**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| Клиент-серверное приложение «Бронирование авиа и ЖД билетов» | |
|  | |
| по дисциплине | Разработка и эксплуатация удаленных баз данных |
|  | |

Пояснительная записка

|  |
| --- |
| КП.0902.12.000000.00 ПЗ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 4-ИС-2 |  |  |  | Пахомов И А |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Лобанова Е А |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *( И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Лобанова Е А |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc120966390)

[1 Исследовательский раздел 5](#_Toc120966391)

[2 Конструкторский раздел 9](#_Toc120966392)

[2.1 Проектирование информационной модели данных 9](#_Toc120966393)

[2.2 Проектирование серверной части приложения 9](#_Toc120966394)

[2.2.1 Разработка структуры сущностей базы данных 10](#_Toc120966395)

[2.2.2 Разработка схемы базы данных 12](#_Toc120966396)

[2.3 Проектирование клиентской части приложения 12](#_Toc120966397)

[2.3.1 Разработка модульной схемы 12](#_Toc120966398)

[2.3.2 Разработка пользовательского интерфейса 12](#_Toc120966399)

[2.3.3 Организация доступа к объектам базы данных 12](#_Toc120966400)

[2.3.4 Разработка блок-схем алгоритмов процедур и функций 13](#_Toc120966401)

[2.4 Обеспечение коллективного доступа. Защита информации 13](#_Toc120966402)

[3 Технологический раздел 17](#_Toc120966403)

[3.1 Тестирование и отладка приложения 17](#_Toc120966404)

[3.2 Инструкция администратора базы данных 17](#_Toc120966405)

[3.3 Инструкция по эксплуатации приложения 18](#_Toc120966406)

[4 Раздел охраны труда 18](#_Toc120966407)

[Заключение 20](#_Toc120966408)

[Список источников 21](#_Toc120966409)

[Приложение А 22](#_Toc120966410)

Введение

На сегодняшний день вычислительная техника тесно связана практически со всеми людьми на планете. Вычислительная техника, упрощает жизнь в десятки раз. С её помощью человек может развлечься, найти буквально все, что хочет в глобальной сети Интернет. В данный момент, человек не может представить жизнь без всех удобств, что дает вычислительная техника.

Вычислительная техника, не всегда была такой удобной и простой для понимания. Все начиналось с машинного кода, на основе которого работают языки программирования, способствующие работе современных программ и приложений. Упомянутые выше языки программирования служат для упрощения программного кода. Первый язык программирования «Ассемблер» – низкоуровневый, создан по принципу «Одна инструкция – одна строка». Низкоуровневые языки, ориентированы на конкретный тип процессора и учитывают его особенности. Их появление, действительно облегчило написание кода, так как вместо единиц и нулей, можно было писать программу командами, состоящую из символов, приближенных к разговорному языку[1].

Наиболее важной проблемой, с которыми столкнулись разработчики языков второго поколения, стала задача убедить клиентов в том, что созданный компилятором код выполняется достаточно хорошо, чтобы оправдать отказ от программирования на ассемблере. Скептицизм по поводу возможности создания эффективных программ с помощью автоматических компиляторов был довольно распространён, поэтому разработчикам таких систем должны были продемонстрировать, что они действительно могут генерировать почти такой же эффективный код, как и при ручном кодировании, причём практически для любой исходной задачи. Но сложность разработки в нём больших программных комплексов, привела к появлению языков третьего поколения — языков высокого уровня. Такие языки были машинно-независимы, для каждого такого языка был компилятор. В связи с ростом сложности программ и развитием программных средств, возникла необходимость увеличить производительность труда программистов, что привело к разработке структурного программирования. Следующим достижением были [процедуры](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [функции](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), то есть, если есть задача, которая выполняется несколько раз, то её можно объявить как функцию или как процедуру и в выполнении программы просто вызывать её. Программный код становится меньше. Следующим достижением было объединение разнородных данных, которые используются в программе в связке, в структуры. С развитием технологий родилось объектно-ориентированное программирование. При использовании структур данных в программе вырабатываются и соответствующие им функции для работы с ними. Это привело к мысли их объединить и использовать совместно, так появились классы. Теперь программирование можно было разбить на классы и тестировать не всю программу, состоящую из 10 000 строк кода, а разбить программу на 100 классов, и тестировать каждый класс. Объектно-ориентированное программирование сочетает лучшие принципы структурного программирования с новыми концепциями [инкапсуляции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [полиморфизма подтипов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) и [наследования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))[2].

В ходе работы над курсовым проектом требуется разработать клиент-серверное приложение на тему «Бронирование авиа и ЖД билетов» для одной из современных сред разработки программного обеспечения. Планируется, что клиент-серверное приложение, будет выполнять все необходимое для полноценного функционала. Актуальность проекта, прежде всего, заключается в возможности использовать его результаты в прикладных программах. Программистам не потребуется писать программный код с нуля, они смогут посмотреть где и как использовался метод или функция, достаточно понять программный код и взять себе необходимое.

Как таковое приложение будет полезно для людей, которые хотят открыть свой бизнес. Для начинающих программистов данный проект будет полезен тем, что позволит взглянуть на возможности, которые им предстоят в будущем, это заинтересует их обучатся программированию.

1 Исследовательский раздел

Для разработки клиент-серверных систем существует два подхода. Первый подход построение систем на основе двухзвенной архитектуры. Состоит из клиентской и серверной части, где как правило серверная часть представляет собой сервер базы данных, на котором расположены общие данные, а клиентская часть представляет приложение, которое связывается с сервером базы данных, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, так как нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, потому что могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть «Интернет», выбирают именно такую архитектуру построения клиент/серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Пользователю необходимо иметь в наличии клиентскую часть, а у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Из-за этих проблем, в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Второй подход построение систем на основе трехзвенной архитектуры состоит из двух частей: клиента и сервера. Но серверная часть в этой архитектуре представляет собой сервер приложений и сервер БД, а в качестве клиента выступает web-браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие web-браузера. Все данные представляются в виде html-разметки, с использование графики и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI-интерфейса. Сервер приложений общается с сервером БД, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов и интерфейсов передачи данных. Злоумышленник может осуществить взлом системы, если она будет не достаточно хорошо проверять поступившие запросы от клиента.

WPF имеет следующие преимущества:

* Приложения на основе страниц;
* Стили и шаблоны;
* Декларативный пользовательский интерфейс;
* Поддержка аудио и видео;
* Анимация;
* Веб-подобная модель компоновки;
* Хорошо развитая текстовая модель.

Для реализации двухзвенной архитектуры выбрана платформа WPF.

Для реализации проекта существуют две наиболее подходящие среды программирования - Borland Delphi 7 и Visual Studio 2019. Рассмотрим Borland Delphi 7. Среда не новая, легкая в понимании, практичная, есть возможность использовать только один язык программирования и к сожалению, среда устарела.

Visual Studio 2019 – это новая среда программирования с удобным интерфейсом и возможностью писать код на многих языках программирования. Онавключает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и в качестве отладчика машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). В отличии от других сред Visual Studio более удобен и практичен, с этой средой намного приятнее работать. У этой среды есть минус – не все ошибки, имеют четкое решение[3].

Сравним среды программирования, начиная с Borland Delphi7. Плюсы среды: работает на старых компьютерах, занимает не много места на диске, работает на Microsoft Windows 2000 и старше. Система предсказуема, то есть имеет решения на все ошибки.

Минусы среды Borland Delphi7: при компилировании имеет достаточно большой размер .exe фалов на диске, писать код можно только на одном языке (Delphi). В целом среда неплохая, для своих задач будет хорошо справляться.

Плюсы Visual Studio 2019: совершенно новая система, удобный интерфейс и дизайн, много полезных дополнений, можно писать на многих языках, к примеру на C#, C++, F#. Среда постоянно обновляется.

Минусы Visual Studio 2019: присутствуют ошибки, на которые так и не найдены решения, не бесплатна, занимает большой объем на диске.

Также был вопрос в выборе СУБД. Наиболее подходящие это Microsoft SQL Server 2019 и Firebird.

Достоинствами MS SQL Server можно отнести масштабирование системы. Взаимодействовать с ней можно как на простых ноутбуках, так и на ПК с мощным процессором, который способен обрабатывать большой объем запросов. Размер страниц – до 8 Кб. Данные извлекаются быстро, а сложную информацию удобнее хранить. Система обрабатывает транзакции в интерактивном режиме, есть динамическая блокировка. Автоматизация рутинных административных задач, например, управление блокировками и памятью, редактура размеров файлов. В программе продуманы настройки, можно создавать профили пользователей. Удобный поиск, можно осуществлять по фразам, словам, тексту либо создавать ключевые индексы.

Недостатками данной СУБД является зависимость от операционной среды - работает только с системой Windows. И при использовании в больших масштабах становится еще и платным продуктом.

Достоинством Firebird нужно отметить высокую эффективность, компактность, языковую поддержку для хранимых триггеров и процедур, а также многоверсионную архитектуру, благодаря которой осуществляется параллельная обработка аналитических и оперативных запросов (это объясняется тем, что пишущие пользователи не блокируются читающими).

К недостаткам можно отнести, что данная СУБД является немного устаревшим вариантом по сравнению с SQL Server.

Для реализации данного проекта, была выбрана Microsoft Visual Studio 2019, так как данная среда более современна и востребована. А также СУБД Microsoft SQL Server 2019 для лучшей совместимости Microsoft Visual Studio 2019.

В глобальной сети «Интернет» есть аналоги разрабатываемого проекта. Рассмотрим сайт [www.aviasales.ru/](http://www.aviasales.ru/4) [4]. Пользователь может найти и забронировать/купить подходящий ему билет. Недостатков у аналога нет, а достоинств много, так как это самый крупный сайт по продаже авиабилетов.

В проекте планируется вывод подходящих для пользователя авиа или ЖД билетов, также их бронирование. Для администратора планируется добавить дополнительный функционал – команды для работы со справочниками, а именно, добавление, копирование, изменение, и удаление записей, а также фильтрация данных в справочнике.

В качестве основного процесса будет выступать особая функция, бронирование билета, так как она наиболее сложная.

2 Конструкторский раздел

2.1 Проектирование информационной модели данных

Модель «Черная сфера»— модель системы, с известными выходными и входными параметрами и неизвестным внутренним устройством. Исследуя известные параметры, с помощью такой модели можно получить представление о внутреннем устройстве. На рисунке 2.1 изображено общее представление черной сферы.



Рисунок 2.1 – Общее представление «черной сферы»

В качестве входных параметров черная сфера использует: управляющие воздействие пользователя, данные об пользователе, размеры. На выходе будем получать изображение и изменения в базе данных. На рисунке 2.2 изображено представление разрабатываемого компонента в виде модели «Черная сфера»



Рисунок 2.2 – Представление разрабатываемого компонента   
в виде модели “Чёрная сфера”

2.2 Проектирование серверной части приложения

2.2.1 Разработка структуры сущностей базы данных

В разделе 2.1 была получена модель «Черная сферы», из которой следуют входные и выходные данные.

Для выявление всех сущностей необходимо провести ряд декомпозиций. Декомпозиция — разделение целого на части. На первом этапе декомпозиции выявлены следующие сущности: клиент и маршруты. На рисунке 2.3 представлен первый этап декомпозиции.

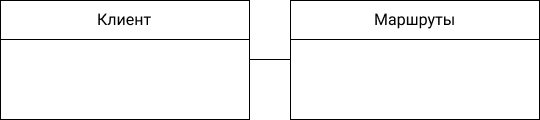


Рисунок 2.3 – Первый этап декомпозиции

На следующем этапе декомпозиции выявляются еще необходимые сущности, такие как транспортные компании и транспорт. На рисунке 2.4 представлен второй этап декомпозиции.



Рисунок 2.4 – Второй этап декомпозиции

На последнем этапе декомпозиции выявляется необходимость хранить данные о бронях, в ходе этого добавляется сущность – брони. На рисунке 2.5 представлен третий этап декомпозиции.



Рисунок 2.5 – Третий этап декомпозиции

Для более детального анализа необходимо сделать логическую модель базы данных. На рисунке 2.6 представлена логическая модель базы данных.

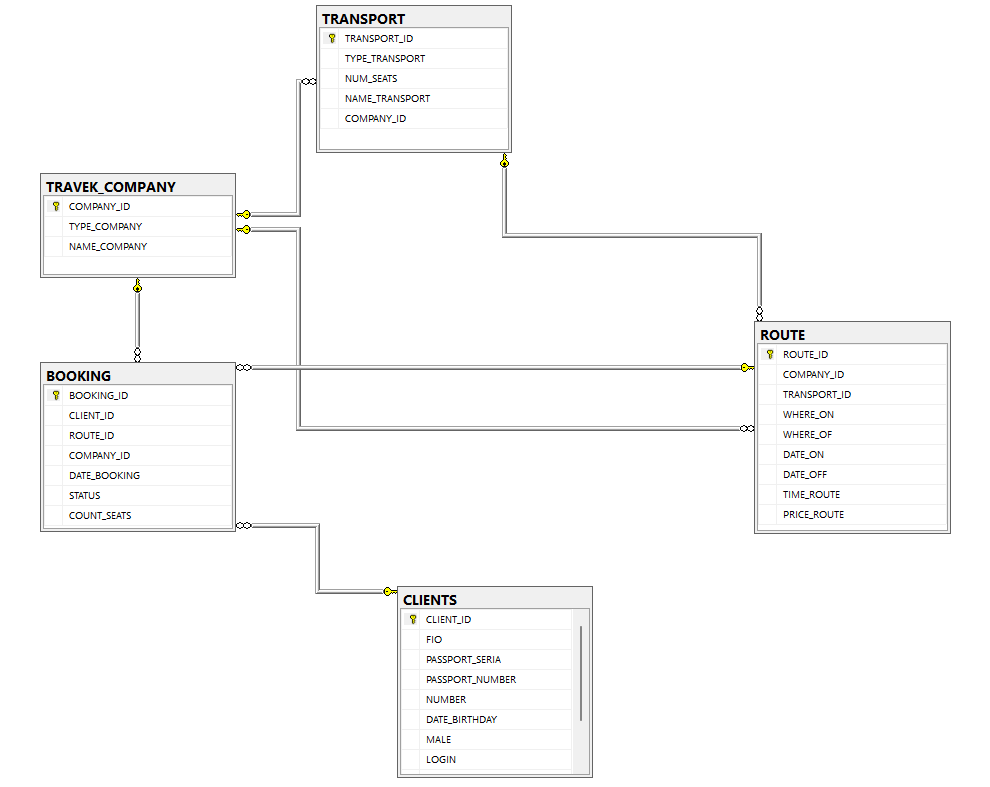


Рисунок 2.6 – Логическая модель базы данных

Для того, чтобы определится с типами данными в базе данных, необходимо создать физическую модель. На рисунке 2.7 представлена физическая модель

Изображение выглядит как текст, снимок экрана, внутренний

Автоматически созданное описание

Рисунок 2.7 – Физическая модель базы данных

2.2.2 Разработка схемы базы данных

В разделе 2.2.1 была получена физическая модель базы данных, состоящая из 5 сущностей: транспорт, транспортные компании, клиенты, маршруты и брони. В таблице 2.1 представлено подробное описание сущностей

Начало таблицы 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя сущности | Атрибуты сущности | Тип данных | Назначение сущности | Подчиняет | Подчиняется |
| TRAVEK\_COMPANY | COMPANY\_ID | int | Необходима для хранения данных о транспортных компаниях | - | BOOKING ROUTE TRANSPORT |
| TYPE\_COMPANY | bit |
| NAME\_ COMPANY | nvarchar(MAX) |

Продолжение таблицы 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя сущности | Атрибуты сущности | Тип данных | Назначение сущности | Подчиняет | Подчиняется |
| TRANSPORT | TRANSPORT\_ID | int | Необходима для хранения данных транспорте | TRAVEK\_COMPANY | ROUTE |
| NAME\_TRANSPORT | nvarchar(MAX) |
| TYPE\_TRANSPORT | bit |
| COMPANY\_ID | int |
| NUM\_SEATS | int |
| BOOKING | BOOKING\_ID | int | Необходима для хранения данных о бронях | TRAVEK\_COMPANY ROUTE CLIENTS | - |
| CLIENT\_ID | int |
| ROUTE\_ID | int |
| COMPANY\_ID | int |
| DATE\_BOOKING | date |
| STATUS | bit |
| COUNT\_SEATS | int |
| CLIENTS | CLIENT\_ID | int | Необходима для хранения данных о клиентах, также данных для входа | - | BOOKING |
| FIO | nvarchar(MAX) |
| PASSPORT\_SERIA | int |
| PASSPORT\_NUMBER | int |
| NUMBER | nvarchar(MAX) |
| DATE\_BIRTHDAY | date |
| MALE | bit |
| LOGIN | nvarchar(50) |
| PASSWORD | nvarchar(50) |
| ROLE\_ID | bit |

Окончание таблицы 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя сущности | Атрибуты сущности | Тип данных | Назначение сущности | Подчиняет | Подчиняется |
| ROUTE | ROUTE\_ID | int | Необходима для хранения информации о маршрутах | TRANSPORT TRAVEK\_COMPANY | BOOKING |
| COMPANY\_ID | int |
| TRANSPORT\_ID | int |
| WHERE\_ON | nvarchar(MAX) |
| WHERE\_OF | nvarchar(MAX) |
| DATE\_ON | date |
| DATE\_OFF | date |
| TIME\_ROUTE | time(7) |
| PRICE\_ROUTE | money |

2.3 Проектирование клиентской части приложения

2.3.1 Разработка модульной схемы

Главным элементом в системе графического интерфейса в WPF является окно, которое содержит все необходимые элементы управления. Окно в WPF представлено классом Window, который является производным от класса Con-tentControl.

Модель для создания страниц в WPF во многом похожа на модель для создания окон. Хотя создавать объекты страниц можно и с помощью одного лишь кода, обычно для каждой страницы создается файл XAML и файл отделенного кода. При компиляции этого приложения компилятор создает производный класс страницы, который объединяет написанный разработчиком код с генерируемыми автоматически связующими элементами (такими как поля, которые ссылаются на каждый именованный элемент на странице).

Страницу можно добавлять в любой проект WPF. Для этого в Visual Studio нужно выбрать в меню Project (Проект) пункт Add Page (Добавить страницу). Хотя страницы и являются самым высокоуровневым компонентом пользовательского интерфейса при проектировании приложения, во время его выполнения контейнером наивысшего уровня они уже не будут. Вместо этого они обслуживаются в другом контейнере. Именно в этом и состоит секрет гибкости, обеспечиваемой WPF в случае страничных приложений.[5]

На рисунке 2.8 изображено представление клиентской части приложения в виде модульной схемы.

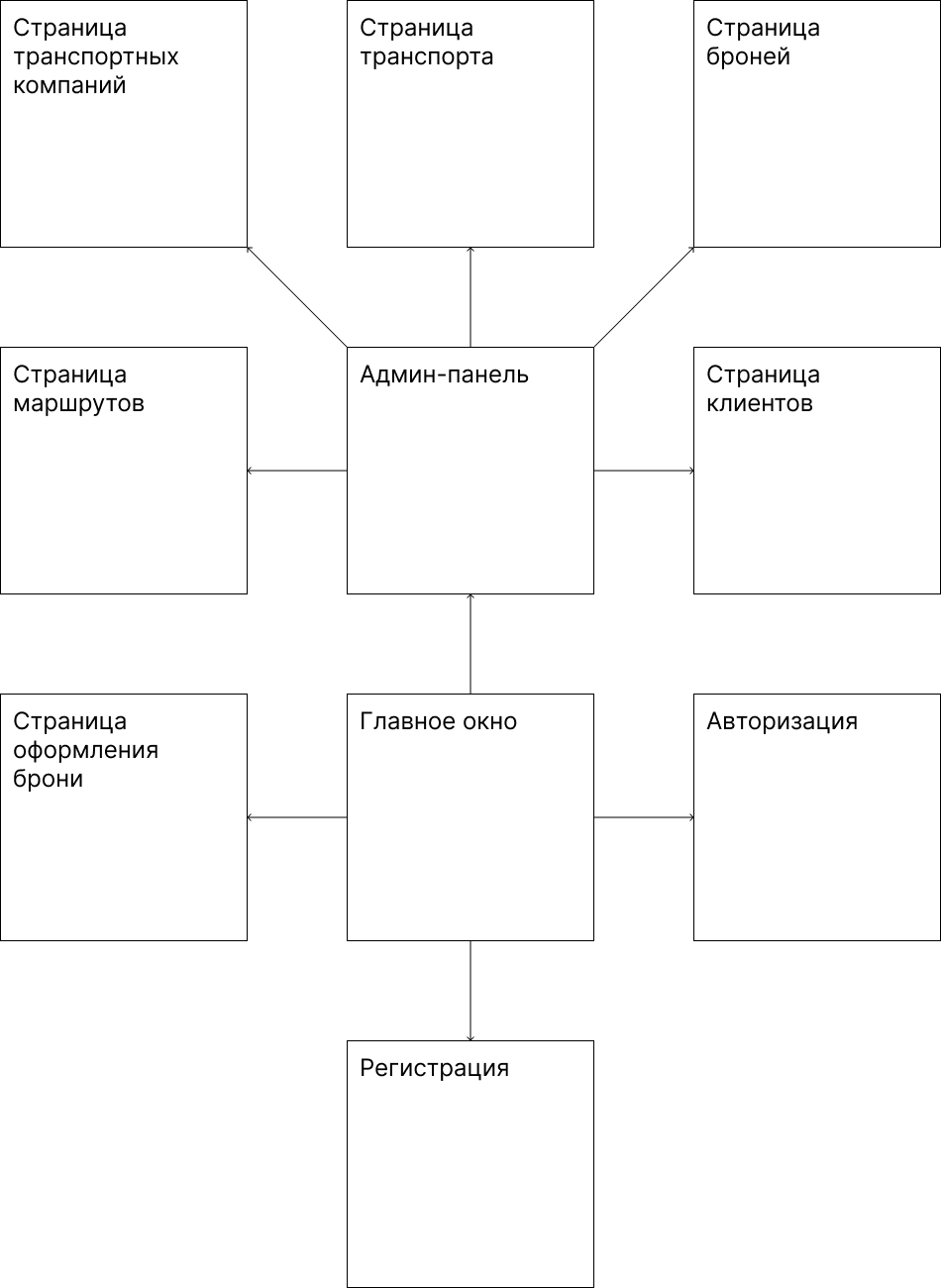


Рисунок 2.8 - Модульная схема клиентского приложения

В состав модульной схемы включены следующие элементы:

- окно авторизации, для авторизации пользователя;

- окно регистрации, для регистрации пользователя;

- главное окно, для бронирования билетов и входа в админ-панель;

- админ-панель, для просмотра справочников.

2.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

**Графический пользовательский интерфейс (Graphical User Interface или GUI).** Самый популярный тип UI. Представляет собой окошко с различными элементами управления. Пользователи взаимодействуют с ними с помощью клавиатуры, мыши и голосовых команд: жмут на кнопки, тыкают мышкой, смахивают пальцем.

**XAML (Extensible Application Markup Language — расширяемый язык разметки приложений)** представляет собой язык разметки, используемый для создания экземпляров объектов .NET. Хотя язык XAML — это технология, которая может быть применима ко многим различным предметным областям, его главное назначение — конструирование пользовательских интерфейсов WPF. Другими словами, документы XAML определяют расположение панелей, кнопок и прочих элементов управления, составляющих окна в приложении WPF.

В состав модульной схемы входят следующие блоки:

* главное окно, представляющая из себя модель, для поиска и бронирования билетов, а также навигационную модель, для перехода на другие страницы используя кнопки. Вставка страниц происходит в Frame.

На рисунке 2.9 представлен код кнопок для перехода на другие страницы.

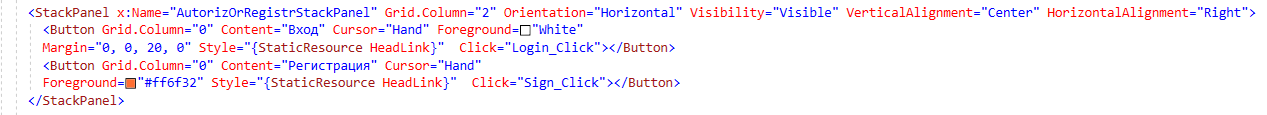


Рисунок 2.9 - Код кнопок для переходов на страницы

На рисунке 2.10 представлен код Frame.



Рисунок 2.10 - Код кнопок Frame

* окно авторизации, пользователь использует 2 текстовых поля для ввода данных (логина и пароля), так же есть 2 кнопки (Войти и Отмена);

На рисунке 2.11 представлен код полей для ввода данных.

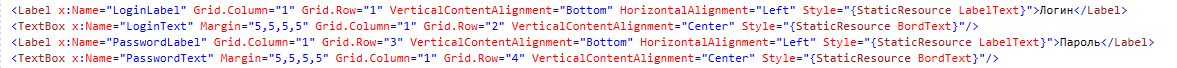


Рисунок 2.11 - Код полей для ввода данных

На рисунке 2.12 представлен код кнопки Войти и Отмена.

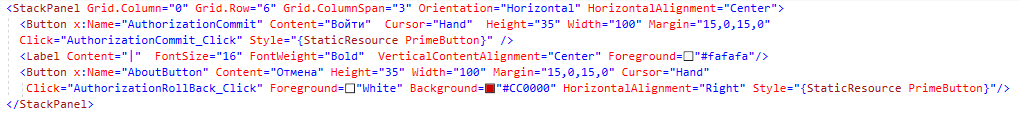


Рисунок 2.12 - Код кнопки кнопки Войти и Отмена

* окно регистрации, почти ничем не отличается от окна авторизации, добавилось больше полей для ввода данных и капча (CAPTCHa), также функция скрытия и показа пароля;

На рисунке 2.13 представлен код кнопки, для показа и скрытия пароля.

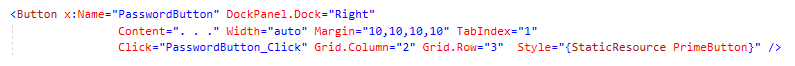


Рисунок 2.13 - Код кнопки кнопки показа и скрытия пароля

* базовые страницы для вывода информации, имеют DataGrid для вывода информации с базы данных, также имеются ComboBox с тестовым полем для фильтрации данных в DataGrid. Присутствует 4 базовых кнопки для работы с данными (добавить, копировать, изменить, удалить). По нажатию на кнопки добавить, копировать, изменить появляется дополнительная колонка с полями для изменения или добавления данных.

На рисунке 2.14 представлен код DataGrid.

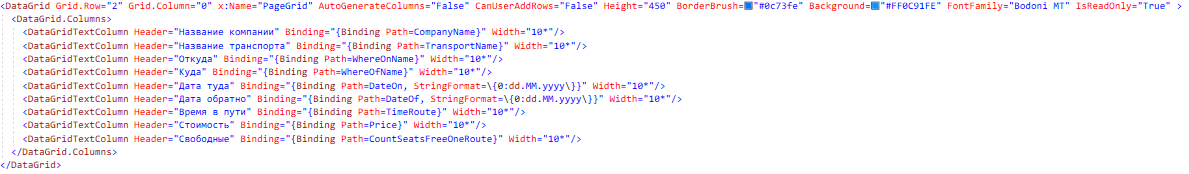


Рисунок 2.14 - Код DataGrid.

На рисунке 2.15 представлен код базовых кнопок и фильтра.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.15 - Код базовых кнопок и фильтра.

На рисунке 2.16 представлен код дополнительного столбца.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.16 - Код дополнительных столбцов.

2.3.3 Организация доступа к объектам базы данных

В WPF привязка (binding) является инструментом программирования, без которого не обходится ни одно клиент-серверное приложение.

Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника. В случае модификации объекта-источника, объект-приемник также будет модифицирован.

На рисунке 2.17 показан пример использования Binding в XAML разметке.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.17 – Использование Binding.

DataGrid верстают под конкретное содержимое, обеспечивая связь с точным источником данных. Допустим для таблицы TRANSPORT на стороне клиента будет доступно для просмотра поле: TYPE\_TRANSPORT (или любое другое, которое есть в сущности TRANSPORT).

2.3.4 Разработка блок-схем алгоритмов процедур и функций

Текст

2.4 Обеспечение коллективного доступа. Защита информации

Данные в базе являются разделяемым ресурсом. Многопользовательский доступ к данным подразумевает одновременное выполнение двух и более запросов к одним и тем же объектам данных (таблицам, блокам и т.п.). Система управления базами данных включает в себя средства, обеспечивающие доступ к данным нескольким пользователям одновременно без нарушения целостности данных. В основе этого лежат механизмы транзакции и блокировки. Транзакция - последовательность выполнения операций в БД, которая рассматривается как некоторое неделимое действие, осмысленное с точки зрения пользователя. Механизм блокировки заключается в том, что если несколько пользователей хотят одновременно изменить одни и те же данные, то только транзакция первого пользователя получит доступ к общему ресурсу, а остальные транзакции будут ждать, пока она не завершится и не разблокирует этот ресурс. После чего начнет выполняться следующую транзакцию и т. д.[6]

При большом количестве пользователей традиционные подсистемы управления доступом становятся крайне сложными для администрирования. Число связей в них пропорционально произведению количества пользователей на количество объектов. Необходимы решения в объектно-ориентированном стиле, способные эту сложность понизить. Таким решением является ролевое управление доступом. Суть его в том, что между пользователями и их привилегиями появляются промежуточные сущности – роли. Для каждого пользователя одновременно могут быть активными несколько ролей, каждая из которых дает ему определенные права. Ролевой доступ нейтрален по отношению к конкретным видам прав и способам их проверки; его можно рассматривать как объектно-ориентированный каркас, облегчающий администрирование, поскольку он позволяет сделать подсистему разграничения доступа управляемой при сколь угодно большом числе пользователей, прежде всего за счет установления между ролями связей, аналогичных наследованию в объектно-ориентированных системах. Кроме того, ролей должно быть значительно меньше, чем пользователей. В результате число администрируемых связей становится пропорциональным сумме (а не произведению) количества пользователей и объектов, что по порядку величины уменьшить уже невозможно.[7]

Для баз данных и систем, основанных на хранении данных, важны три основных аспекта информационной безопасности – конфиденциальность, целостность и доступность, т. е. под защитой данных понимают систему мер, предохраняющих данные от несанкционированного использования, разрушения или искажения. В английском языке используются следующие термины: secrecy, integrity, and availability of data. Общая идея защиты базы данных состоит в следовании рекомендациям, сформулированным для класса безопасности C2 в «Критериях оценки надежных компьютерных систем». Необходимо полагать, что абсолютная защита данных практически не реализуема, поэтому обычно довольствуются относительной защитой информации - гарантированно защищают ее на тот период времени, пока несанкционированный доступ к ней, потеря ее целостности или доступности влечет какие-либо последствия. Перечислим типовые ситуации, в которых хранимые данные могут быть похищены, искажены или потеряны.

Несанкционированный доступ (НСД). Если любой желающий может получить доступ к любым хранимым данным и внести в них любые изменения, то данные могут быть разрушены некомпетентным или злонамеренным пользователем либо использованы с ущемлением прав владельца или во вред ему.

Неуправляемый параллелизм. Как правило, в многопользовательской системе с базой данных одновременно работает несколько пользователей. Некоторые из них могут пытаться одновременно изменять состояние БД. Если они будут делать это независимо, то результирующее состояние БД может оказаться несогласованным.

Локальный сбой. В процессе выполнения прикладной программы может возникнуть аварийная ситуация (например, деление на ноль), в результате которой выполнение программы будет прекращено. Если прикладная программа выполняла обновление данных, то БД может оказаться в несогласованном состоянии.

Потеря оперативной памяти (мягкий сбой). Несмотря на высокую надежность персональных компьютеров, в любой момент может произойти системный сбой (например, отключение питания), в результате которого будет утрачено содержимое системных буферов и буферов приложений, размещенных в оперативной памяти. Подобные ситуации называются мягкими сбоями системы. Поскольку состояние БД в момент мягкого сбоя непредсказуемо, оно может оказаться несогласованным после перезагрузки системы. В отличие от локального сбоя при мягком сбое могут пострадать данные всех пользователей.

Требования защиты данных от разрушения в обобщенном виде сводятся к следующему: никакие случайные или преднамеренные разрушения данных не могут быть необратимыми.

Вышеперечисленные ситуации могут быть, в свою очередь, следствием: некомпетентных, безответственных или преступных действий пользователя; несогласованных обновлений, производимых одновременно и независимо несколькими пользователями; возникновения необработанной должным образом ошибки в прикладной программе; внедрением вредоносного кода в программное обеспечение; аварийного отключения энергии, сбоя или отказа различных устройств системы, нарушения линий связи; физическим уничтожением компьютерной системы или ее частей.

Следовательно, для защиты данных от потери целостности, разрушения и несанкционированного использования система должна обеспечивать: поддержку определяемых информационной политикой предприятия ограничений прав и полномочий доступа пользователей к данным; защиту от вредоносных действий пользователей, направленных на повреждение данных или получения конфиденциальных данных на основе анализа доступной информации; дисциплину параллельной работы многих пользователей, исключающую возможность внесения несогласованных изменений в состояние БД; восстановление целостного состояния БД после локальных сбоев и мягких сбоев системы; восстановление БД после жестких сбоев.[8]

В соответствии с назначенными ролями определяются разрешенные полномочия пользователей. Ключевыми достоинствами данной модели являются: легкая управляемость и контролируемость поведения объектов и субъектов в системе; возможности построения иерархий и масштабирования; выражение средствами ролевой модели дискреционную и мандатную модели разграничения доступа, которые получили наибольшее применение в различных автоматизированных системах. На рисунке 2.16 представлено окно авторизации

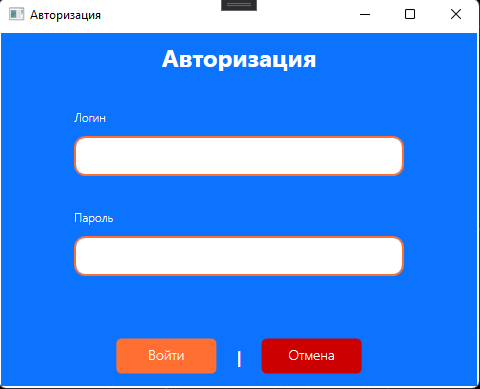


Рисунок 2.16 – Окно авторизации

Пользователь вводит в поля логин и пароль, далее нажимает войти и авторизуется.

На рисунке 2.17 представлено окно регистрации.



Рисунок 2.17 – Окно регистрации

Пользователь заполняет все необходимые поля. Если логин уже существует, то возникает предупреждение о том, что логин занят. Также пароль должен быть надежным, если в нем нет цифр и заглавных букв, то пользователь не сможет зарегистрироваться. Присутствует CAPTCHa, если пользователь не введёт или введёт её неправильно, то не сможет зарегистрироваться. Также пользователь должен быть совершеннолетним, чтобы пройти регистрацию.

3 Технологический раздел

3.1 Тестирование и отладка приложения

В разработке программных продуктов важную часть занимает процесс отладки полученной программы. Отладка продукта осуществляется для своевременного обнаружения и исправления ошибок при работе, перед тем как выпустить его на рынок.

Отладка программного продукта представляет собой ввод различных наборов данных, с той целью, чтобы проверить как отреагирует программа в различных ситуациях.

Для опасных методов и событий разрабатываемой информационной системы, был проведен процесс отладки, результат которой представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| ChangeWindow | Вызывается при нажатии на кнопки перехода на другие окна | Закрытие текущего окно и открытие нового |
| CheckPassword | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Вызывает окно предупреждения, если пароль был введен некорректно. |
| CheckPhoneNumber | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Вызывает окно предупреждения, если номер телефона был введен некорректно |
| CheckLoginExist | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Вызывает окно предупреждения, если пользователь с таким именем уже существует |
| Page\_Loaded | Вызывается при создании страниц клиентов, фильмов и сеансов | Заполнение comboBox для фильтрации записями с заголовками таблицы и блокировка сортировки столбцов нажатием на заголовок столбца |
| UpdateGrid | Вызывается при добавлении и удалении записей | Обновление таблицы новыми данными |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| GenerateCaptcha | Вызывается при нажатии кнопки «Регистрация» | Заполнение текстового поля случайными символами длиной в 6 символов |
| ShowUser | Вызовется при сборке главного окна если пользователь авторизован | Отображение панели с именем пользователя и кнопкой выхода. Если пользователь с доступом админа, отобразиться кнопка с панелью администратора |
| DlgLoad | Вызовется при помощи кнопок добавления, копирования и изменения записей на странице, и кнопок добавления и отмены в окне редактирования записи | Отображение и скрытие окна редактирования записей |

3.2 Инструкция администратора базы данных

Перед началом работы с клиент-серверным приложением необходимо установить и настроить SQL Server 2019. Перед установкой необходимо выбрать в сети компьютер, который будет выступать в роли сервера для базы данных.

Установка сервера осуществляется с помощью дистрибутива [Microsoft SQL Server 2019](https://softcomputers.download/SQL-Server-2019/SQLServer2019-x64-RUS.iso). Далее открываем установщик и начинаем инсталляцию приложения, как показано на рисунке 3.1. Навигация между окнами осуществляется с помощью кнопок «Ок», «Отмена» или «Завершить».

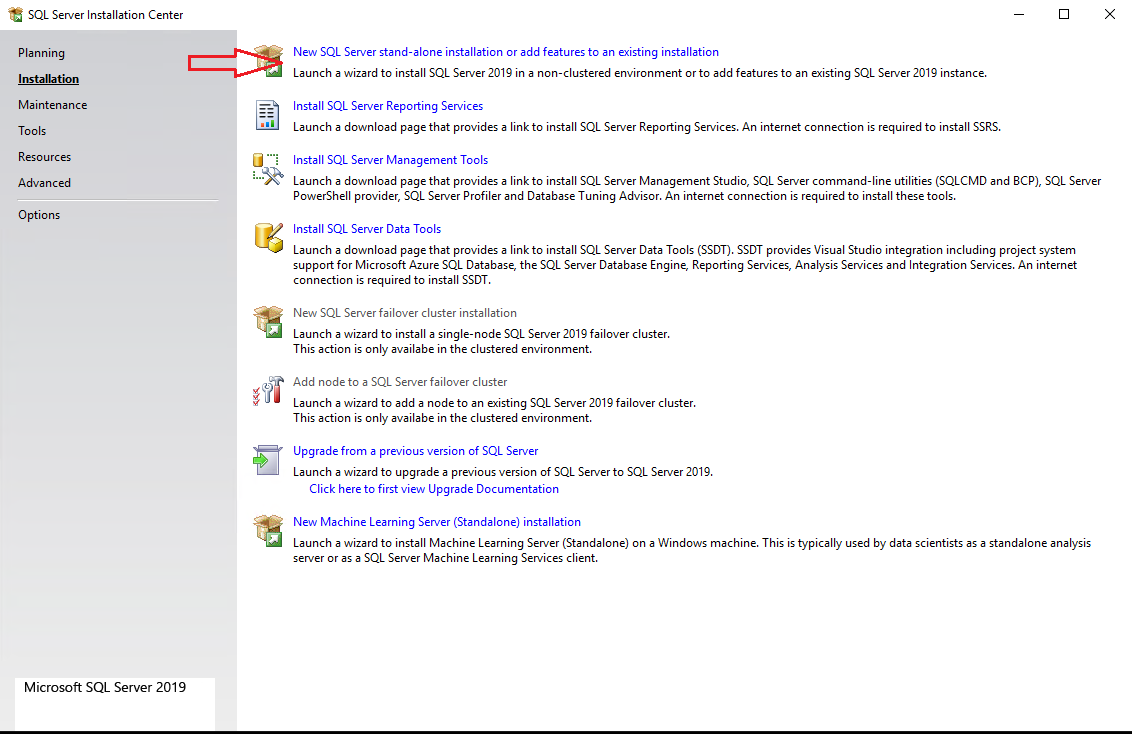


Рисунок 3.1 – Центр установки SQL Server

Далее, у нас запросят выбрать версию установки. Можно ввести ключ продукта, как показано на рисунке 3.2 или выбрать бесплатную версию. Бесплатные ознакомительные версии работают только 180 дней пробного периода, далее приложение закрывает свой функционал.

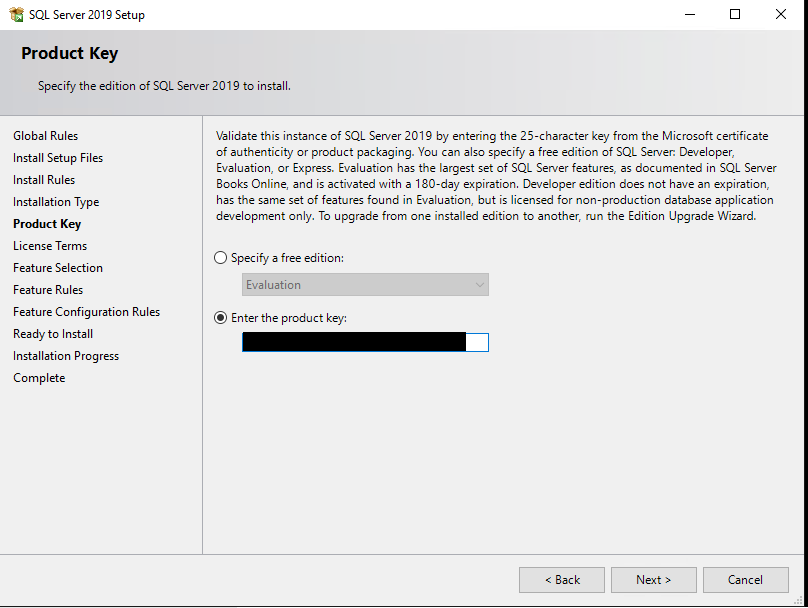


Рисунок 3.2 – Выбор версии установки

На рисунке 3.3, предлагается выбрать компоненты для установки в SQL Server. После того как все выбрано, нажать кнопку «Next»

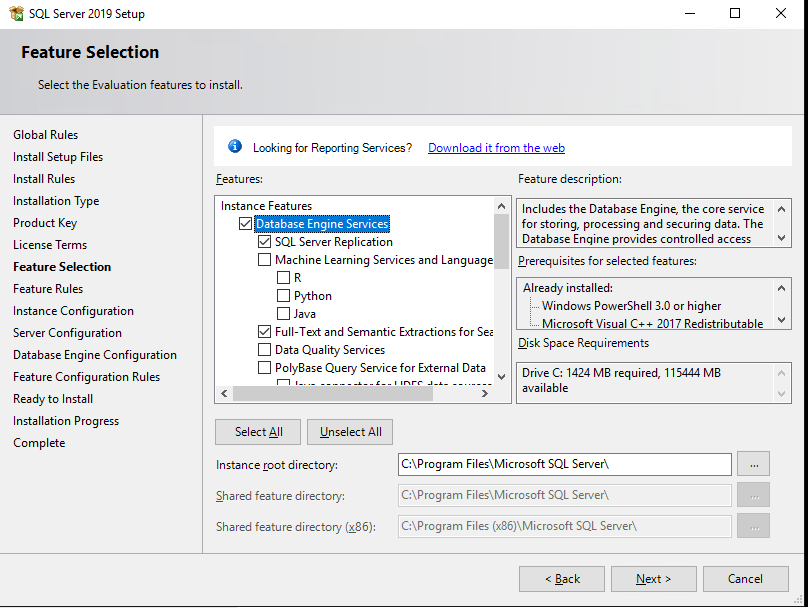


Рисунок 3.3 – Выбор компонентов

Далее задаем имя интерфейса, как показано на рисунке 3.4. Если вы планируете использовать только один экземпляр, то можете выбрать пункт «Экземпляр по умолчанию», в этом случае подключение будет осуществляться к этому экземпляру.

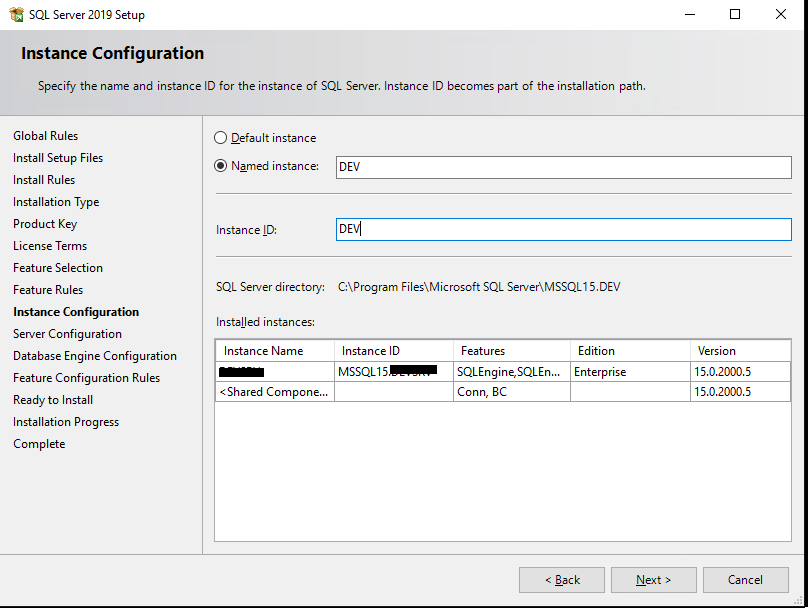


Рисунок 3.4 – Настройка экземпляра

На рисунке 3.5, переходим к настройкам конфигурации сервера, обратим внимание на раздел «Имя учетной записи» — это те учетные записи из-под которых будут стартовать службы SQL Server Agent, SQL Server Database, SQL Server Browser. По умолчанию используется ваша локальная учетная запись. Обязательно ставим галочку в чекбокс «Grant Perform Volume Maintenance Task privilege to SQL Server Database Engine», его включение означает что старые данные не будут перезаписываться нулями при создании, восстановлении базы данных.

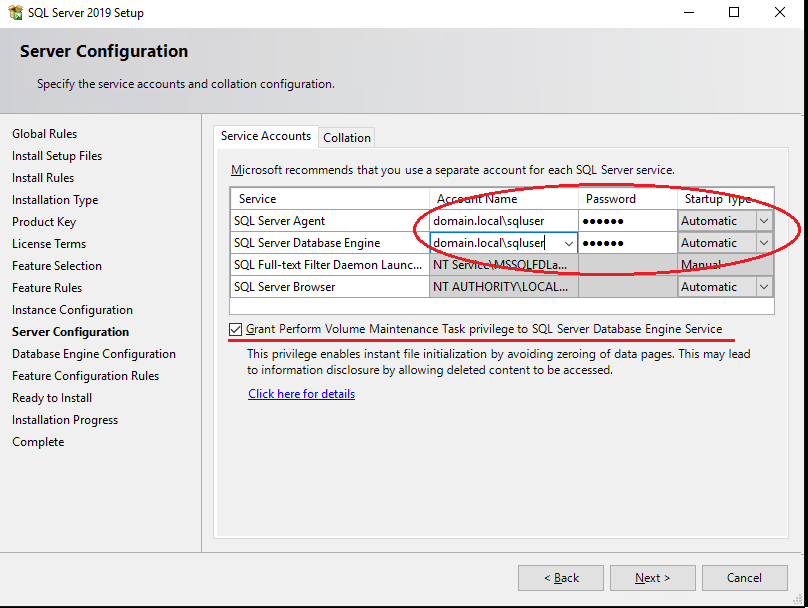


Рисунок 3.5 – Настройка конфигурации сервера

В настройках можно выбрать учетную запись домена, если ваш сервер входит в домен. Так же, можно задать тип запуска какой-либо службы. Поставить ее на автозапуск, вручную, или вообще отключить. Так же можем зайти в меню "Параметры сортировки" — это настройки таблицы кодировок. Выполнять сортировку, как учитывать верхний и нижний регистр, как реагировать на символы, и т.п.

Далее переходим к настройкам учетных записей для подключения к SQL Server 2019. Нам предлагают выбор режим входа под учетными записями Windows, либо смешанный режим, то есть возможность входа под учетной записью Windows и под учетной записью SQL Server, если выбрать смешанную, то вам предлагается создать учетную запись SQL Server.

На рисунке 3.6 у вас на выбор есть 2 режима: **Windows authentication mode** и **Mixed mode**. **Windows** authentication mode авторизоваться смогут только пользователи вашего домена или компьютера под управлением Windows. В **Mixed** mode помимо windows авторизации станет доступна авторизация по учетным данным самого SQL Server’a. Мы выберем **Windows authentication mode.**

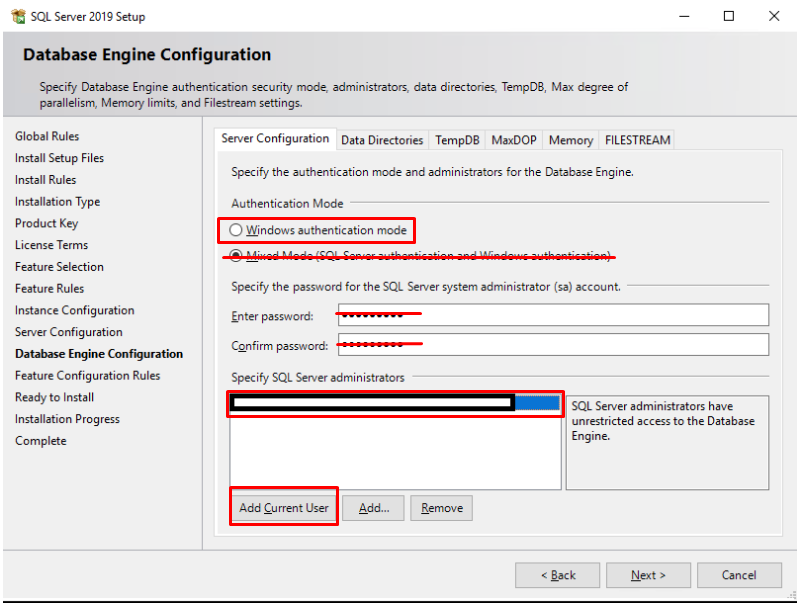


Рисунок 3.6 – Настройка ядра СУБД

Теперь в следующем окне можно наблюдать все установки и настройки, которые вы ранее задавали. Нажимаем «Установить», как показано на рисунке 3.7.

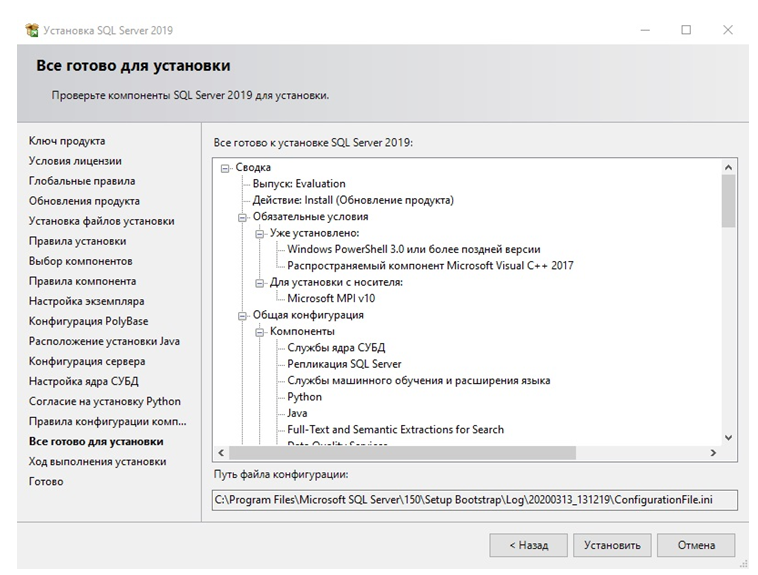


Рисунок 3.7 – Установка

Если все прошло успешно, закрываем окно. После того, как установка SQL Server 2019 завершена, нам нужно установить приложение, с помощью которого мы будем подключаться к серверу баз данных. Это приложение SQL Server Management Studio (SSMS).

Заходим снова в центр установки SQL Server и наживаем "Установить средства управления SQL Server", как показано на рисунке 3.8.

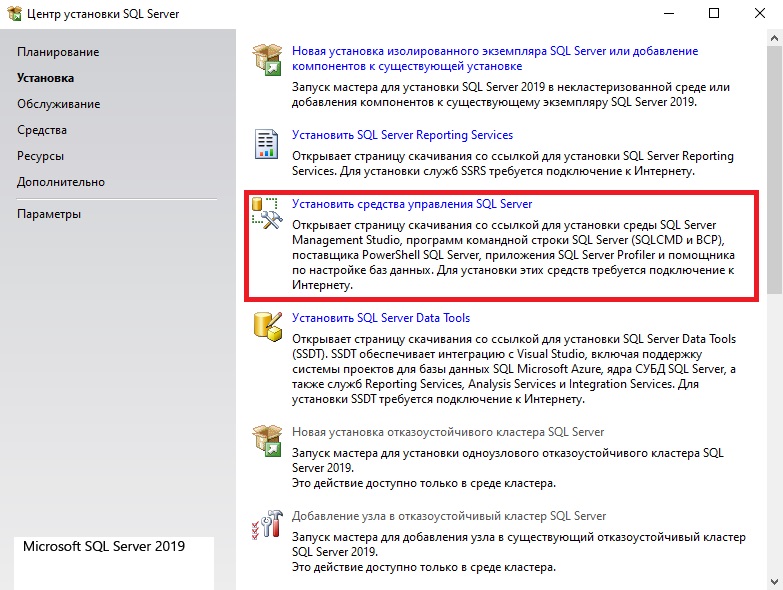


Рисунок 3.8 – Установка SQL Server Management Studio

При нажатии у нас откроется [сайт Microsoft](https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15) и нам нужно будет скачать SSMS. Нажимаем "Установить", как показано на рисунке 3.9.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.9 – Установка SQL Server Management Studio

И дожидаемся окончания установки. Теперь через поиск ищем приложение «Microsoft SQL Server Management Studio». Теперь Вы можете авторизоваться под вашей учетной записью в SQL Server, как показано на рисунке 3.10.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.10 – Соединение с сервером

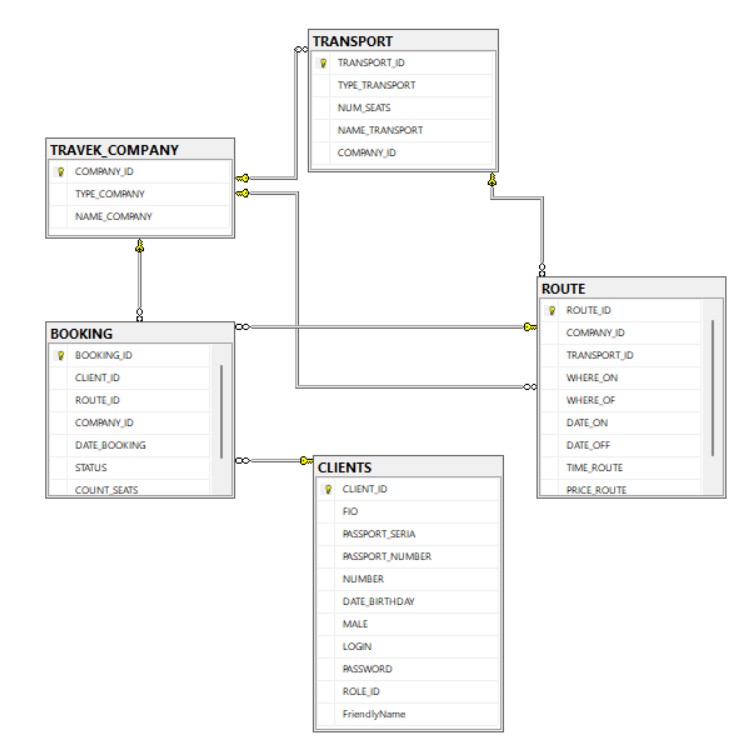


Рисунок 3.11 – Реляционная схема базы данных

База данных состоит из пяти сущностей, где CLIENTS – клиенты, ROUTE – маршруты, TRANSPORT – транспорт, TRAVEK\_COMPANY – транспортные компании и BOOKING – брони.

3.3 Инструкция по эксплуатации приложения

Текст

4 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [10].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота — 72,5 см, ширина — 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [11].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных “В”. При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [12].

Заключение

Разработанное клиент-серверное приложение «Бронирование авиа и ЖД билетов» позволяет быстро и эффективно работать с данными в данной предметной области. Данное приложение может быть использовано любой организацией, осуществляющей посредническую деятельность в сфере букинга билетов. Пользователями системы могут быть кто угодно – руководитель, бухгалтер, отдел кадров, заинтересованное лицо и так далее.

Достоинством приложения является простота и наглядность использования, также то, что оно является настольным (desktop) приложением. Приложение можно использовать для любых компаний в сфере букинга.

Разработанное клиент-серверное приложение можно доработать, добавив алгоритм поиска выгодных билетов по маршруту, фотографии на каждый транспорт, информацию о пересадках и т. д.

Список источников

1. <https://studentopedia.ru/informatika/pervij-etap-razvitiya-yazikov-programmirovaniya-1940---1960-gg---mashinnie-kodi-i-assembler--.html>
2. <https://ru.wikipedia.org/wiki/История_языков_программирования>
3. <https://info-comp.ru/programmirovanie/739-install-visual-studio-2019-community.html>
4. [www.aviasales.ru/](http://www.aviasales.ru/)
5. [WPF | Страничные интерфейсы (professorweb.ru)](https://professorweb.ru/my/WPF/UI_WPF/level24/24_2.php)
6. https://studfile.net/preview/16467497/page:21/
7. <https://studfile.net/preview/1362051/page:21/>
8. https://elib.bsu.by/bitstream/123456789/48524/5/Защита%20информации%20в%20БД.pdf

Приложение А

Программный код