средств разработки очень велик. Для разработки системы автозаполнения баз данных наиболее удобными являются объектно-ориентированные языки. Самые популярные на данный момент языки C++, C# и Java.

Языки C# и Java были разработаны в качестве потомков языка C++, поэтому для выбора языковых средств разработки проведём сравнительный анализ языков C# и Java.

C# – это язык программирования общего назначения, который впервые появился в 2000 году в рамках инициативы Microsoft .NET. Он был разработан для общей языковой инфраструктуры (CLI) – открытой спецификации, разработанной Microsoft и стандартизированной ISO и ECMA.

Java, первоначально выпущенный Sun Microsystems в 1995 году, является языком программирования общего назначения, который был разработан с конкретной целью, позволяющей разработчикам “write once, run anywhere” – написать код единожды и запускать в любом месте. Java-приложения скомпилированы в байт-код, который может запускаться при реализации виртуальной машины Java (JVM). Подобно CLI, JVM помогает преодолеть разрыв между исходным и машинным кодом [16].

Далее перечислены некоторые сходные преимущества языков:

– оба языка компилируются в машинно-независимый и независимый от языка программирования программный код, запускаемый в рамках управляемой среды выполнения (для Java – Java Virtual Mashine, для C# - Common Language Runtime);

– автоматический сбор мусора;

– весь код помещается в пакеты, благодаря чему отсутствуют заголовочные файлы;

– все элементы должны принадлежать какому-то классу, нельзя объявлять глобальные функции и константы.

К различиям можно отнеси, что Java имеет более сложную структуру, в связи с этим обучение языку и сопровождение уже готовых систем имеют более длительные сроки, чем на C#. Но главным различием между языками является полная интеграция C# со стандартом Component Object Model (COM). Разработчики могут наследовать компоненты, созданные с помощью любого языка программирования, который поддерживается .NET. Эти компоненты в дальнейшем могут сами выступить в роли базовых для программ на других языках. Код на промежуточном языке MSIL не интерпретируется, а компилируется в машинный код с помощью JIT-компилятора [18]. Вследствие этого приложения созданные с помощью C#, выполняются быстрее, чем те, что написаны Java, именно поэтому для разработки приложения был выбран язык программирования C#.

2.5.2 Обоснование выбора инструментальных средств разработки

Наиболее распространенной и удобной для написания кода на C# является интегрированная среда разработки Microsoft Visual Studio 2019 (версия 4.16). Данная среда обеспечивает полную поддержку в написании кода на протяжении всего цикла от разработки до внедрения. На сегодняшний день Visual Studio одно из самых мощных средств разработанных компанией Microsoft. Для удобной работы на языке C# данная среда поддерживает архитектуру .NET Framework. Благодаря этой платформе, разработка на C# – это удобный и быстрый процесс. Главная

особенность каркаса .NET – общеязыковая среда исполнения Common Language Runtime (CRM), с помощью которой удобно вести разработку на разных языках [19].

Некоторые возможности среды:

– поддержка нескольких языков: C#,VB.NET, C++, F#,Delphi.NET;

– кроссплатформенность: разработка для платформ Windows, MacOS, Linux, Android, iOS, Tizen;

– широкая библиотека классов;

– среда позволяет быстро писать код за счет легкого перехода к любому символу, типу, элементу или файлу;

– возможность просмотра истории, авторов, даты изменения в строке;

– анализ кода;

– подключение облачного хранения;

– пакет Entity Framework для удобного подключения к различным СУБД;

– системы Windows Presentation Foundation (WPF) и WinForms для привлекательного взаимодействия с пользователем;

– удобный интерфейс для работы с тестами [20];

– множество возможностей для будущей Web-разработки.

Данный список всего лишь часть преимуществ разработки на Visual Studio .NET Framework. Сегодня платформа развивается и содержит множество надстроек для повышения эффективности разработки. Именно поэтому для создания системы была выбрана среда Microsoft Visual Studio.

3 Разработка приложения пользователя для системы автозаполнения баз данных

3.1 Разработка интерфейса пользователя

Интерфейс пользователя разработан с помощью Windows Forms на C# в Visual Studio. Главная форма приложения представлена на рисунке 17.

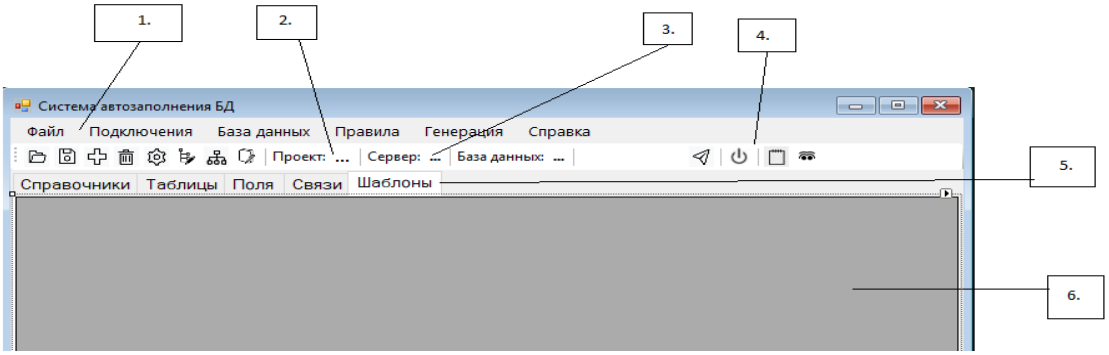


Рисунок 17 – Главная форма

Компоненты, указанные цифровыми выносками на рисунке 17 описаны в таблице 6.

Таблица 6 – Описание компонентов главной формы рисунка ТТ

№ п/п	Имя компонента	Тип компонента	Назначение компонента
1	menuStrip	меню	Компонент, описывающий меню системы
2	toolStrip	Панель инструментов	Компонент для создания панели инструментов
3	toolStripLabel	Надпись	Компонент надписи на панели инструментов
4	toolStripButton	Кнопка	Представляет кнопку на панели инструментов
5	tabControl, tabPage	Группа вкладок	Компоненты для создания страниц вкладок. tabControl управляет связанным набором страниц вкладок, tabPage описывает отдельную страницу вкладки
6	dataGridView	Таблица	Компонент предоставляет настраиваемую таблицу для отображения данных

На главной форме расположены множество активных элементов управления, представленные кнопками. Таким образом пользователю нет необходимости вводить данные с клавиатуры. В зависимости от выбранной кнопки, либо происходит вызов других форм, компоненты которых описываются далее, либо происходит активация группы вкладок tabControl и таблиц dataGridView.

Форма «Подключение» открывается при нажатии кнопки пункта меню «Добавить подключение». Данная форма представлена на рисунке 18.

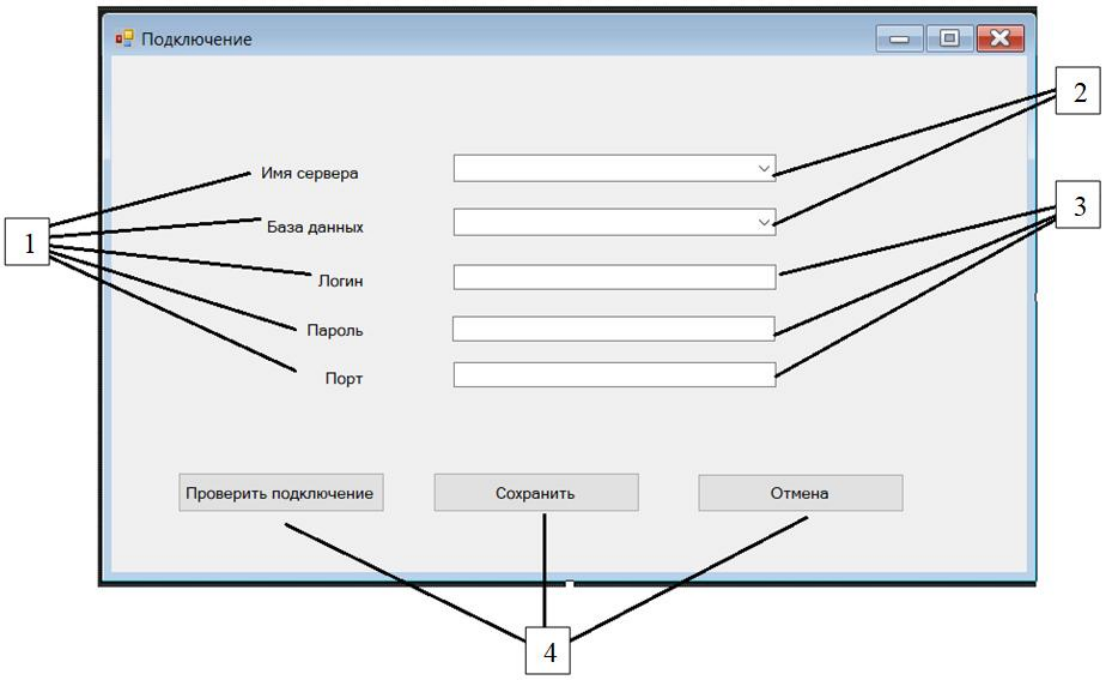


Рисунок 18 – Форма «Подключение»

Компоненты, указанные цифровыми выносками на рисунке 18 описаны в таблице 7.

Таблица 7 – Описание компонентов формы «Подключение»

№ п/п	Имя компонента	Тип компонента	Назначение компонента
1	label	Надпись	Отображает стандартную метку Windows
2	comboBox	Поле со списком	Отображает поле ввода со списком выбора
3	textBox	Поле	Текстовое поля для ввода
4	button	Кнопка	Представляет кнопку на панели инструментов

В отличие от формы «Подключение» форма «Правила автозаполнения для поля» содержит большее количество разнообразных элементов, которые предоставляют возможность не только ввода данных с клавиатуры, но и переключения между правилами. Форма «Правила автозаполнения для поля» представлена на рисунке 19.

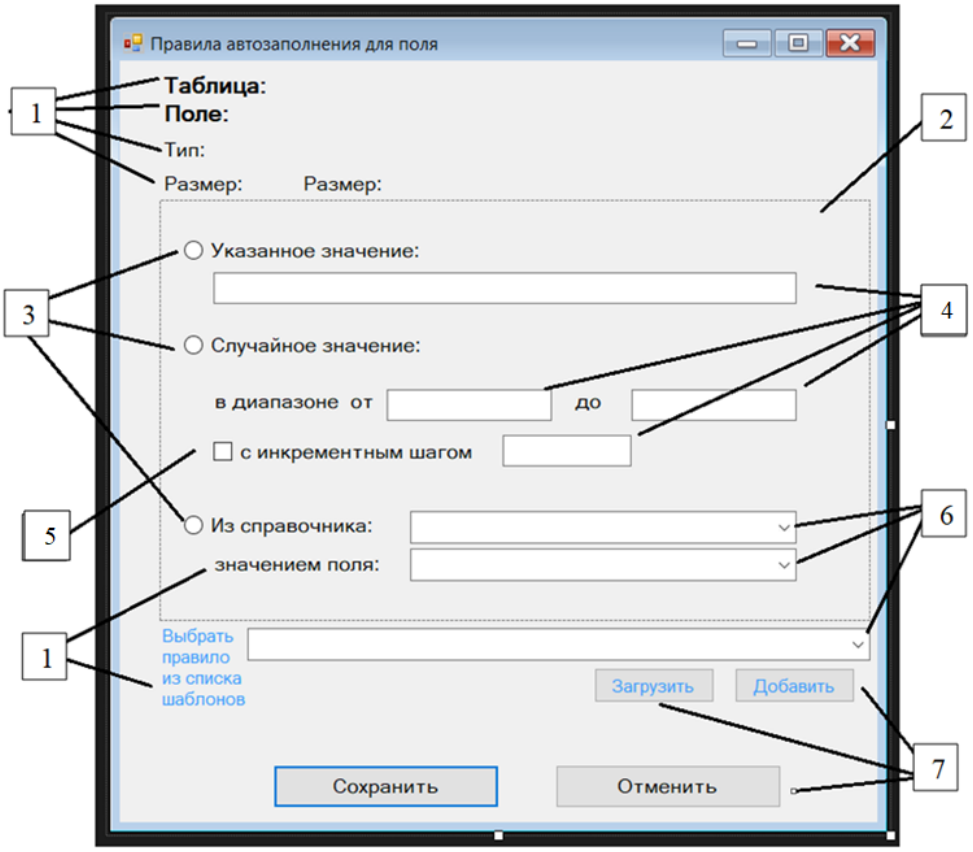


Рисунок 19 – Форма выбора правил для поля

Компоненты, указанные цифровыми выносками на рисунке 19 описаны в таблице 8.

Таблица 8 – Описание компонентов формы выбора правил

№ п/п	Имя компонента	Тип компонента	Назначение компонента
1	label	Надпись	Отображает стандартную метку Windows
2	panel	Панель	Используется для группировки коллекции элементов управления
3	radioButton	Переключатель	Позволяет выбрать один элемент управления из группы
4	textBox	Поле	Текстовое поля для ввода

Продолжение таблицы 8

№ п/п	Имя компонента	Тип компонента	Назначение компонента
5	checkBox	Флажок	Предоставляет пользователю возможность выбрать «да» или «нет»
6	comboBox	Поле со списком	Отображает поле ввода со списком выбора
7	button	Кнопка	Представляет кнопку на панели инструментов

Элементы управления, описанные в дополнительно вызываемых формах, являются стандартными элементами управления.

3.2 Разработка структуры приложения

Для наглядного отображения программной структуры приложения пользователя для системы автозаполнения БД использована диаграмма компонентов, представленная на рисунке 20.

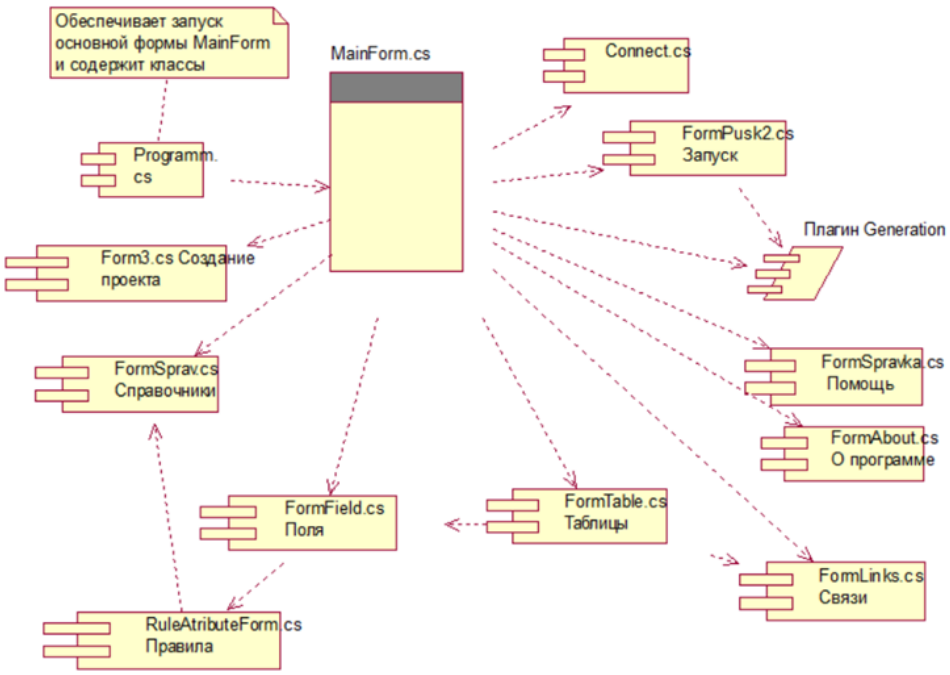


Рисунок 20 – Диаграмма компонентов

Краткое описание каждого компонента приведено далее:

– Program.cs – точка входа в приложение, обеспечивает запуск MainForm.cs и хранит в себе классы моделей;

– MainFrom.cs – главная форма приложения, содержит вызовы всех форм и логику получения и передачи данных с программными средствами взаимодействия с СУБД;

– Connect.cs – форма подключения к серверу и базе данных;

– Form3.cs – форма для создания нового проекта, позволяет выбрать директорию для сохранения файла проекта и название;

– FormTable.cs – форма для заполнения правил для таблицы;

– FromField.cs – форма для добавления, изменения или удаления таблицы;

– FormLinks.cs – форма настройки правил по связям;

– FormSprav.cs – форма для добавления справочника или классификатора с локального или удалённого сервера;

– RuleAtributeForm.cs – форма для настройки правил для поля$

– FormPusk2.cs – форма для настройки условий генерации;

– FormSpravka.cs – форма, содержащая ссылки на справочную информацию по работе в системе;

– FormAbout.cs – форма, содержащая справочную информацию о программе.

3.3 Спецификация методов приложения пользователя

Файл program.cs является главной точкой для входа приложения, который далее запускает файл MainForm.cs, содержащий основную форму. Также в файле program.cs объявлены классы типовых конструкций системы:

– public class Diction – базовый класс, структура справочника;

– public class MTable – базовый класс, структура таблицы;

– public class MRule – базовый класс, структура поля;

– public class MLinks – базовый класс, структура связи;

– public class MShabl – базовый класс, структура шаблона правила.

Так как класс MainForm является основным, в нем объявляются соответствующие объекты указанных классов, представляющие

переопределение стандартного типизированного класса List<T>. Эти объекты создаются на все время сессии системы, поэтому они объявлены статическими. Объекты представлены в листинге 1.

Листинг 1 – Статические объекты типизированного класса List<T>

```
public static List<Diction> listDics;
public static List<MLinks> listMLinks;
public static List<MTable> listMTables;
public static List<MRule> listMRules;
public static List<MShabl> listMShabls.
```

На форме класса MainForm, представленной на рисунке 17, имеются пять соответствующих вкладок, на которых размещены компоненты dataGridView, связываемые с вышеописанными типизированными списками, служащими в качестве источников данных для отображения в табличной форме.

Так как на основной и дополнительных формах содержится много стандартных обработчиков нажатия кнопок, пунктов меню и закрытия окон, который не несут информативной нагрузки, они не представлены в таблице, описывающей спецификации методов системы. Методы, которые содержат основную логику работы системы, описаны в таблице 9.

Таблица 9 – Спецификация методов

№ п/п	Название	Назначение	Входные параметры	Возвращаемые значения
1	2	3	4	5
1	новыйПроектToolStripMenuItem_Click()	создать новый проект ADD	_	Название проекта, каталог проекта
2	добавитьToolStripMenuItem_Click	Создать/изменить основное подключение к Базе данных	Название проекта	Параметры подключения: сервер, база данных
3	списокТаблицToolStripMenuItem_Click	Открыть вкладку со списком таблиц БД	Сервер, База данных	Список таблиц
4	правилаПоПолямToolStripMenuItem_Click	Открыть вкладку со списком полей по выбранной таблице	Наименование таблицы	Список полей класса List<MRule>
5	списокToolStripMenuItem_Click	Открыть вкладку со списком справочников/ классификаторов	Название проекта	Список полей класса List<Diction>

Продолжение таблицы 9

1	2	3	4	5
6	шаблоныПравилToolStripMenuItem_Click	Открыть вкладку со списком шаблонов правил	_	Список полей класса List<MShabl >
7	списокСвязейToolStripMenuItem_Click	Открыть вкладку со списком связанных по выбранной таблице	Наименование таблицы или null	Список полей класса List<MLinks >
8	toolStripButton2_Click	Обработка кнопки / пункта меню -добавить элемент в выбранной области	Наименование выбранной вкладки	_
9	toolStripButtDel_Click	Обработка кнопки / пункта меню -удалить элемент в выбранной области	Наименование выбранной вкладки	_
10	toolStripButton3_Click	Обработка кнопки / пункта меню - редактировать элемент в выбранной области изменить	Наименование выбранной вкладки	
11	MTGet	Синхронизировать данные класса в форме свойств Таблицы	MTable tt	
12	MTChange	Синхронизировать данные формы свойств Таблицы в классе		MTable tt
13	MTRess	Получить новый объект класса из формы свойств Таблицы		MTable tt
14	MLGet	Синхронизировать данные класса в формы свойств Связи	MLinks tt	
15	MLChange	Синхронизировать данные формы свойств Связи в классе		MLinks tt
16	MLRess	Получить новый объект класса из формы свойств Связи		MLinks tt
17	MDGet	Синхронизировать данные класса в формы свойств Справочника	Diction tt	
18	MDChange	Синхронизировать данные формы свойств Справочника в классе		Diction tt
19	MDRess	Получить новый объект класса из формы свойств Справочника		Diction tt

Окончание таблицы 9

1	2	3	4	5
20	SetPolya	Синхронизировать данные класса Поля в форме Правил автозаполнения поля и подгрузить шаблоны	MRule tt, String tname, List<MShabl>	
21	MFGet	Синхронизировать данные класса в форме свойств Поля	MRule mr класс поля , ss название таблицы	
22	MFRess	Получить новый объект класса из формы свойств Поля		MRule mr
23	SetDict	Загрузить список справочников в форму Правил автозаполнения	Diction dd in listDics	
24	RuleToString	Вернуть из формы установленное правило генерации		String - правило заполнения
25	SetShabl	Добавить в список на добавленные форме шаблоны	listMShabls	listMShabls - список шаблонов

Описанные в таблице 9 методы находятся в листинге программы, представленном в приложении Б.

4 Руководство пользователя

Программа Auto Data Download (ADD), «Система автоматического заполнения баз данных», далее просто «система», имеет многооконный дружелюбный интерфейс с пользователями. Она разработана, чтобы предоставить специалисту, тестировщику или профилировщику, удобный и эффективный инструмент для заполнения и тестирования баз данных.

4.1 Описание главного меню

Система имеет меню, панель инструментов и набор вкладок и форм отображающих необходимые средства. Общий вид системы представлен на рисунке 21.

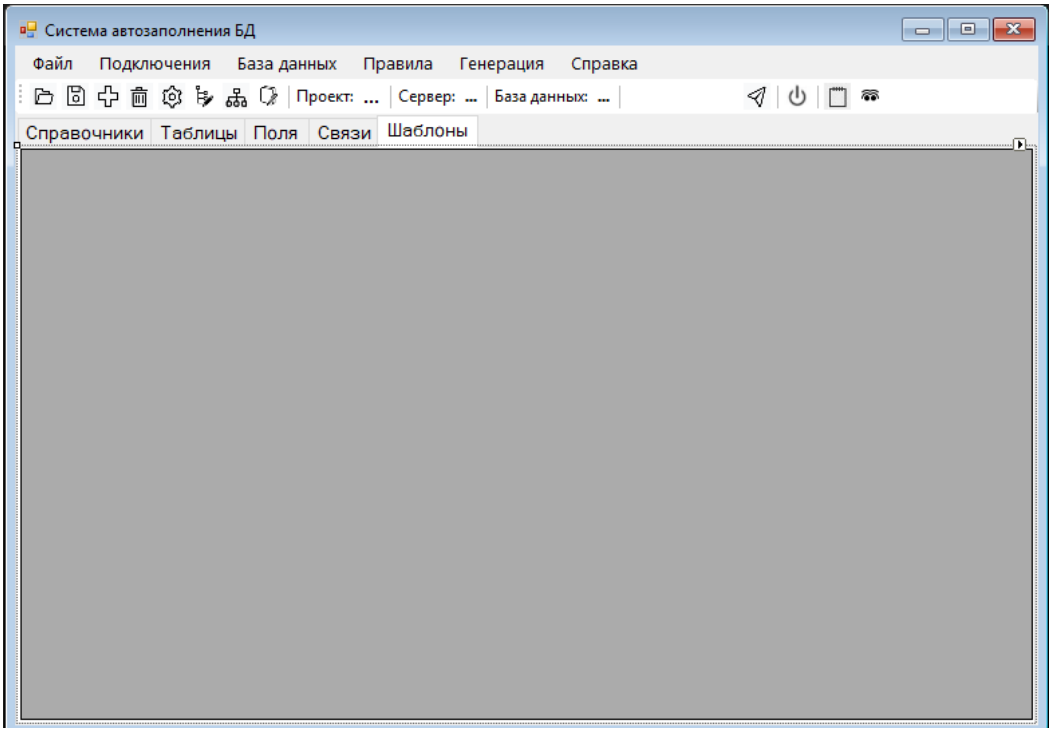


Рисунок 21 – Общий вид системы. Главное окно

Далее представлено описание меню системы, состоящего из нескольких пунктов. Меню «Файл» содержит средства для работы с проектами, показанный на рисунке 22, который позволяет создать новый проект, открыть существующий и сохранить проект для дальнейшего использования.

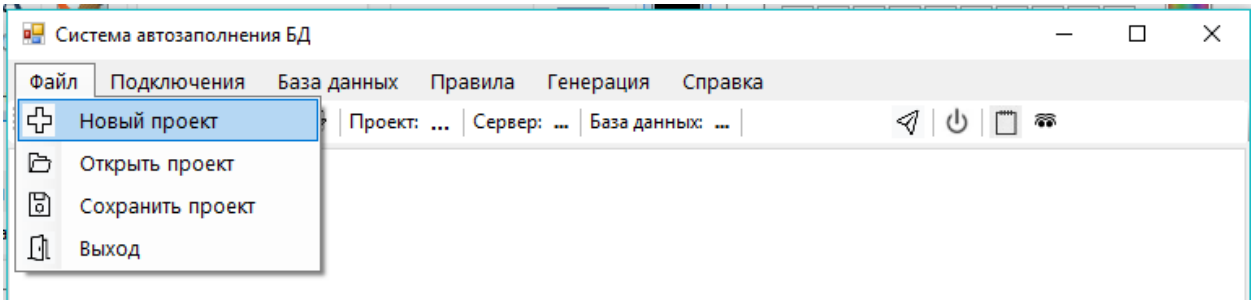


Рисунок 22 – Меню «Файл» системы

Меню «Подключения» содержит пункты, позволяющие управлять подключением к тестовой базе данных и к справочникам или классификаторам, используемым для автозаполнения базы. Данное меню представлено на рисуке 23.

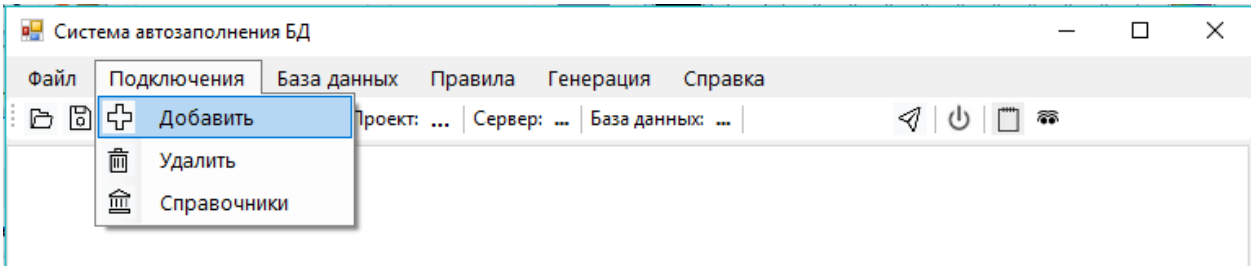


Рисунок 23 – Меню «Подключения» системы

Меню «База данных» позволяет открыть список таблиц, список полей выбранной таблицы, список связей между таблицами, и отображает уровень определения таблиц и полей для заполнения. Меню «База данных» показано на рисунке 24.

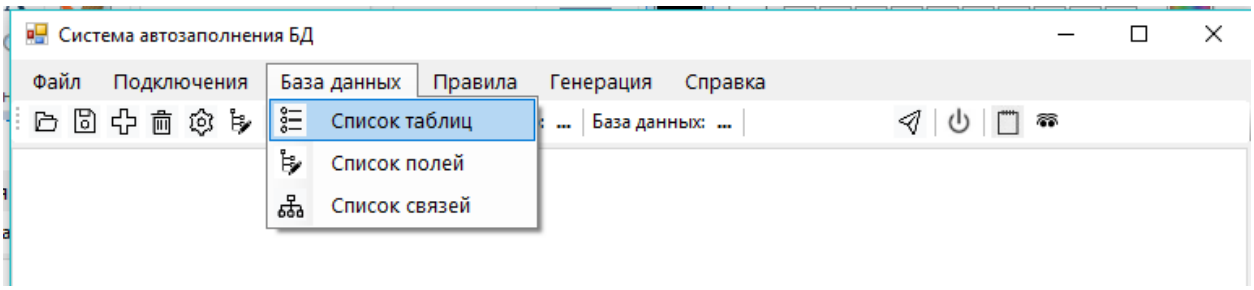


Рисунок 24 – Меню «База данных»

Меню «Правила» содержит инструменты, описывающие нижний уровень детализации процесса автозаполнения по полю таблицы, по алгоритму заполнения строк, а также в зависимости от связанности таблиц.

Меню позволяет открыть список шаблонов правил для типов данных. Вид меню «Правила» представлен на рисунке 25.

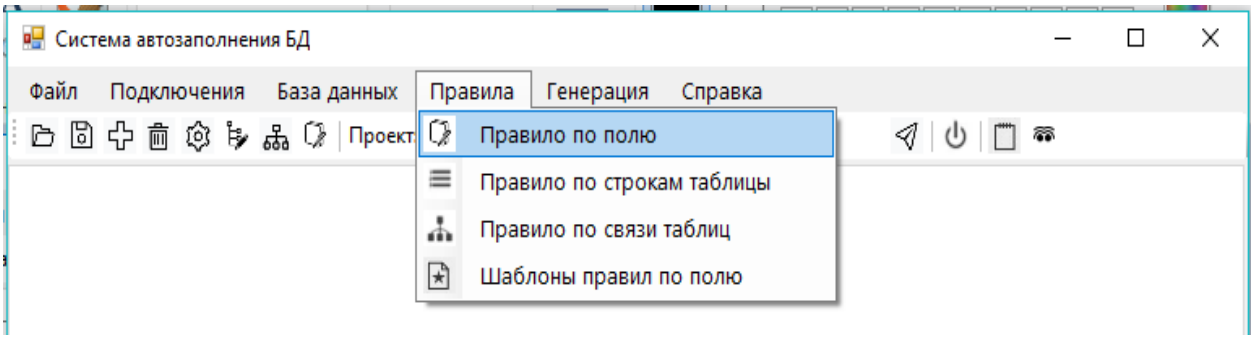


Рисунок 25 – Меню «Правила»

Меню «Генерация» содержит пункты, управляющие процессом автозаполнения выбранных полей таблиц указанными правилами. А также позволяющие просмотреть результаты генерации. Меню «генерация» открыто на рисунке 26.

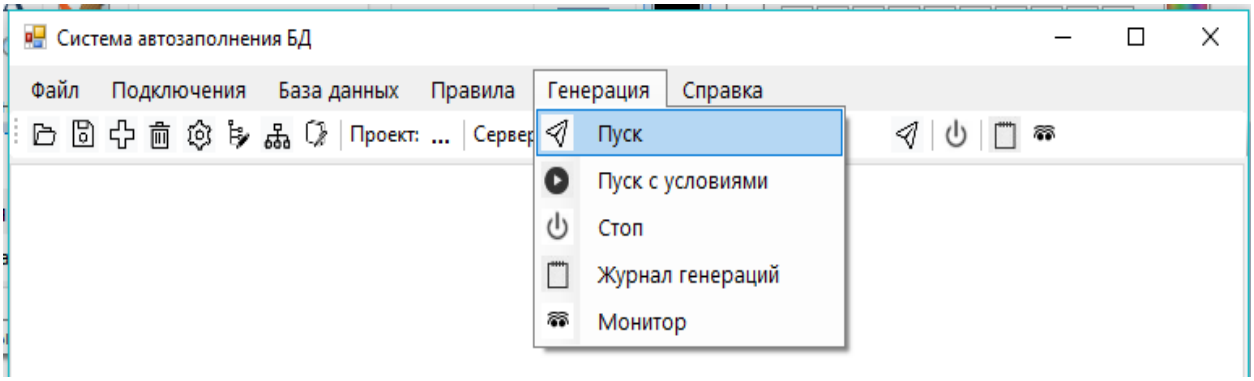


Рисунок 26 – Меню «Генерация»

Последний пункт меню, представленный на рисунке 27 – «Справка», содержит разделы описывающие инструменты данной программы.

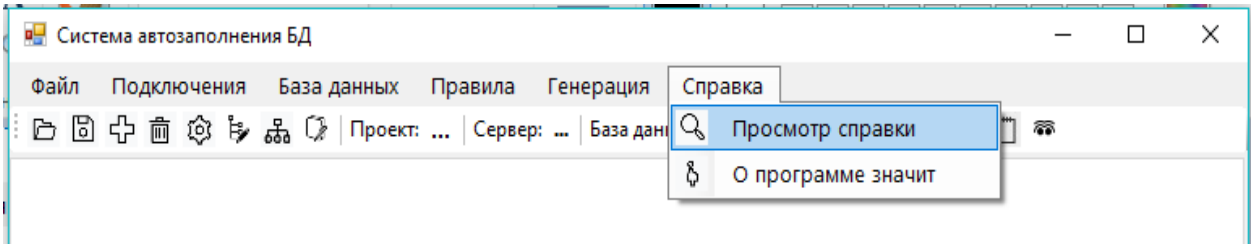


Рисунок 27 – Меню «Справка»

Кнопки добавления, изменения и удаления работаю в зависимости от открытой вкладки на рабочей области.

4.2 Работа с вкладками и дополнительными окнами

Процесс работы с системой начинается с создания нового проекта. Для этого задаются каталог на диске, который будет рабочим для этого проекта, и имя проекта, представленные на рисунках 28 и 29 соответственно.

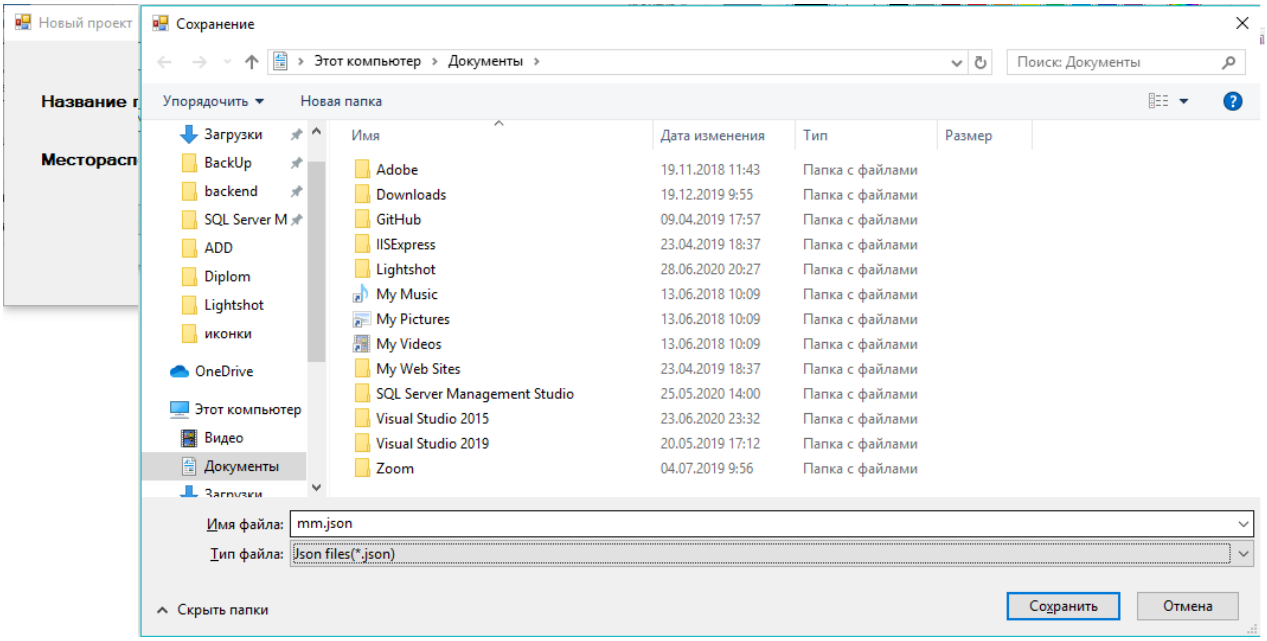


Рисунок 28 – Выбор каталога для создания проекта

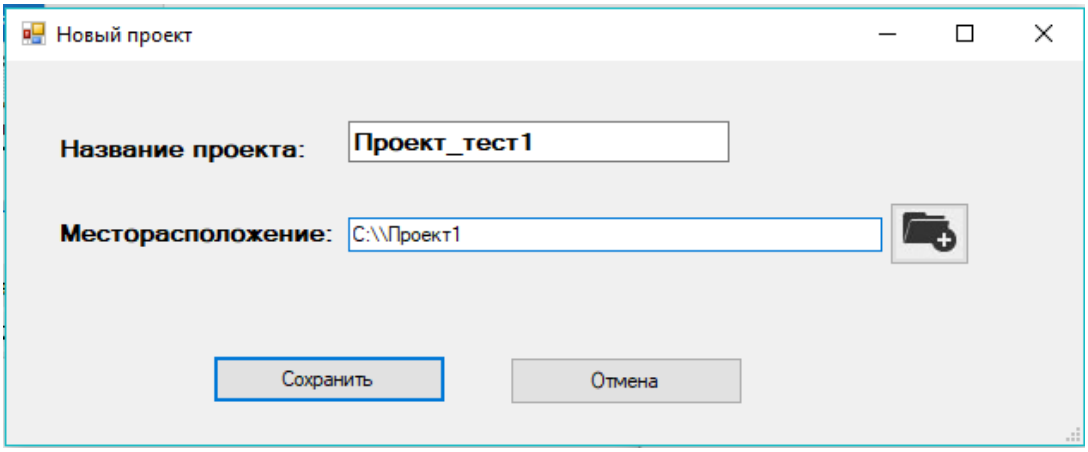


Рисунок 29 – Окно создания нового проекта

После создания проекта необходимо создать новое подключение, которое описывается в отдельном окне, представленном на рисунке 30.

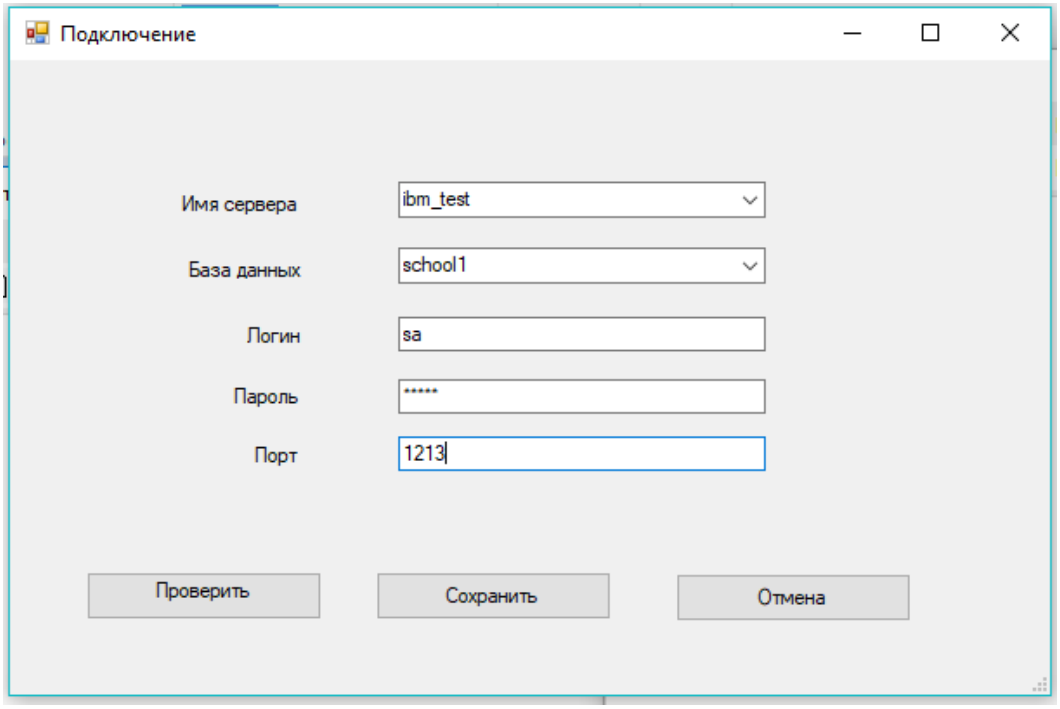


Рисунок 30 – Окно создания нового подключения

Подключение необходимых справочников и классификаторов так же находится в меню «Подключение». Для добавления нового справочника необходимо открыть список справочников и потом добавлять новый. Окно подключения нового справочника или классификатора представлено на рисунке 31.

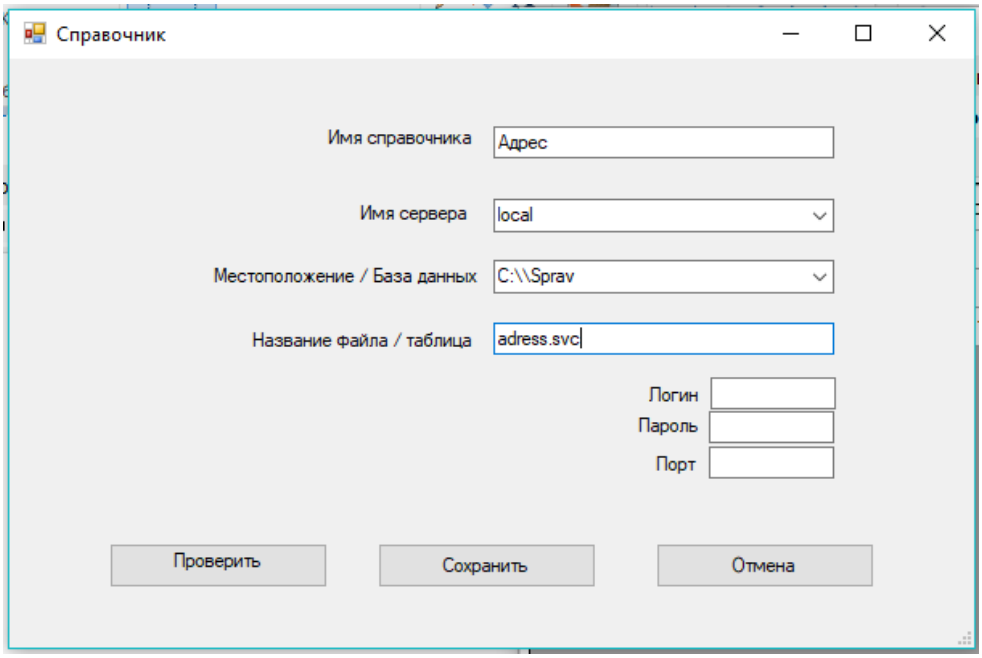


Рисунок 31 – Окно подключении нового справочника

Далее мы можем управлять подключенными классификаторами, которые представлены в виде списка с названием «Справочники», через меню или пункты панели инструментов. Вкладка «Справочники» показана на рисунке 32.

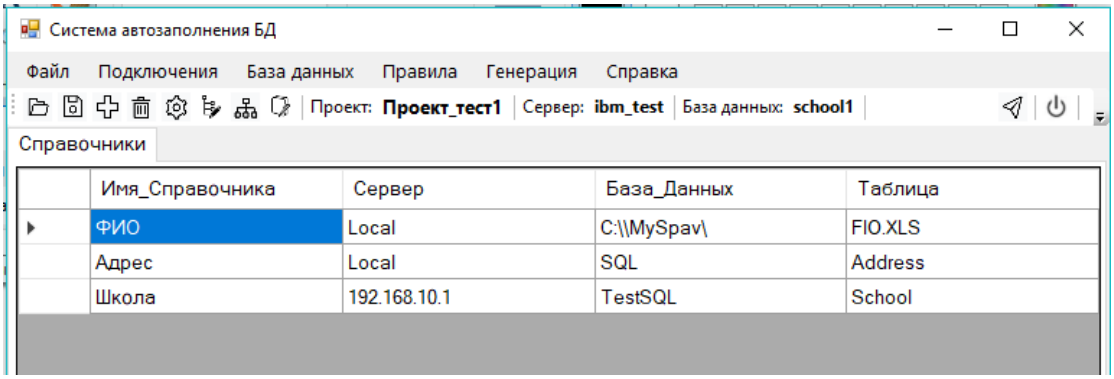


Рисунок 32 – Вкладка «Справочники»

Управление таблицами и их свойствами осуществляется на вкладке «Таблицы». Список содержит наименование таблиц, содержимое которых будет заполняться по установленным здесь правилам. Также таблице можно задать приоритет, то есть порядок вызова таблиц для автозаполнения, увидеть число внешних связей этой таблицы, а также указать число строк, которое будет добавлено процессе генерации данных. Пример работы с таблицей приведен на рисунке 33.

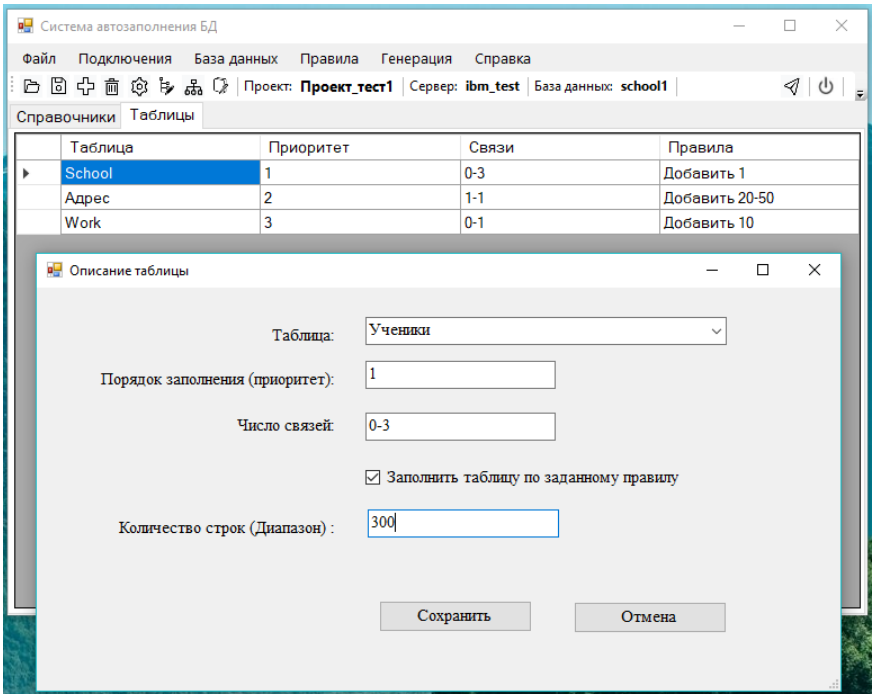


Рисунок 33 – Вкладка «Таблицы» и окно описания таблиц

Из списка таблиц, выбором пункта меню «Список полей» или нажатием кнопки на панели инструментов, открывается список полей заданной таблицы. Название вкладки содержит название таблицы и пояснение, что это список полей данной таблицы. Пример открытой вкладки полей для таблицы School представлен на рисунке 34.

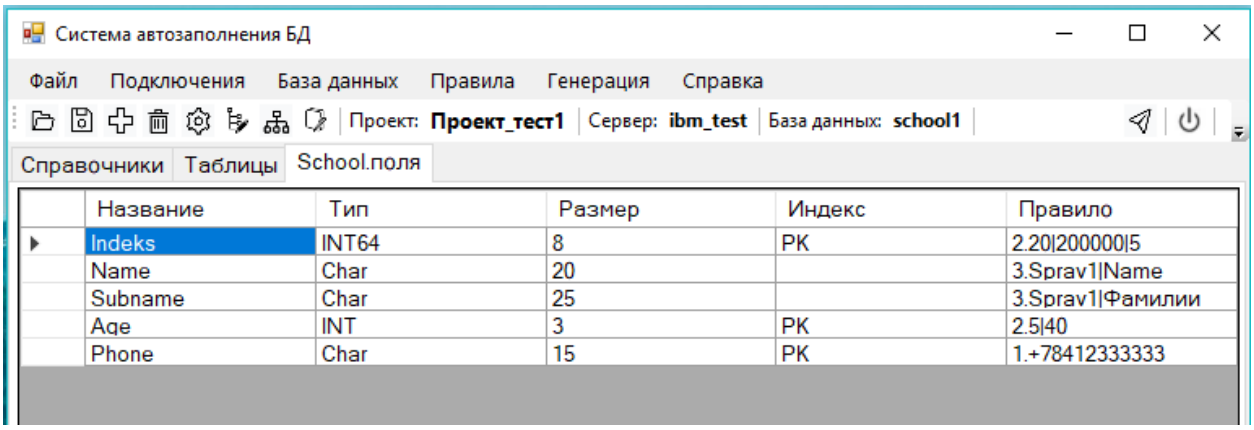


Рисунок 34 – Вкладка списка полей таблицы School

Этот список содержит только те поля, которые подлежат автозаполнению. В списке присутствует атрибут: тип и размер поля, является ли поле индексным, а также в кратком виде правило заполнения поля, начинающееся с номера правила. Изменить, добавить или удалить поле в списке можно с помощью окна свойств поля, показанного на рисунке 35.

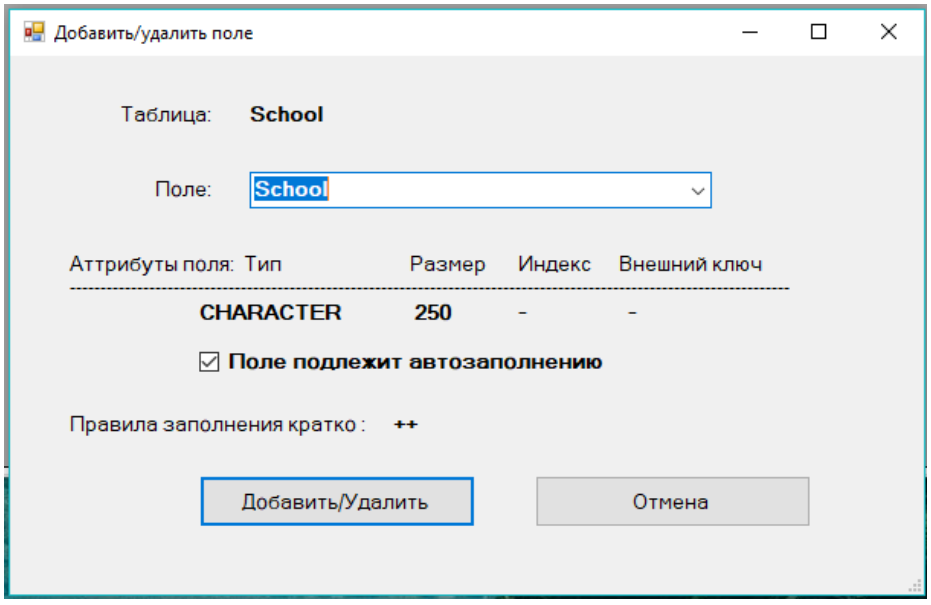


Рисунок 35 – Окно свойств поля

При выборе пункта меню «Список связей» или по нажатию кнопки на панели инструментов открывается вкладка «Связи». В ней присутствует основная таблица, связанная с ней таблица и поле, по которому осуществляется связь. Данная информация получается на основе анализа схемы выбранной базы данных. Также в таблице указано число строк, которое должно быть заполнено по данному виду связи. Вкладка «Связи» для всех таблиц представлена на рисунке 36.

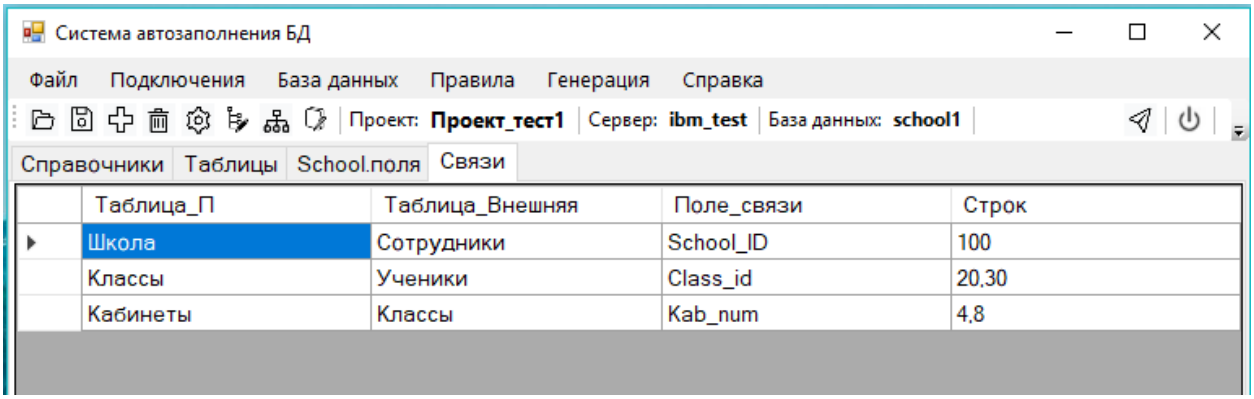


Рисунок 36 – Вкладка «Связи»

Добавить, удалить и настроить число строк на вкладке можно в вызываемом окне, представленном на рисунке 37.

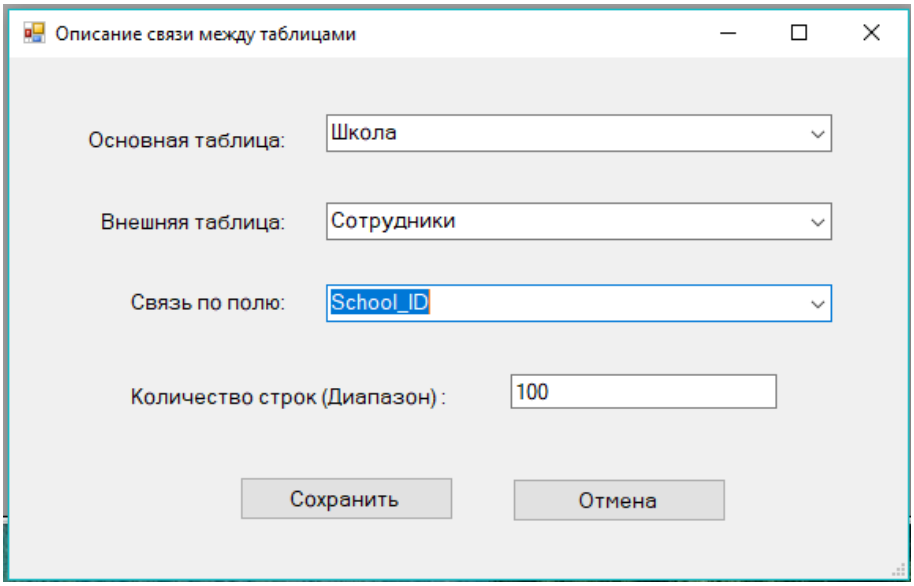


Рисунок 37 – Окно описания связей между таблицами

Из вкладки содержащей список полей можно, с помощью пункта меню «Правило по полю» или специальной кнопки на панели инструментов,

вызвать форму настройки правила автозаполнения поля. Окно настройки правил показано на рисунке 38.

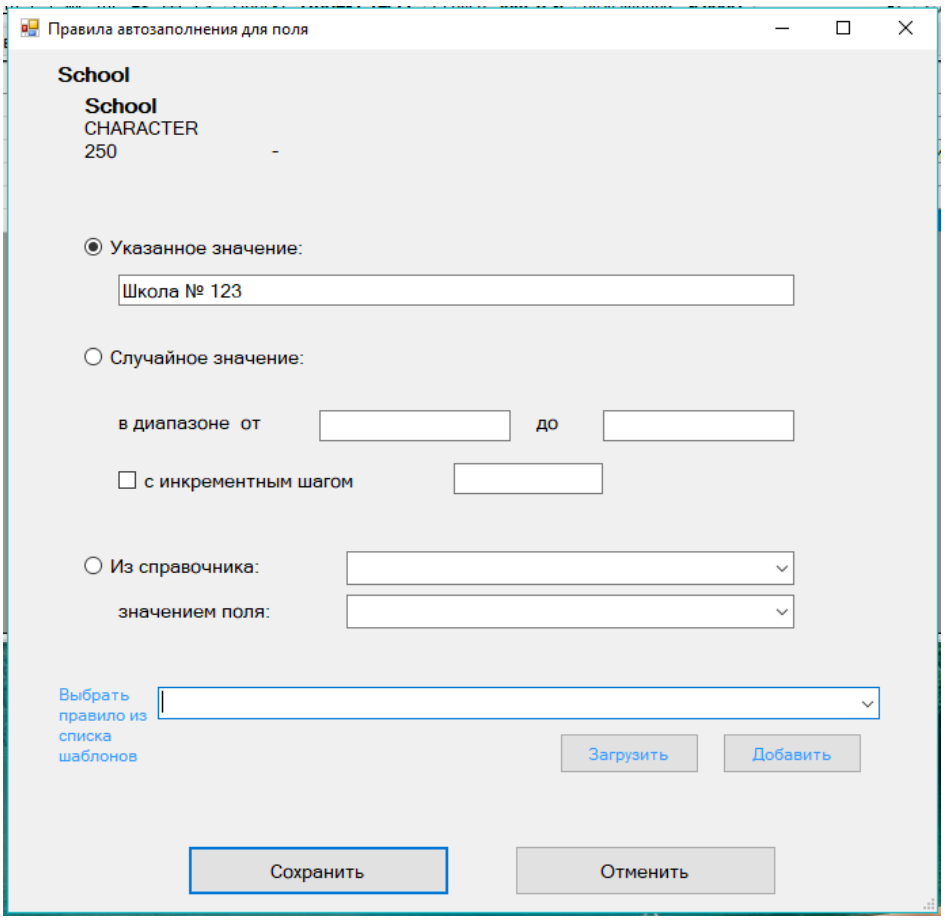


Рисунок 38 – Окно настройки правил для поля

По рисунку 38 видно, что в системе возможны три варианта заполнения поля автоматическими данными:

– заданным значением;

– случайным значением в указанном диапазоне;

– данными, из выбранного поля подключенного ранее справочника.

Также предусмотрен список шаблонов правил, для удобного сохранения настроенного варианта автозаполнения. При открытии окна настройки правила, из списка шаблонов формируется выпадающий список для поля по критерию совпадения типа поля. То есть в списке шаблонов правила группируются по типу данных поля.

В окне «Правила автозаполнения для поля» можно выбрать настройку из готового шаблона или сохранить свою настройку как новый шаблон. Пример работы с шаблоном представлен на рисунке 39.

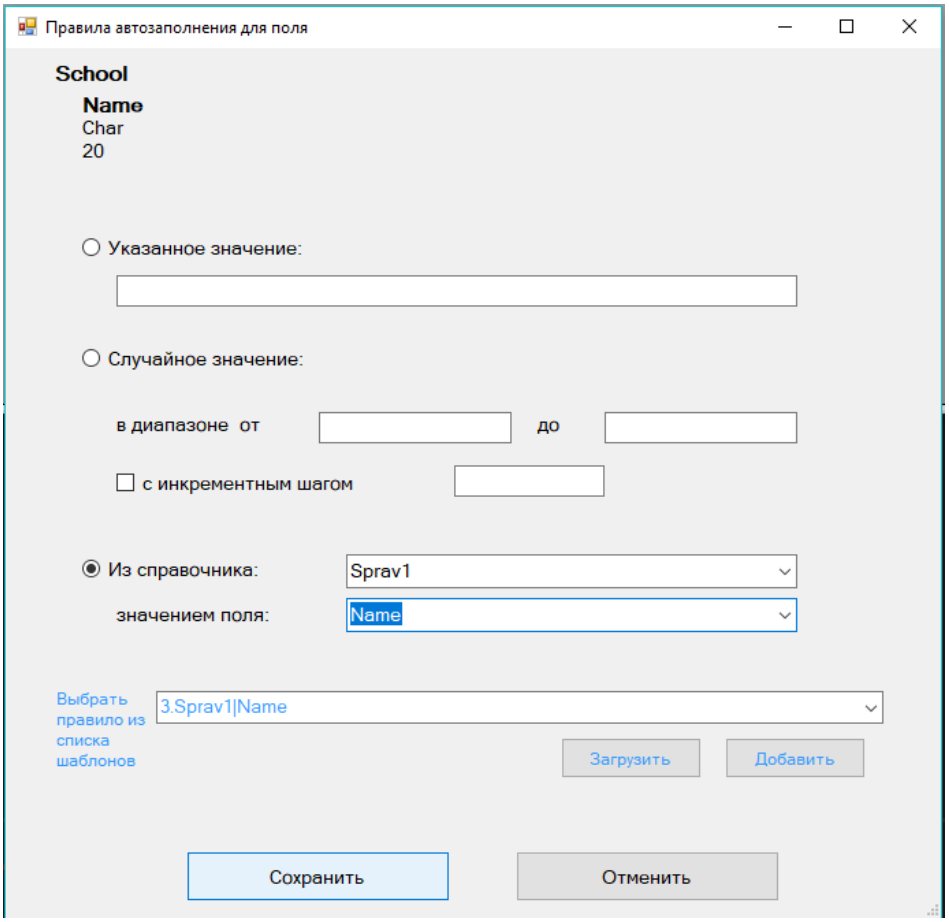


Рисунок 39 – Окно настройки правил для поля. Выбор шаблона

Управлять элементами в списке шаблонов можно на вкладке «Шаблоны». Вкладка открывается выбором подпункта меню «Правила», «Список шаблонов». Вкладка «Шаблоны» представлена на рисунке 40.

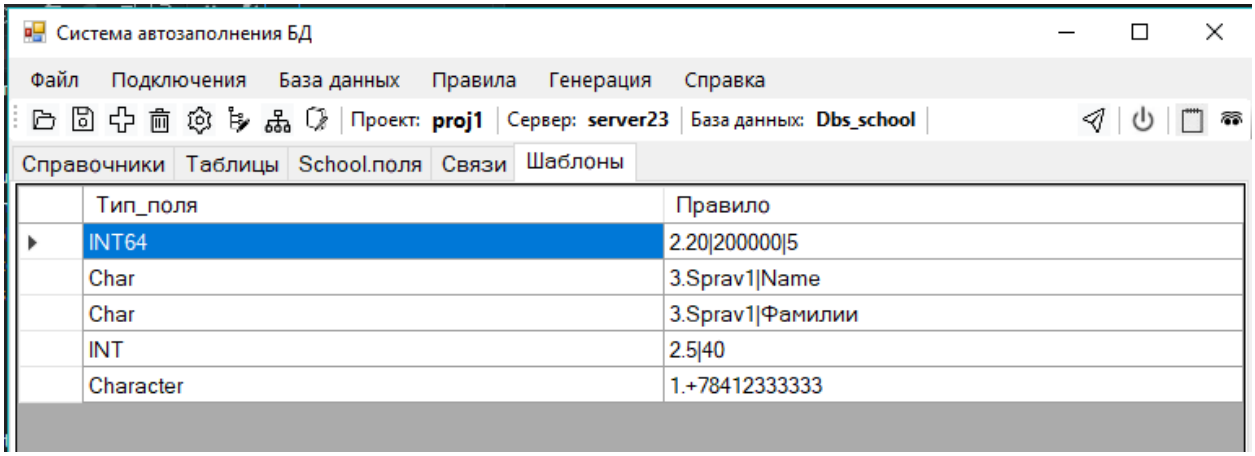


Рисунок 40 – Вкладка «Шаблоны»

После того как для каждой таблицы из списка настроены правила заполнения избранных полей, настроены связи, если они есть, и определено количество записываемых строк и приоритет можно запустить процесс генерации.

В случае, когда база данных не доступна в рабочее время для осуществления тестов автозаполнения данных, можно создать отложенный запуск в указанное время и с заданными дополнительными условиями. Окно настройки пуска с условиями представлено на рисунке 41.

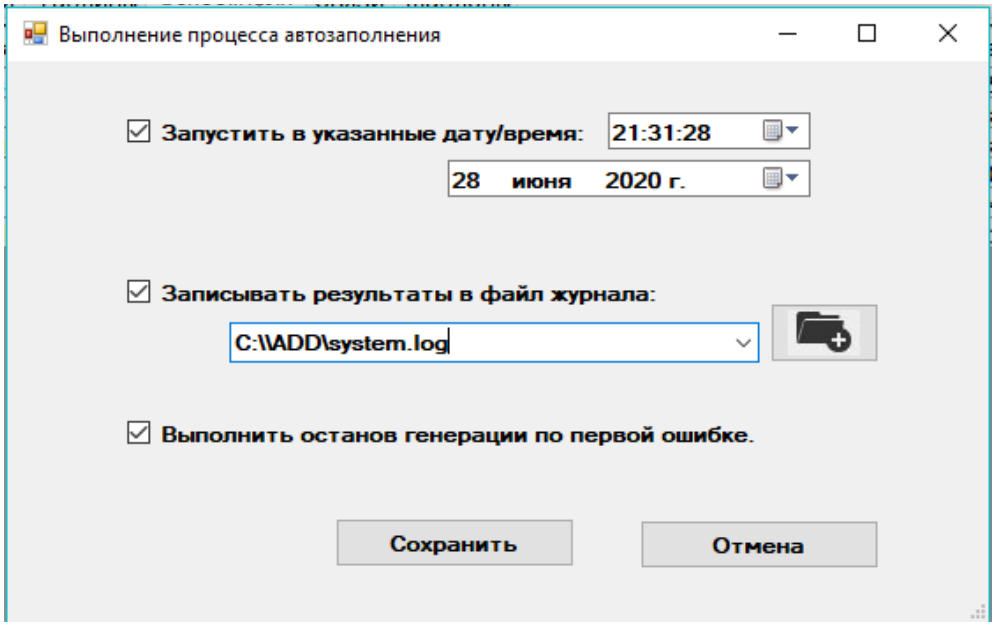


Рисунок 41 – Окно настройки запуска с условиями

Результаты запуска процесса автозаполнения можно увидеть в сформированном файле журнала генерации. А процесс автозаполнения можно будет отследить в разрабатываемом инструменте мониторинга.

4.3 Справочная информация

Для получения помощи в процессе настройки правил, подключений и процесса генерации создано окно информации «Просмотр справки», вызываемое из меню «Справка». Информационное окно программы изображено на рисунке 42.

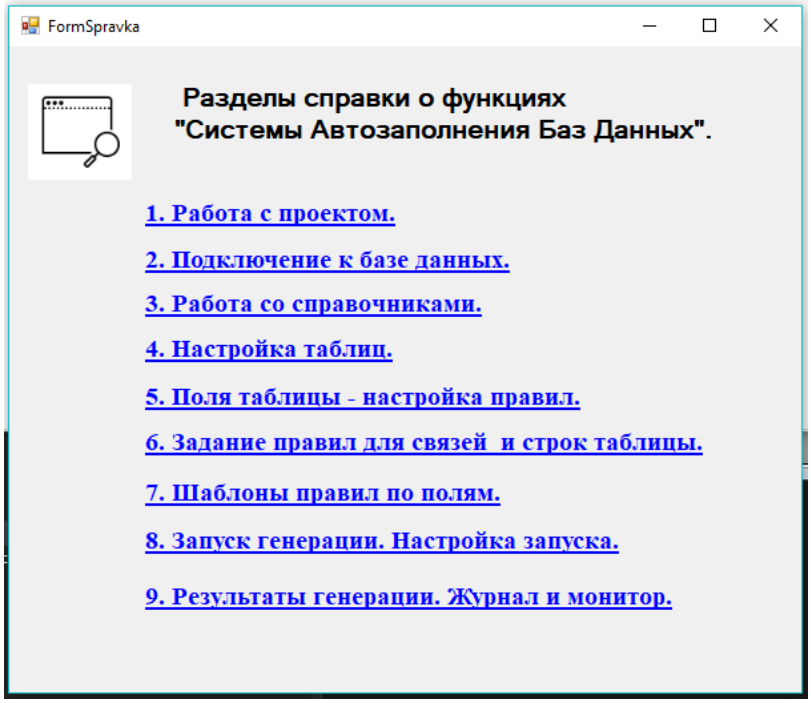


Рисунок 42 – Окно «Просмотр справки»

Узнать о разработке и о текущей версии можно выбрав пункт меню «О программе». Окно «О программе» представлено на рисунке 43.

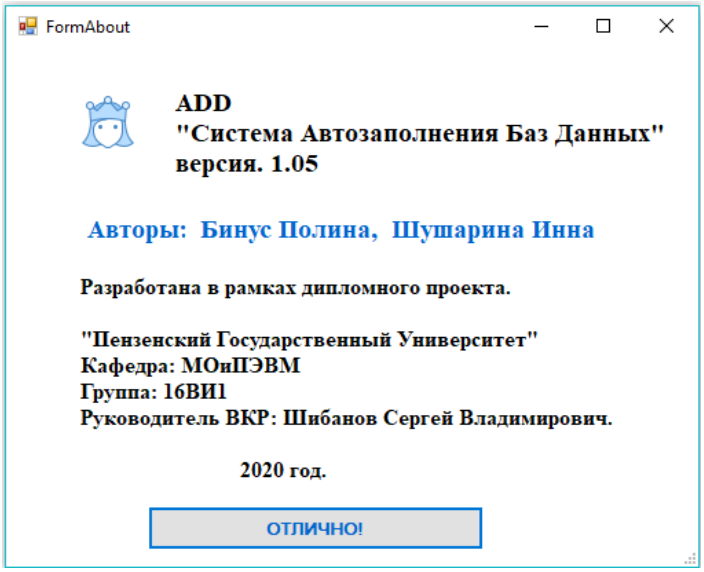


Рисунок 43 – Окно «О программе»

По мере совершенствования системы, справочная информация и руководство пользователя будут обновляться.

5 Тестирование приложения пользователя для системы автозаполнения баз данных

5.1 Выбор методологии тестирования

На сегодняшний день не существует таких методов тестирования программного обеспечения, которые позволяют однозначно и полностью выявить все ошибки и установить корректность функционирования анализируемой программы. Поэтому все методы тестирования являются формальными процессами проверки исследуемого или разрабатываемого программного обеспечения [26].

Тестирование понимается, как процесс, который применяется для обнаружения ошибок. Но для пользовательского приложения важно, чтобы во время тестирования было проверено не только поведение системы, но и интерфейс пользователя на эргономичность и удобство использования.

Все виды тестирования программного обеспечения можно условно разделить на три группы, в зависимости от преследуемых целей [27]:

– функциональные;
– нефункциональные;
– связанные с изменениями.

Функциональные тесты базируются на особенностях, функциях и взаимодействии с другими системами. Они представлены на всех уровнях тестирования: компонентном или модульном, интеграционном, системном и приемочном. Так же функциональные виды тестирования рассматривают внешнее поведение системы.

Нефункциональное тестирование описывает нужные для определения характеристик программного обеспечения тесты, которые измеряются различными величинами. Данный вид тестирования описывает как работает система.

Связанные с изменения тестирования проводится в качестве повторного уже после внесения необходимых изменений в программу.

Проводится для подтверждения, что найденная ранее проблема была действительно решена и с целью поиска ошибок, допущенных при исправлении предыдущих дефектов.

Для приложения пользователя для системы автозаполнения БД был выбран метод функционального тестирования.

5.2 Проведение тестирования приложения

Функциональное тестирование – процесс основанный на заранее известном или предполагаемом поведении пользователя, детальном анализе и изучении функциональной спецификации приложения. Данный вид тестирования проводится на основе документации к проекту и общения с заказчиком. Состоит на основе прецедентов, указанных в диаграмме вариантов использования (смотри рисунок 11). В таблице 10 представлено описание тест-кейсов.

Таблица 10 – Описание тестирования

Дымовое тестирование	Тестирование критического пути	Расширенное тестирование
1	2	3
Создание проекта	Создание через пункт меню. Создание через кнопку на панели задач.	Создание через пункт меню без открытого проекта. Создание через пункт меню с открытым проектом. Создание через кнопку на панели задач без открытого проекта. Создание через кнопку на панели задач с открытым проектом.
Выбор папки из окна выбора	Выбор папки при заданном изначально пути. Выбор папки при отсутствующем пути.	Выбор папки при заданном изначально корректном пути. Выбор папки при заданном изначально некорректном пути. Выбор папки при отсутствующем пути
Открытие формы подключения.	Открытие формы подключения через пункт меню и через кнопку на панели задач.	Открытие формы подключения через пункт меню и через кнопку на панели задач, при наличии открытого проекта и при его отсутствии, при наличии открытого подключения и без него, при открытых других окнах и без них.

Продолжение таблицы 10

1	2	3
Открытие подключения внутри формы «Подключения»	Выбор или ввод сервера. Выбор или ввод базы данных. Ввод логина. Ввод пароля. Ввод порта. Проверка подключения по кнопке. Проверка сохранения подключения.	Выбор или ввод сервера, корректные и некорректные значения. Выбор или ввод наименования базы данных, корректные и некорректные значения. Ввод логина, корректные и некорректные значения. Ввод пароля, корректные и некорректные значения. Ввод порта, корректные и некорректные значения. Проверка подключения по кнопке, корректные и некорректные значения. Проверка сохранения подключения, корректные и некорректные значения
Открытие вкладки Таблицы	Открытие вкладки Таблицы через пункт меню и через кнопку на панели задач.	Открытие вкладки Таблицы через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении.
Открытие вкладки Поля таблицы	Открытие вкладки Поля таблицы через пункт меню и через кнопку на панели задач.	Открытие вкладки Поля таблицы через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении, при не открытой вкладке Таблицы, при не выбранной таблице.
Открытие вкладки Связи	Открытие вкладки Связи через пункт меню и через кнопку на панели задач.	Открытие вкладки Связи через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении, при не открытой вкладке Таблицы, при не выбранной таблице.
Открытие вкладки Справочники	Открытие вкладки Справочники через пункт меню и через кнопку на панели задач.	Открытие вкладки Справочники через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении.
Открытие вкладки Шаблоны	Открытие вкладки Шаблоны через пункт меню и через кнопку на панели задач	Открытие вкладки Шаблоны через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении.

Продолжение таблицы 10

1	2	3
Открытие формы создания/добавления справочника	Открытие формы создания или добавления справочника через кнопку на панели задач	Открытие формы создания или добавления справочника через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении
Добавление нового Справочника	Заполнение полей формы создания или добавления справочника. Проверка подключения к справочнику. Проверка добавления в список справочников.	Заполнение полей формы создания или добавления справочника, корректными и некорректными данными. Проверка подключения к справочнику, при различных данных и вариантах подключения. Проверка добавления в список справочников, в случае корректных и некорректных данных.
Открытие формы добавления таблицы	Открытие формы добавления таблицы через кнопку на панели задач.	Открытие формы добавления таблицы через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении.
Подключение новой Таблицы	Заполнение полей формы подключения новой таблицы Проверка добавления в список таблиц.	Заполнение полей формы подключения новой таблицы корректными и некорректными данными. Проверка добавления в список таблиц, в случае корректных и некорректных данных.
Открытие формы добавления поля	Открытие формы добавления поля через кнопку на панели задач	Открытие формы добавления поля через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении
Подключение нового поля	Заполнение полей формы подключения нового поля. Проверка добавления в список полей.	Заполнение полей формы подключения нового поля корректными и некорректными данными. Проверка добавления в список полей, в случае корректных и некорректных данных.
Открытие формы добавления связи	Открытие формы добавления связи через кнопку на панели задач.	Открытие формы добавления связи через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении.
Подключение новой связи	Заполнение полей формы подключения новой связи Проверка добавления в список связей.	Заполнение полей формы подключения новой связи. Проверка добавления в список связей, в случае корректных и некорректных данных.

Продолжение таблицы 10

1	2	3
Проверка удаления поля	Удаление поля по кнопке на панели задач.	Удаление поля по кнопке на панели задач, при наличии других открытых вкладок и без них. Если не выбрана таблица. Проверка вопроса об удалении и исключения из списка полей.
Проверка удаления справочника	Удаление справочника по кнопке на панели задач.	Удаление справочника по кнопке на панели задач, при наличии других открытых вкладок и без них. Если не выбран справочник. Проверка вопроса об удалении и исключения из списка справочников.
Проверка удаления таблицы	Удаление таблицы по кнопке на панели задач.	Удаление таблицы по кнопке на панели задач, при наличии других открытых вкладок и без них. Если не выбрана таблица. Проверка вопроса об удалении и исключения из списка таблиц.
Проверка удаления связи	Удаление связи по кнопке на панели задач.	Удаление связи по кнопке на панели задач, при наличии других открытых вкладок и без них. Если не выбрана связь. Проверка вопроса об удалении и исключения из списка связей.
Проверка удаления шаблона	Удаление шаблона по кнопке на панели задач.	Удаление шаблона по кнопке на панели задач, при наличии других открытых вкладок и без них. Если не выбран шаблон. Проверка вопроса об удалении и исключения из списка шаблонов.
Проверка открытия формы Создания или редактирования правила	Проверка открытия формы Создания или Редактирования правила через пункт меню и через кнопку на панели задач	Проверка открытия формы Создания или Редактирования правила через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при не открытом подключении, при неоткрытой вкладке полей или не выбранном поле.
Проверка работы формы Создания/ Редактирования правила	Заполнение полей формы Создания/ Редактирования правила. Проверка сохранения результатов в правило по полю в списке.	Заполнение полей формы Создания или Редактирования правила автозаполнения поля" корректными и некорректными данными. Проверка автозагрузки списка справочников и списка шаблонов в форму. Проверка поведения полей в форме. Проверка сохранения шаблонов и выбора из списка шаблонов. Проверка сохранения результатов и действий по кнопке «Отменить».

Окончание таблицы 10

1	2	3
Проверка выполнения запуска генерации	Проверка выполнения запуска генерации через пункт меню и через кнопку на панели задач	Проверка выполнения запуска генерации через пункт меню и через кнопку на панели задач, при не открытом проекте, при не открытом подключении, при незаполненных правилах полей или не подключенной библиотеки генерации.
Проверка открытия формы Запуск с условиями.	Проверка открытия формы Запуск с условиями через пункт меню и через кнопку на панели задач	Проверка открытия формы Запуск с условиями через пункт меню и через кнопку на панели задач, при наличии уже открытых и не закрытых вкладок. При не открытом проекте, при незаполненных правилах полей или не подключенной библиотеки генерации.
Проверка работы формы Запуск с условиями.	Проверка работы формы Запуск с условиями. Проверка отложенного запуска и сохранение результатов.	Проверка работы формы Запуск с условиями. Проверка выбора даты и времени запуска, корректные и некорректные. Проверка отложенного запуска и сохранения результатов. Тоже при не открытом проекте, при незаполненных правилах полей или не подключенной библиотеки генерации.
Проверка открытия формы справки по разделам системы	Проверка открытия формы справки по разделам системы через пункт меню и через кнопку на панели задач	Проверка открытия формы справки по разделам системы через пункт меню и через кнопку на панели задач, при не открытом проекте, при не открытом подключении.
Проверка открытия формы «О программе»	Проверка открытия формы «О программе» через пункт меню	Проверка открытия «О программе» через пункт меню, при не открытом проекте, при не открытом подключении.

Результаты проведения данных тест-кейсов показали корректную и безошибочную работу программы.

5.3 Метрики Хостледа

Метрикой программного обеспечения называется мера, которая служит для получения численного значения некоторого свойства программного обеспечения или его спецификаций. В данной работе расчет метрик приведен по метрикам Холстеда для метода получения правила для поля, представленного на рисунке 44.

```
private void SetRule(string s0)
{
    String s1 = s0.Substring(0, 1);
    int n1;
    if (s1.Equals("1"))
    {
        radioButton1.Checked = true;
        textBox1.Text = s0.Substring(2);
    }
    else if (s1.Equals("2"))
    {
        radioButton2.Checked = true;
        n1 = s0.IndexOf("|");
        textBox2.Text = s0.Substring(2, n1 - 2);
        s1 = s0.Substring(n1+1);
        n1 = s1.IndexOf("|");
        if (n1 < 0)
        {
            textBox3.Text = s1;
            checkBox1.Checked = false;
        }
        else
        {
            textBox3.Text = s1.Substring(0, n1);
            checkBox1.Checked = true;
            textBox4.Text = s1.Substring(n1 + 1);
        }
    }
    else if (s1.Equals("3"))
    {
        radioButton3.Checked = true;
        n1 = s0.IndexOf("|");
        comboBox1.Text = s0.Substring(2, n1 - 2);
        comboBox2.Text = s0.Substring(n1 + 1);
    }
}
```

Рисунок 44 – метод получения правила для поля

Для проведения расчетов метрики необходимо выяснить значение таких параметров, как количество уникальных операторов, количество уникальных операндов, общее количество операторов, общее количество операндов. Полученные результаты находятся в таблице 11.

Таблица 11 – Операторы и операнды

Операторы	Количество	Операнды	Количество
if ( )	4	s1	9
=	17	s0	8
+	3	n1	11
<	1		
-	2		
n1 = 5	N1 = 27	n2 = 3	N2 = 28

На основе полученных показателей определяем словарь программы по формуле (1):

$$n = n1 + n2 = 8, \quad (1)$$

Также определяем длину программы по формуле (2):

$$N = N1 + N2 = 55. \quad (2)$$

При наличии этих значений можно рассчитать объем кода (3) и уровень качества программирования (4):

$$V = N \times \log2\, n, \quad (3)$$

$$L' = \frac{(2 \times n2)}{(n1 \times N2)}. \quad (4)$$

После подстановки значений в формулы, получаем следующие значения: V = 165; L′ = 0,043. Теперь посчитаем сложность качества программы, используя вычисленные значения объема кода и уровня качества программирования (5):

$$Ec = \frac{V}{L'} \times 2. \quad (5)$$

В результате получаем: E_c = 7674,42 . Данный результат означает, что разработанный код программы не является компактным и простым для понимания. Однако это объясняется тем, что в коде присутствуют множество операндов.

Заключение

В процессе выполнения выпускной квалификационной работы был выполнен анализ предметной области, в рамках которого было рассмотрена актуальность создания систем для автозаполнения баз данных, также был проведен обзор существующих программных средств, специализирующихся на автоматическом заполнении баз данных, была поставлена задача на разработку и определен бюджет разработки. В рамках проектирования приложения пользователя была проведена разработка требований к создаваемой системе автозаполнения баз данных в целом и приложения пользователя в частности. Разработка требований включала в себя определение требований к архитектуре, технологиям разработки и используемым программным средствам.

В результате было разработано приложение пользователя для прототипа системы автозаполнения баз данных.

Список использованных источников

1. Бинус, П.И. Сравнительный анализ средств автоматического заполнения баз данных / П.И. Бинус, И.С. Шушарина // НИТиС: материалы XVI Международной научно-технической конференции. – Пенза: Изд-во ПГУ, 2019. – С. 190 – 193.

2. Бинус, П.И. Функциональные возможности системы автоматического заполнения баз данных / П.И. Бинус, И.С. Шушарина // Информационные технологии в науке и образовании. Проблемы и перспективы: материалы VII ежегодной всероссийской межвузовской научно-практической конференции. – Пенза: Изд-во ПГУ, 2020. – С. 123 – 125.

3. Справочник24 – Жизненный цикл БД [Электронный ресурс]. – URL: https://spravochnick.ru/bazy_dannyh/zhiznennyy_cikl_bd/ (дата обращения 28.04.2020).

4. Демаков, А.В. Генерация тестовых данных сложной структуры с учетом контекстных ограничений / А.В. Демаков, С.В. Зеленов, С.А. Зеленова // Труды института системного программирования РАН. – 2006. – Т. 9. С. 83-96.

5. Костычев, Е.А. Нацеленная генерация данных для тестирования приложений над базами данных / Е.А. Костычев, В.А. Омельченко, С.В. Зеленов // Труды института системного программирования РАН. – 2011. – Т. 20. С. 253-268.

6. Демаков, А.В. Генератор сложных данных Pinery: реализация новых возможностей UniTESK / А.В. Демаков, С.В. Зеленов, С.А. Зеленова // Труды института системного программирования РАН. – 2008. – Т. 14. № 1. С. 119-136.

7. ООО Тестматик – Утилиты для генерации тестовых данных [Электронный ресурс]. – URL: https://testmatick.com/ru/utility-dlya-generatsii-testovyh-dannyh/ (дата обращения 18.03.2020).

8. ООО «EMC Софтваре Девелопмент» – EMS DATA GENERATOR [Электронный ресурс]. – URL: https://www.sqlmanager.net/ru/products/datagenerator (дата обращения 18.03.2020).

9. DTM soft - DTM Data Generator [Электронный ресурс]. – URL: http://www.sqledit.com/dg/ (дата обращения 18.03.2020).

10. DBeaver Community – Universal Database Tool [Электронный ресурс]. – URL: https://dbeaver.io/ (дата обращения 18.03.2020).

11. We Manage IT – Red Gate SQL Data Generator [Электронный ресурс]. – URL: https://www.sigirgroup.ru/programmirovanie/red-gate-sql-data-generator.html (дата обращения 18.03.2020).

12. Devart – DBFORGE STUDIO [Электронный ресурс]. – URL: https://www.devart.com/ru/dbforge/sql/studio/features.html (дата обращения 18.03.2020).

13. Microsoft User Group Community – Создание Plugin архитектуры с помощью C# [Электронный ресурс]. – URL: http://msugvnua000.web710.discountasp.net/Posts/Details/3477 (дата обращения 21.04.2020).

14. Wrike, Inc. – Что такое программный интерфейс приложений (API) [Электронный ресурс]. – URL: https://www.wrike.com/ru/blog/chto-takoe-programmnyj-interfejs-prilozhenij-api/ (дата обращения 30.04.2020).

15. Gartner, Inc. – Magic Quadrant for Operational Database Management Systems [Электронный ресурс]. – URL: https://www.gartner.com/doc/reprints?id=1-1XT9MEFJ&ct=191125&st=sb (дата обращения 02.05.2020).

16. ITVDN – C# против Java: какой язык программирования общего назначения выбрать? [Электронный ресурс]. – URL: https://itvdn.com/ru/blog/article/csharp_vs_java (дата обращения 10.05.2020).

17. Забудский, Е.И. Учебно-методический комплекс дисциплины «Объектно-ориентированный анализ и программирование». М.: Кафедра АПС ГУ-ВШЭ, 2008.

18. MSNET.NAROD.RU. – Сравнение C#, Java и C++? [Электронный ресурс]. – URL: http://msnet.narod.ru/art/art_001/art_001.htm (дата обращения 11.05.2020)

19. Microsoft Corporation – Visual Studio IDE [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения 12.05.2020)

20. METANIT – Язык C# и платформа .NET Core [Электронный ресурс]. – URL: https://metanit.com/sharp/tutorial/1.1.php (дата обращения 12.05.2020)

21. Projectimo – Место диаграммы Ганта в планировании проекта [Электронный ресурс]. – URL: http://projectimo.ru/planirovanie-proekta/diagramma-ganta.html (дата обращения 20.05.2020)

22. Microsoft Corporation – Выбор подходящего отчета в Project [Электронный ресурс]. – URL: https://cutt.ly/IiJ6QJt (дата обращения 20.05.2020)

23. Архитектура «клиент-сервер» [Электронный ресурс]. – URL: http://www.mstu.edu.ru/study/materials/zelenkov/ch_7_1.html (дата обращения 03.05.2020)

24. Flexberry – Диаграмма последовательности [Электронный ресурс]. – URL: https://flexberry.github.io/ru/fd_sequence-diagram.html (дата обращения 03.06.2020)

25. Бодягин, И. Model-View-Controller в .Net / Бодягин И. // RSDN Magazine РОССИЯ – 2006, – #2 (64), с.28-22

26. OUTCOLDMAN – Паттерны: MVC, MVP и MVVM [Электронный ресурс]. – URL: https://cutt.ly/OiZaBot (дата обращения 05.06.2020)

27. ООО «Аплана» – Функциональное тестирование [Электронный ресурс]. – URL: https://www.appline.ru/services/testing/functionalnoe-testirovanie (дата обращения 08.06)

28. ProTesting.RU – Виды тестирования программного обеспечения [Электронный ресурс]. – URL: http://www.protesting.ru/testing/testtypes.html (дата обращения 08.06)

Приложение А.
Глоссарий

(справочное)

База данных (БД) – представленная в объективной форме совокупность самостоятельных материалов.

Система управления базами данных (СУБД) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными (и совместно использовать многими пользователями).

Unified Modeling Language (UML) – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Информационная система (ИС) – система, предназначенная для хранения, поиска и обработки информации.

Жизненный цикл информационной системы (ЖЦИС) – это непрерывный процесс, начинающийся с момента принятия решения о создании информационной системы и заканчивающийся в момент полного изъятия ее из эксплуатации.

Жизненный цикл базы данных (ЖЦБД) – это процесс проектирования, реализации и поддержки базы данных.

Comma-Separated Values (CSV) – текстовый формат, предназначенный для представления табличных данных.

Structured Query Language (SQL) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Dynamic Link Library (DLL) – это библиотека, содержащая код и данные, которые могут использоваться одновременно более чем одной программой.

Программные средства (ПС) – это совокупность программ, которые и заставляют аппаратную часть системы выполнять необходимые действия

Model View Controller (MVC) – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер.

Model-View-Presenter (MVP) – шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса.

Java Virtual Machine (JVM) – основная часть исполняющей системы Java.

Common Language Runtime (CLR, общеязыковая исполняющая среда) – исполняющая среда для байт-кода, в которой компилируются программы, написанные на .NET-совместимых языках программирования. CLR является одним из основных компонентов пакета Microsoft .NET Framework.

Auto Data Download (ADD) – Система автоматического заполнения баз данных.

Приложение Б.

Листинг программы

(рекомендуемое)

Листинг Б.1 – Код файла Program.cs

```
static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
    public class Diction
    {  // СПРАВОЧНИК
        public string Имя_Справочника { get; set; }
        public string Сервер { get; set; }
        public string База_Данных { get; set; }
        public string Таблица { get; set; }
        public Diction( string DN, string DS, string DB, string DT)
        {
            Имя_Справочника = DN;
            Сервер = DS;
            База_Данных = DB;
            Таблица = DT;
        }
        public Diction()
        {
            Имя_Справочника = "";
            Сервер = "";
            База_Данных = "";
            Таблица = "";
        }
    }
    public class MLinks
    {  // СВЯЗИ
        public string Таблица_П{ get; set; }
        public string Таблица_Внешняя { get; set; }
        public string Поле_связи { get; set; }
        public string Строк { get; set; }
        public MLinks(string DN, string DS, string DB, string DT)
        {
            Таблица_П = DN;
            Таблица_Внешняя = DS;
            Поле_связи = DB;
            Строк = DT;
        }
    }
    public class MTable
    {  // Таблицы
        public string Таблица { get; set; }
        public string Приоритет { get; set; }
        public string Связи { get; set; }
        public string Правила { get; set; }
        public MTable(string DN, string DS, string DB, string DT)
        {
            Таблица = DN;
            Приоритет = DS;
            Связи = DB;
            Правила = DT;
        }
    }
    public class MRule
```

```
    {  // Поля таблицы
        public string Название { get; set; }
        public string Тип { get; set; }
        public string Размер { get; set; }
        public string Индекс { get; set; }
        public string Правило { get; set; }
        public MRule(string DN, string DS, string DB, string DI, string DT)
        {
            Название = DN;
            Тип = DS;
            Размер = DB;
            Индекс = DI;
            Правило = DT;
        }
    }
    public class MShabl
    {  // Поля шаблона
        public string Тип_поля { get; set; }
        public string Правило { get; set; }
        public MShabl(string DN, string DS)
        {
            Тип_поля = DN;
            Правило = DS;
        }
    }
```

Листинг Б.2 – Код файла MainForm.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ADD
{
    public partial class MainForm : Form
    {
        public string mProjName;
        public string mProjDir;
        public string mProjServer;
        public string mProjDB;
        public static List<Diction> listDics;
        public static List<MLinks>  listMLinks;
        public static List<MTable>  listMTables;
        public static List<MRule>   listMRules;
        public static List<MShabl>  listMShabls;
        public MainForm()
        {
            InitializeComponent();
            listDics = new List<Diction>();
            listMLinks = new List<MLinks>();
            listMTables = new List<MTable>();
            listMRules = new List<MRule>();
            listMShabls = new List<MShabl>();
            tabControl1.Controls.Clear();
            openFileDialog1.Filter = "Text files(*.txt)|*.txt" +
                                     "|Json files(*.json)|*.json" +
                                     "|All files(*.*)|*.*";
        }
        private void новыйПроектToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //добавить новый ПРОЕКТ  ***проект
            FormNewPrj fNewP = new FormNewPrj();
            //fNewP.MdiParent = this;
```

```
            fNewP.ShowDialog();
            if (fNewP.pname != "")
            {
                mProjName = fNewP.pname;
                mProjDir  = fNewP.ppath;
                toolStripLabel1.Text = mProjName;
             }
            fNewP.Close();
        }
        private void добавитьToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            //добавить подключение
            Connect fConnect = new Connect();
            fConnect.ShowDialog();
            if (fConnect.server != "")
            {
                mProjServer = fConnect.server;
                mProjDB = fConnect.database;
                toolStripLabel4.Text = mProjServer;
                toolStripLabel6.Text = mProjDB;
            }
            fConnect.Close();
        }

        private void списокТаблицToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            // Здесь заполняем список Таблиц -  ПОЛУЧИТЬ из метода Инны
            MTable spr1 = new MTable("School", "1", "0-3", "Добавить 1");
            MTable spr2 = new MTable("Адрес", "2", "1-1", "Добавить 20-50");
            MTable spr3 = new MTable("Work", "3", "0-1", "Добавить 10");
            listMTables.Add(spr1);
            listMTables.Add(spr2);
            listMTables.Add(spr3);
            dataGridView4.AutoGenerateColumns = true;
            dataGridView4.DataSource = listMTables;
            dataGridView4.AllowUserToAddRows = true;
            if (!tabControl1.Controls.Contains(this.tabPage2))
            {
                tabControl1.Controls.Add(this.tabPage2);
                tabControl1.Visible = true;
            }
            tabControl1.SelectTab(tabPage2);
            dataGridView4.Show();
        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            DialogResult dial= MessageBox.Show("Вы точно хотите выйти?",
"Внимание", MessageBoxButtons.YesNo);
            if (dial == DialogResult.Yes)  {Close(); }

        }

         private void правилаПоПолямToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            // Список полей для выбранной таблицы
            if (tabControl1.TabCount > 0)
            {
                if (tabControl1.SelectedTab.Text.Equals("Таблицы"))
                {
                  // ОТКРЫТЬ СПИСОК ПОЛЕЙ
                  MTable tt = (MTable)dataGridView4.CurrentRow.DataBoundItem;
                  MRule spr;
```

```
                    tabPage3.Text = tt.Таблица + ".поля";
                    spr = new MRule("Indeks", "INT64", "8", "PK",
"2.20|200000|5");
                    listMRules.Add(spr);
                    spr = new MRule("Name", "Char", "20", "",
"3.Sprav1|Name");
                    listMRules.Add(spr);
                    spr = new MRule("Subname", "Char", "25", "",
"3.Sprav1|Фамилии");
                    listMRules.Add(spr);
                    spr = new MRule("Age", "INT", "3", "PK", "2.5|40");
                    listMRules.Add(spr);
                    spr = new MRule("Phone", "Char", "15", "PK",
"1.+78412333333");
                    listMRules.Add(spr);
                    dataGridView5.AutoGenerateColumns = true;
                    dataGridView5.DataSource = listMRules;
                    dataGridView5.AllowUserToAddRows = true;
                    if (!tabControl1.Controls.Contains(this.tabPage3))
                    {
                        tabControl1.Controls.Add(this.tabPage3);
                        tabControl1.Visible = true;
                    }
                    tabControl1.SelectTab(tabPage3);
                    dataGridView5.Show();
                    return;
                }
            }
  MessageBox.Show("Внимание! Не выбрана таблица для открытия списка полей!");
        }
        private void закрытьВкладкуToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            tabControl1.Controls.Remove(tabControl1.SelectedTab);
        }
        private void toolStripButton6_Click(object sender, EventArgs e)
        {
            // Список полей для выбранной таблицы
            if (tabControl1.TabCount > 0)
            {
                string ss = tabControl1.SelectedTab.Text;
                string srs = ss.Substring(ss.IndexOf(".")+1, 4);
                if (srs.Equals("поля"))
                {

                    MRule tt = (MRule)dataGridView5.CurrentRow.DataBoundItem;
                    RuleAtributeForm rf = new RuleAtributeForm();
                    rf.SetPolya(tt, ss.Substring(0, ss.IndexOf(".")),
listMShabls);
                    foreach (Diction dd in listDics)
                    {
                        rf.SetDict(dd);
                    }
                    rf.ShowDialog();
                    if (rf.frFlag)
                    {// вернуть данные из формы
                      tt.Правило = rf.RuleToString();
                      rf.SetShabl(listMShabls);//добавить добавленные шаблоны
                        dataGridView5.DataSource = null;
                        dataGridView5.Update();
                        dataGridView5.DataSource = listMRules;
                        dataGridView5.Refresh();
                    }
                    rf.Close();
                    return;
                }
```

```
            }
   MessageBox.Show("Внимание! Не выбрано поле для открытия правила!");
        }
        private void открытьПроектToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var t = openFileDialog1.InitialDirectory;
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                t = openFileDialog1.FileName;
            }
        }

        private void списокToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            //var listDics = new List<Diction>();
            Diction spr1 = new
Diction("ФИО","Local","C:\\\\MySpav\\","FIO.XLS");
            Diction spr2 = new Diction("Адрес","Local","SQL","Address");
            Diction spr3 = new
Diction("Школа","192.168.10.1","TestSQL","School");
            listDics.Add(spr1);
            listDics.Add(spr2);
            listDics.Add(spr3);
            dataGridView1.AutoGenerateColumns = true;
            dataGridView1.DataSource = listDics;
            dataGridView1.AllowUserToAddRows = true;
            if (!tabControl1.Controls.Contains(this.tabPage1))
            {
                tabControl1.Controls.Add(this.tabPage1);
                tabControl1.Visible = true;
            }
            tabControl1.SelectTab(tabPage1);
            dataGridView1.Show();
        }
        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            if (tabControl1.TabCount < 1)
            {  // изменяем основное подключение
                добавитьToolStripMenuItem_Click(sender, e);
                return;
            }
            string ss = tabControl1.SelectedTab.Text;
            string srs = ss.Substring(ss.IndexOf(".") + 1, 4);
            // обработка кнопки ДОБАВИТЬ
            if (ss.Equals("Справочники"))
            {
                //добавить СПРАВОЧНИК
                FormSprav fCon = new FormSprav();
                fCon.ShowDialog();
                if (fCon.flgSave)
                {
                    listDics.Add(fCon.MDRess());
                    dataGridView1.DataSource = null;
                    dataGridView1.Update();
                    dataGridView1.DataSource = listDics;
                    dataGridView1.Refresh();
                }
                fCon.Close();
            }
            if (ss.Equals("Связи"))
            {
                //добавить СВЯЗЬ
                FormLinks fCon = new FormLinks();
                fCon.ShowDialog();
```

```
                if (fCon.flgLSave)
                {
                    listMLinks.Add(fCon.MLRess());
                    dataGridView2.DataSource = null;
                    dataGridView2.Update();
                    dataGridView2.DataSource = listMLinks;
                    dataGridView2.Refresh();
                }
                fCon.Close();
            }
            if (ss.Equals("Таблицы"))
            {
                //добавить Таблицу
                FormTable fCon = new FormTable();
                fCon.ShowDialog();
                if (fCon.flTSave)
                {
                    listMTables.Add(fCon.MTRess());
                    dataGridView4.DataSource = null;
                    dataGridView4.Update();
                    dataGridView4.DataSource = listMTables;
                    dataGridView4.Refresh();
                }
                fCon.Close();
            }
            if (srs.Equals("поля"))
            {
                //добавить поле в список полей
                FormField fCon = new FormField();
                fCon.MFNew(ss.Substring(0, ss.IndexOf(".")));  // передадим
название таблицы
                fCon.ShowDialog();
                if (fCon.ffSave == 1)
                {
                    MRule mr = fCon.MFRess();
                    foreach (MRule mmm in listMRules)
                    {
                        if (mr.Название == mmm.Название)
                        {
                            MessageBox.Show("Добавляемое поле <" +
mr.Название + "> уже есть в списке полей!");
                            return;
                        }
                    }
                    listMRules.Add(fCon.MFRess());
                    dataGridView5.DataSource = null;
                    dataGridView5.Update();
                    dataGridView5.DataSource = listMRules;
                    dataGridView5.Refresh();
                }
                fCon.Close();

                if (fCon.ffSave == -1)
                {
                    MessageBox.Show("Для удаления поля используйте
специальный режим!");
                }
            }
        }
        private void toolStripButtDel_Click(object sender, EventArgs e)
        {
            if (tabControl1.TabCount < 1)
            {  // изменяем основное подключение
                добавитьToolStripMenuItem_Click(sender, e);
                return;
            }
```

```
            string ss = tabControl1.SelectedTab.Text;
            // Удалить элемент
            if (ss.Equals("Справочники"))
            {
                //удалить СПРАВОЧНИК
                Diction tt = (Diction)
dataGridView1.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                   DialogResult nn = MessageBox.Show("Вы подтверждаете
удаление справочника: " +
                      tt.Имя_Справочника, "Внимание",
MessageBoxButtons.OKCancel);
                    if ( nn == DialogResult.OK)
                     {
                         listDics.Remove(tt);
                         dataGridView1.DataSource = null;
                         dataGridView1.Update();
                         dataGridView1.DataSource = listDics;
                         dataGridView1.Refresh();
                     }
                }
                return;
            }
            if (ss.Equals("Шаблоны"))
            {
                //удалить Шаблон
                MShabl tt = (MShabl)dataGridView3.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    DialogResult nn = MessageBox.Show("Вы подтверждаете
удаление шаблона: " +
                         tt.Тип_поля+" "+tt.Правило, "Внимание",
MessageBoxButtons.OKCancel);
                    if (nn == DialogResult.OK)
                    {
                         listMShabls.Remove(tt);
                         dataGridView3.DataSource = null;
                         dataGridView3.Update();
                         dataGridView3.DataSource = listMShabls;
                         dataGridView3.Refresh();
                    }
                }
                return;
            }
            if (ss.Equals("Связи"))
            {
                //удалить СВЯЗЬ
                MLinks tt = (MLinks)dataGridView2.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    DialogResult nn = MessageBox.Show("Вы подтверждаете
удаление связи: " +
                         tt.Таблица_П + " - " + tt.Таблица_Внешняя,
"Внимание", MessageBoxButtons.OKCancel);
                    if (nn == DialogResult.OK)
                    {
                         listMLinks.Remove(tt);
                         dataGridView2.DataSource = null;
                         dataGridView2.Update();
                         dataGridView2.DataSource = listMLinks;
                         dataGridView2.Refresh();
                    }
                }
                return;
            }
```

```
            if (ss.Equals("Таблицы"))
            {
                //удалить Таблицу
                MTable tt = (MTable)dataGridView4.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    DialogResult nn = MessageBox.Show("Вы подтверждаете
удаление таблицы из списка: " +
                        tt.Таблица, "Внимание", MessageBoxButtons.OKCancel);
                    if (nn == DialogResult.OK)
                    {
                        listMTables.Remove(tt);
                        dataGridView4.DataSource = null;
                        dataGridView4.Update();
                        dataGridView4.DataSource = listMTables;
                        dataGridView4.Refresh();
                    }
                }
                return;
            }
            string srs = ss.Substring(ss.IndexOf(".") + 1, 4);
            if (srs.Equals("поля"))
            {
                //удалить поле из списка
                MRule tt = (MRule)dataGridView5.CurrentRow.DataBoundItem;
                if (tt != null)
                {   FormField fCon = new FormField();
                    fCon.MFGet(tt, ss.Substring(0, ss.IndexOf(".")));  //
передадим название таблицы
                    fCon.ShowDialog();
                    if (fCon.ffSave == -1)
                    {
                        MRule mr = fCon.MFRess();
                        foreach (MRule mmm in listMRules)
                        {
                            if (mr.Название == mmm.Название)
                            {
                                listMRules.Remove(mmm);
                                dataGridView5.DataSource = null;
                                dataGridView5.Update();
                                dataGridView5.DataSource = listMRules;
                                dataGridView5.Refresh();
                                fCon.Close();
                                return;
                            }
                        }
                        MessageBox.Show("Удаляемое поле <" + mr.Название + ">
отсутствует в списке полей!");
                    }
                    if (fCon.ffSave == 1)
                    {
                        MessageBox.Show("Для добавления поля используйте
специальный режим!");
                    }
                   fCon.Close();
                }
                return;
            }
        }
        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            if (tabControl1.TabCount < 1)
            {  // изменяем основное подключение
                добавитьToolStripMenuItem_Click(sender, e);
                return;
            }
```

```
            // ИЗМЕНИТЬ элемент
            if (tabControl1.SelectedTab.Text.Equals("Справочники"))
            {
                //изменить СПРАВОЧНИК
                Diction tt = (Diction)dataGridView1.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    FormSprav fCon = new FormSprav();
                    fCon.MDGet(tt);
                    fCon.ShowDialog();
                    if (fCon.flgSave)
                    {
                        fCon.MDChange(tt);
                        dataGridView1.DataSource = null;
                        dataGridView1.Update();
                        dataGridView1.DataSource = listDics;
                        dataGridView1.Refresh();
                    }
                    fCon.Close();
                }
            }
            if (tabControl1.SelectedTab.Text.Equals("Связи"))
            {
                //изменить СВЯЗЬ
                MLinks tt = (MLinks)dataGridView2.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    FormLinks fCon = new FormLinks();
                    fCon.MLGet(tt);
                    fCon.ShowDialog();
                    if (fCon.flgLSave)
                    {
                        fCon.MLChange(tt);
                        dataGridView2.DataSource = null;
                        dataGridView2.Update();
                        dataGridView2.DataSource = listMLinks;
                        dataGridView2.Refresh();
                    }
                    fCon.Close();
                }
            }
            if (tabControl1.SelectedTab.Text.Equals("Таблицы"))
            {
                //изменить Свойства Таблицы
                MTable tt = (MTable)dataGridView4.CurrentRow.DataBoundItem;
                if (tt != null)
                {
                    FormTable fCon = new FormTable();
                    fCon.MTGet(tt);
                    fCon.ShowDialog();
                    if (fCon.flTSave)
                    {
                        fCon.MTChange(tt);
                        dataGridView4.DataSource = null;
                        dataGridView4.Update();
                        dataGridView4.DataSource = listMTables;
                        dataGridView4.Refresh();
                    }
                    fCon.Close();
                }
            }
        }
        private void оПрограммеЗначитToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //добавить подключение
```

```
            FormAbout fCon = new FormAbout();
            fCon.ShowDialog();
        }
        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            if (tabControl1.TabCount < 1)
            {
                MessageBox.Show("Внимание! Не выбрана таблица для открытия
списка полей!");
                return;
            }
            if (tabControl1.SelectedTab.Text.Equals("Таблицы"))
            { // ОТКРЫТЬ СПИСОК ПОЛЕЙ
                MTable tt = (MTable)dataGridView4.CurrentRow.DataBoundItem;
                правилаПоПолямToolStripMenuItem_Click(sender, e);
                tabPage3.Text = tt.Таблица + ".поля";
            }
            else  MessageBox.Show("Внимание! Не выбрана таблица для открытия
списка полей!");
        }

        private void описаниеЗначитToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //СПРАВКА
            FormSpravka fCon = new FormSpravka();
            fCon.Show();

        }

        private void списокСвязейToolStripMenuItem_Click(object sender,
EventArgs e)
        {

            MLinks spr1 = new MLinks("Школа", "Сотрудники", "School_ID",
"100");
            MLinks spr2 = new MLinks("Классы", "Ученики", "Class_id",
"20,30");
            MLinks spr3 = new MLinks("Кабинеты", "Классы", "Kab_num", "4,8");
            listMLinks.Add(spr1);
            listMLinks.Add(spr2);
            listMLinks.Add(spr3);
            dataGridView2.AutoGenerateColumns = true;
            dataGridView2.DataSource = listMLinks;
            dataGridView2.AllowUserToAddRows = true;
            if (!tabControl1.Controls.Contains(this.tabPage4))
            {
                tabControl1.Controls.Add(this.tabPage4);
                tabControl1.Visible = true;
            }
            tabControl1.SelectTab(tabPage4);
            dataGridView2.Show();
        }
        private void toolStripMenuItem1_Click(object sender, EventArgs e)
        {  //список полей
            toolStripButton4_Click(sender, e);
        }
        private void шаблоныПравилToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //шаблоны правил
            MShabl spr;
            spr = new MShabl("INT64", "2.20|200000|5");
            listMShabls.Add(spr);
            spr = new MShabl("Char", "3.Sprav1|Name");
            listMShabls.Add(spr);
```

```
            spr = new MShabl("Char", "3.Sprav1|Фамилии");
            listMShabls.Add(spr);
            spr = new MShabl("INT",  "2.5|40");
            listMShabls.Add(spr);
            spr = new MShabl("Character",  "1.+78412333333");
            listMShabls.Add(spr);
            dataGridView3.AutoGenerateColumns = true;
            dataGridView3.DataSource = listMShabls;
            dataGridView3.AllowUserToAddRows = true;
            if (!tabControl1.Controls.Contains(this.tabPage5))
            {
                tabControl1.Controls.Add(this.tabPage5);
                tabControl1.Visible = true;
            }
            tabControl1.SelectTab(tabPage5);
            dataGridView3.Show();
        }
        private void пускСУсловиямиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            FormPusk2 fp2 = new FormPusk2();
            fp2.ShowDialog();
        }
        private void сохранитьПроектToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            MessageBox.Show("Проект сохранен!\n"+"Путь: D://1.txt",
"Внимание");
        }
    }
}
```

Листинг Б.3 – Код файла RuleAtributeForm.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ADD
{
    public partial class RuleAtributeForm : Form
    {
        public bool frFlag;
        public RuleAtributeForm()
        {
            InitializeComponent();
            frFlag = false;
        }
        public void SetPolya(MRule tt, String tname, List<MShabl> lMs)
        {
            label1.Text = tt.Название;
            label2.Text = tname;
            label6.Text = tt.Тип;
            label7.Text = tt.Размер;
            label8.Text = tt.Индекс;
            SetRule(tt.Правило);   // загрузим правило
                                   // загрузим шаблоны для этого типа поля
            foreach (MShabl mms in lMs)
            {
                if (mms.Тип_поля == tt.Тип)
                {
                    comboBox3.Items.Add(mms.Правило);
                }
            }
```

```
}

private void SetRule(string s0)
    {
    String s1 = s0.Substring(0, 1);
    int n1;
    if (s1.Equals("1"))
    {
        radioButton1.Checked = true;
        textBox1.Text = s0.Substring(2);
    }
    else if (s1.Equals("2"))
    {
        radioButton2.Checked = true;
        n1 = s0.IndexOf("|");
        textBox2.Text = s0.Substring(2, n1 - 2);
        s1 = s0.Substring(n1+1);
        n1 = s1.IndexOf("|");
        if (n1 < 0)
        {
            textBox3.Text = s1;
            checkBox1.Checked = false;
        }
        else
        {
            textBox3.Text = s1.Substring(0, n1);
            checkBox1.Checked = true;
            textBox4.Text = s1.Substring(n1 + 1);
        }
    }
    else if (s1.Equals("3"))
    {
        radioButton3.Checked = true;
        n1 = s0.IndexOf("|");
        comboBox1.Text = s0.Substring(2, n1 - 2);
        //загрузить список полей справочника и выбрать нужное
        comboBox2.Text = s0.Substring(n1 + 1);
    }
}
private void button1_Click(object sender, EventArgs e)
{
    // загрузить выбранное правило из шаблона
    SetRule(comboBox3.Text);
    //
}
private void button2_Click(object sender, EventArgs e)
{   // сохранить правило в шаблон
    comboBox3.Text = RuleToString();
    comboBox3.Items.Add(comboBox3.Text);
}
public string RuleToString( )
{  // правило в строку
    String s0 = "";
    if (radioButton1.Checked == true)
    {
        s0 = "1." + textBox1.Text;
    }
    else if (radioButton2.Checked == true)
    {
        if ( checkBox1.Checked) { s0 = "|" + textBox4.Text; }
        s0 = "2." + textBox2.Text + "|" + textBox3.Text + s0;
    }
    else if (radioButton3.Checked == true)
    {
        s0 = "3." + comboBox1.Text + "|" + comboBox2.Text;
    }
```

```
            return s0;
        }
        public void SetShabl(List<MShabl> lMs)
        {
            bool blf;
            // проверим и дополним список шаблонов
            foreach (string ss in comboBox3.Items)
            {
                blf = false;
                foreach (MShabl mm in lMs)
                {
                    if ((mm.Тип_поля == label6.Text) &
                        (mm.Правило == ss))
                    {
                        blf = true;
                    }
                }
                if (!blf)
                {
                    MShabl ttnew = new MShabl(label6.Text, ss);
                    lMs.Add(ttnew);
                }
            }
        }

        public void SetDict(Diction dd)
        {
            // подгрузить справочники
            comboBox1.Items.Add(dd.Имя_Справочника);
        }
        private void button3_Click(object sender, EventArgs e)
        {
            // сохранить
            frFlag = true;
            this.Visible = false;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            // отменить
            frFlag = false;
            this.Visible = false;
        }
    }
}
```

Листинг Б.4 – Код файла Form3.cs

```
using System;
using System.Windows.Forms;

namespace ADD
{
    public partial class FormNewPrj : Form
    {
        public string pname;
        public string ppath;
        public FormNewPrj()
        {
            InitializeComponent();
            saveFileDialog1.Filter = "Text files(*.txt)|*.txt|Json
files(*.json)|*.json";
            saveFileDialog1.DefaultExt = "Json files(*.json)|*.json";
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            saveFileDialog1.InitialDirectory = textBoxDir.Text;
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                textBoxDir.Text = saveFileDialog1.FileName;
            }
            //это я пытаюсь получить имя файла в название проекта
            //textBox1.Text = saveFileDialog1.FileName;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            pname = "";
            ppath = "";
            this.Visible = false;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            pname = textBox1.Text;
            ppath = textBoxDir.Text;
            if (pname.Equals("")) { MessageBox.Show("Введите данные!",
"Ошибка!"); }

            // получаем выбранный файл
            string filename = saveFileDialog1.FileName;
            // сохраняем текст в файл
            if ((filename != null) & (filename != ""))
            {
                System.IO.File.WriteAllText(filename, textBox1.Text);
                // MessageBox.Show("Файл сохранен");
            }

            this.Visible = false;
        }
        private void textBoxDir_TextChanged(object sender, EventArgs e)
        {
        }
    }
}
```

Листинг Б.5 – Код файла Connect.cs

```
using Project.Application.Models;
using System;
using System.Windows.Forms;

namespace ADD
{
    public partial class Connect : Form
    {
        public string server;
        public string database;
        public Connect()
        {
            InitializeComponent();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            //сохранить, добавляет это подключение в список подключений
            server = comboBox1.Text;
            database = comboBox2.Text;
            this.Visible = false;
        }
        private void button3_Click(object sender, EventArgs e)
        {
```

```
            отмена подключения
            server = "";
            database = "";
            this.Visible = false;
        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            ServerPlugin server = new ServerPlugin();
            comboBox1.Items.Add(server.Name);
            comboBox1.Items.Add("serverName");
        }
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs
e)
        {
            DatabaseEntity database = new DatabaseEntity();
            comboBox2.Items.Add("databaseName");
        }
    }
}
```

Листинг Б.6 – Код файла ConnectionManager.cs

```
using Project.Domain.Models;
using System.Collections.Generic;

namespace Project.Application
{
    public static class ConnectionManager
    {
        public static List<Connection> GetServers(string pluginName)
        {
            var result = PluginsManager.Plugins[pluginName].GetServers();
            return result.Success ? result.Result : new List<Connection>();
        }

        public static List<string> GetDatabases(string plugin, Connection
connection)
        {
            var result =
PluginsManager.Plugins[plugin].GetDatabases(connection);
            return result.Success ? result.Result : new List<string>();
        }

        public static List<string> GetTables(string plugin, string database,
Connection connection)
        {
            var result = PluginsManager.Plugins[plugin].GetTables(connection,
database);
            return result.Success ? result.Result : new List<string>();
        }

        public static List<Column> GetColumns(string plugin, string database,
string table, Connection connection)
        {
            var result =
PluginsManager.Plugins[plugin].GetColumns(connection, database, table);
            return result.Success ? result.Result : new List<Column>();
        }

        public static TableData GetData(string plugin, string database,
string table, Connection connection)
        {
            var result = PluginsManager.Plugins[plugin].GetData(connection,
database, table);
```

```
            return result.Success ? result.Result : new TableData();
        }

        public static string GetConnection(Connection connection)
        {
            return
$"{connection.Name}:{connection.Server}:{connection.Port}:{connection.Login}:
{connection.Password}";
        }

        public static bool CreateTable(string plugin, string database, string
table, Connection connection, List<Column> columns)
        {
            return PluginsManager.Plugins[plugin].CreateTable(connection,
database, table, columns);
        }

        public static bool RemoveTable(string plugin, string database, string
table, Connection connection)
        {
            return PluginsManager.Plugins[plugin].RemoveTable(connection,
database, table);
        }

        public static bool InsertData(string plugin, string database, string
table, Connection connection, TableData tableData)
        {
            return PluginsManager.Plugins[plugin].InsertData(connection,
database, table, tableData);
        }

        public static bool GenerateData(string plugin, string database,
string table, Connection connection, int rows, Dictionary<string, RuleBase>
rules)
        {
            return PluginsManager.Plugins[plugin].GenerateData(connection,
database, table, rows, rules);
        }
    }
}
```

Листинг Б.7 – Код файла FormField.cs

```
using System;
using System.Windows.Forms;

namespace ADD
{
    public partial class FormField : Form
    {
        public int ffSave;
        private string pole;
        public FormField()
        {
            InitializeComponent();
            ffSave = 0;
        }
        public void MFGet(MRule MyDD,string tblname)
        {
            label5.Text = tblname;
            comboBox2.Text = MyDD.Название;
            pole = MyDD.Название;
            label9.Text = MyDD.Тип;
            label6.Text = MyDD.Размер;
            label7.Text = MyDD.Индекс;
```

```
            label10.Text = MyDD.Правило;
            checkBox1.Checked = true;
        }
        public void MFNew(string tblname)
        {
            label5.Text = tblname;
        }
        public MRule MFRess()
        {
            return new MRule(comboBox2.Text, label9.Text,
                label6.Text, label7.Text, label10.Text);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            // УДАЛИТЬ / Добавить
            if (checkBox1.Checked)
            {
                if (pole == comboBox2.Text) { ffSave = 0; }
                else {
                    ffSave = 1;  // добавляем поле в список полей
                }
            }
            else { ffSave = -1;  // удаляем поле из списка полей
            }

            FormLinks.ActiveForm.Visible = false;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            // ОТМЕНИТЬ
            ffSave = 0;
            FormLinks.ActiveForm.Visible = false;
        }
    }
}
```

Листинг Б.8 – Код файла FormTable.cs

```
using System;
using System.Windows.Forms;

namespace ADD
{
    public partial class FormTable : Form
    {
        public bool flTSave;
        public FormTable()
        {
            InitializeComponent();
        }
        public void MTGet(MTable MyDD)
        {
            comboBox1.Text = MyDD.Таблица;
            textBox3.Text = MyDD.Приоритет;
            textBox5.Text = MyDD.Связи;
            textBox4.Text = MyDD.Правила;
        }
        public MTable MTRess()
        {
            return new MTable(comboBox1.Text, textBox3.Text,
                textBox5.Text, textBox4.Text);
        }
        public void MTChange(MTable MyDD)
        {
            MyDD.Таблица = comboBox1.Text;
```

```
            MyDD.Приоритет = textBox3.Text;
            MyDD.Связи = textBox5.Text;
            MyDD.Правила = textBox4.Text;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            // Сохранение свойств таблицы
            flTSave = true;
            FormTable.ActiveForm.Visible = false;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            // выход без сохранения
            flTSave = false;
            FormTable.ActiveForm.Visible = false;
        }
    }
}
```

Листинг Б.9 – Код файла FormLinks.cs

```
using System;
using System.Windows.Forms;

namespace ADD
{
    public partial class FormLinks : Form
    {
        public bool flgLSave;
        public FormLinks()
        {
            InitializeComponent();
        }
        public void MLGet(MLinks MyDD)
        {
            comboBox1.Text = MyDD.Таблица_П;
            comboBox2.Text = MyDD.Таблица_Внешняя;
            comboBox3.Text = MyDD.Поле_связи;
            textBox1.Text = MyDD.Строк;
        }
        public MLinks MLRess()
        {
            return new MLinks(comboBox1.Text, comboBox2.Text,
                comboBox3.Text, textBox1.Text);
        }
        public void MLChange(MLinks MyDD)
        {
           MyDD.Таблица_П = comboBox1.Text;
            MyDD.Таблица_Внешняя = comboBox2.Text;
           MyDD.Поле_связи = comboBox3.Text;
           MyDD.Строк = textBox1.Text;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            // СОХРАНИТЬ
            flgLSave = true;
            FormLinks.ActiveForm.Visible = false;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            // ОТМЕНИТЬ
            flgLSave = false;
            FormLinks.ActiveForm.Visible = false;
        }
    }}
```