РЕФЕРАТ

Дипломная работа по теме «Создание информационной системы для отеля «Виктория» в г. Ачинске» содержит 53 страницы текстового документа, 20 использованных источников, 30 рисунков.

ИНФОРМАЦИОННАЯ СИСТЕМА, СТРУКТУРНАЯ МОДЕЛЬ, УПРАВЛЕНИЕ, АУТЕНТИФИКАЦИ, АВТОРИЗАЦИЯ, БАЗА ДАННЫХ, ТЕХНИЧЕСКОЕ ЗАДАНИЕ.

Цель дипломной работы - реализация информационной системы, позволяющей управлять гостиничным комплексом с наименьшими затратами денежных средств.

В результате работы была разработана информационная система обеспечения работы гостиницы, которая соответствует специфике работы гостиничного комплекса и спланирована дальнейшая ее адаптация под сеть таких организаций.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 Специфика гостиничного бизнеса 5
1.1 Введение в теорию гостиничного бизнеса 5
1.2 Срочный характер гостиничных услуг 7
1.3 Вывод по специфике гостиничного бизнеса 8
2 Выбор языка программирования 10
2.1 Скорость разработки 10
2.2 Кроссплатформенность 11
2.3 Производительность кода и требовательность к ресурсам 11
2.4 Библиотеки 12
2.5 Удобство отладки 14
2.6 Язык и Синтаксис 14
2.7 Стоимость поддержки 15
2.8 Самодостаточность приложений 15
2.9 Удобство сборки 16
2.10 Выводы 16
3 Выбор среды разработки для базы данных 19
4 Разработка базы данных 23
5 Разработка структурной схемы информационной системы и ее дальнейшая реализация 27
ЗАКЛЮЧЕНИЕ 51
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 52

ВВЕДЕНИЕ

Автоматизированное управление организацией является неотъемлемой частью современной концепции руководительской деятельности. В наше время редко возможно найти организацию, которая хотя бы частично не была бы автоматизирована. Хранение базы клиентской , документов и информации необходимой для устойчивой работы компании требует существенных денежных затрат и временных .

Цель дипломной работы - реализация информационной системы, позволяющей управлять гостиничным комплексом с наименьшими затратами денежных средств.

Для достижения цели сформулированы следующие задачи:

– изучение специфики гостиничного бизнеса;
– выбор языка программирования;
– выбор среды разработки базы данных;
– разработка базы данных;
– разработка структурной схемы информационной системы;
– разработка программного обеспечения;
– тестирование и отладка программного обеспечения;
– анализ итогов внедрения информационной системы.

1 Специфика гостиничного бизнеса

1.1 Введение в теорию гостиничного бизнеса

Многие рассматривают понятие «гостиница» как производное от латинского «hospitalis», что означает «гостеприимный». По поводу происхождения этого понятия сегодня существуют множество мнений. Одна из распространенных точек зрения сводится к тому, что слово английское «hospitality» (гостеприимство) происходит от старофранцузского «hospice» (оспис), что означает дом «гостеприимный ». Старейшим из учреждений такого рода является Hospice de Beaune в Бургундии, известное также как Hotel Dieu - «Дом Бога». Было оно основано в 1443 году как больница и приют для бедняков. Эта больница функционирует и в настоящее время.

В отечественной практике управления, сохранившейся до времени настоящего, слово «hospitality» связывают прежде всего с понятием «госпиталь», означающим военно-медицинское учреждение для стационарного лечения.

С точки зрения смысла здравого, о каком-либо гостеприимстве не может быть и речи без удовлетворения потребностей первичных человека - потребности в питании, отдыхе и сне. В этой связи наиболее обоснованным и достаточно логичным является следующее определение гостиницы.

Гостиница – это предприятие, представляющее людям, находящимся вне дома комплекс услуг, важнейшими среди которых (комплексообразующими) в равной степени являются услуга размещения и питания.

В данном определении указывается на обязательное наличие двух главных услуг в гостинице - размещения и питания. При этом их соотношение между собой сильно может сильно различаться.

Содержание услуги размещения состоит в том, что, во-первых, в пользование предоставляются специальные помещения (гостиничные номера); во-вторых, предоставляются услуги, выполняемые непосредственно

персоналом гостиницы: портье по приему и оформлению гостей, горничными по уборке гостиничных номеров и т.д.

Такие тенденции связаны, с активным развитием отечественной туристической сферы, а также с появлением новых категорий постояльцев гостиниц.

И так, для полноценной работы гостиницы просто необходима высококлассная координация и связь различных групп служб, иначе возникающие у клиента проблемы не будут решаться оперативно и без вмешательства высшего руководства отеля, что, естественно, отразится на качестве организации. Как правило, все свои претензии клиенты высказывают службе приема, которая, однако, не всегда непосредственно должна исправлять допущенные ошибки. Помещение гостиницы содержит в своих стенах представителей различных профессий (от 5 до 9 специальностей). При этом практически каждая услуга, которая оказывается клиенту, требует совместной работы нескольких служб, координации действий работников данных специальностей.

Если гость недоволен уборкой номеров, то сделать все как следует должен персонал из службы фонда номерного . Так, необходимо, чтобы представители различных служб без проблем могли связаться друг с другом. Без налаженного взаимодействия подобную проблему быстро решить не удастся.

Содержание услуги размещения состоит в том, что, во-первых, в пользование предоставляются специальные помещения (гостиничные номера); во-вторых, предоставляются услуги, выполняемые непосредственно персоналом гостиницы: портье по приему и оформлению гостей, горничными по уборке гостиничных номеров и т.д.

Неодновременность процессов производства и потребления в отношении предоставляемого в гостинице комплекса услуг не в полной степени соответствует такая общая для услуг характеристика, как одновременность, неразрывный характер процессов производства и потребления. Отдельные

гостиничные услуги не связаны с присутствием клиента. Например, уборка и подготовка номера к продаже не совпадает по месту и времени с моментом продажи номера в службе рецепции и непосредственным заселением клиента.

Гостиничные номера являются основным элементом услуги размещения. Это многофункциональные помещения, предназначенные для отдыха, сна, работы проживающих гостей. Например, в гостиницах делового назначения очень важной функцией номеров является обеспечение гостю возможности поработать, то есть здесь необходим письменный стол, телефон, факс, компьютер и т.д. Вследствие того, что гостиничные номера используются гостями преимущественно в вечернее и ночное время, важнейшей их функцией является обеспечение возможности сна. Значимость других функций гостиничных номеров, прежде всего, зависит от назначения гостиницы и потребностей гостей.

1.2 Срочный характер гостиничных услуг

Отличия гостиничных предприятий состоит в том, что проблемы, касающиеся обслуживания, должны решаться быстро. Для решения этой проблемы на многих гостиничных предприятиях введены стандарты обслуживания - комплекс обязательных для исполнения правил обслуживания клиентов, которые призваны гарантировать установленный уровень качества всех производимых операций.

Критерии стандартов обслуживания в гостиничном комплексе:

– время ответа на звонок о получении информации или бронировании (15, 20, 30 секунд);

– время оформления в службе размещения. Независимо от массовости заезда клиенты не должны стоять в очереди более установленного на конкретном гостиничном предприятии времени (5,10,15 минут);

– время, затрачиваемое на оказание конкретной услуги. Например, багаж доставляется в номер не позднее 3 минут после заселения клиента,

стирка и чистка личных вещей проживающих укладывается в сутки и т.д.;

– внешний вид и наличие униформы;

– знание иностранных языков персоналом обслуживания.

Данный фактор наряду с месторасположением является наиболее значимым при выборе гостиницы. Время предоставления ряда гостиничных услуг измеряется даже секундами.

Важной особенностью гостиничных услуг, отличающей их, прежде всего от услуг промышленности, где в большей степени используются машины и автоматы, является широкое участие людей в производственном процессе. Вовлечение человеческого фактора оказывает сильное влияние на неоднородность, изменчивость качества и связанный с ними недостаток стандартизации, что представляет одну из важнейших проблем в сфере развития гостиничных услуг.

1.3 Вывод по специфике гостиничного бизнеса

В соответствии с вышеприведенными спецификациями, можно сделать вывод, что информационная система, используемая в гостиничном деле должна обеспечивать:

– полноту и достоверность содержащей в себе информации;

– скорость работы, соответствующую критериям обслуживания;

– связь между отдельными категориями служащих организации;

– автоматизацию передаваемой информации дочерним службам.

Для того чтобы информационная система всегда содержала в себе достоверную и полную информацию необходимо, контролировать правильность и корректность, вводимых оператором информационной системы данных, своевременную их корректировку и уточнение.

Быстрота работы зависит как от быстроты выполнения запросов к базе данных, так и от навыков работы оператора информационной системы. Для

убыстрения обучения оператора необходимо чтобы информационная система имела подсознательно понятный интерфейс, а так же корректно составленное руководство пользователя. В зависимости от занимаемой должности программное обеспечение будет выдавать тот функционал, который необходим для выполнения своих обязательств, это так же поможет в скорости работы оператора информационной системы.

Для организации связи между отдельными категориями служащих, для поручения определенных заданий и контролем их исполнения, для данной информационной системы предлагается разработать модуль, который автоматизирует передачу заданий непосредственно с места оператора информационной системы.

2 Выбор языка программирования

На первичном этапе разработки приложения встал вопрос, какой из известных языков программирования использовать, C++ или C#?

Для того чтобы ответить на возникший вопрос, необходимо сравнить их по следующим параметрам:

- скорость разработки;
- кроссплатформенность;
- производительность кода и требовательность к ресурсам;
- библиотеки;
- удобство отладки;
- язык и синтаксис;
- стоимость поддержки;
- риски;
- самодостаточность приложений;
- удобство сборки;
- перспективы;
- вывод.

2.1 Скорость разработки

C# позволяет стартовать разработку быстрее, что, в свою очередь, позволяет быстрее получить начальное решение. Скорость разработки на C# на начальных этапах проекта значительно выше по сравнению с C++.

Однако, когда остов проекта создан, основные подходы и библиотеки выбраны, быстрота разработки на C++ и стремительность разработки на C# становятся практически одинаковыми.

Таким образом, в недолгих малобюджетных проектах C# будет обладать преимуществом по скорости разработки, но в длинных и сравнительно дорогих

данное преимущество будет незначительным.

2.2 Кроссплатформенность

C++ кроссплатформенный, хотя и с отдельными оговорками, дополнительными затратами.

C#, по факту, оказался не кроссплатформенный, несмотря на существование частных .net окружений под разными платформами.

И хотя C# может использовать для построения приложений под не-Windows платформы, проблемы, рождаемые использованием .net в не-Windows окружении, сводят на нет многие преимущества предпочтения C#. При этом для разработки на C++ сформировалась практически равнозначная инфраструктура на большинстве имеющихся платформ, есть множество библиотек, которые скомпилированы или могут быть скомпилированы под любые имеющиеся платформы. Поэтому советовать его для кроссплатформенного использования можно разве что если код на C# уже написан. При этом нужно четко понимать, что в будущем это будет приносить дополнительные затраты на помощь.

Именно поэтому существует немалое количество кроссплатформенных приложений и библиотек различного масштаба, написанных на C++, на ряду с кроссплатформенными библиотеками есть и библиотеки специфические для некоторых платформ. Все это дает практически равнозначные шансы для развития приложений на разных платформах.

2.3 Производительность кода и требовательность к ресурсам

Несомненным является факт того, что потенциалы по оптимизации неуправляемого кода куда шире, чем возможности по оптимизации управляемого кода. Таким образом, пиковая продуктивность кода достижима только в неуправляемом исполнении, т.е. в пределе, почти любая задача на C++

может быть решена с наименьшими требованиями к ресурсам. Фундаментальные основы преимуществ C++ в возможности писать код, который будет выполняться непосредственно процессором, и возможности прямой работы с памятью. Поэтому в трудных задачах, связанных с обработкой немалого количества данных, C++ имеет сильные преимущества перед C#.

Но стоит понимать, что при выборе ошибочного подхода, на C++ вполне можно написать код, который будет работать медлительнее кода на C#, осуществляющего туже задачу.

Конечно, свобода дает больше возможностей создать себе проблемы, но в ряде случаев это лучше, чем невозможность преодоления потолка производительности. Однако это не значит, что продуктивный код на C++ обязательно будет страшным или сложным для восприятия, просто при его написании понадобится более оригинальный подход для удовлетворения перечисленных мер одновременно И этот потолок вполне может привести, например, к тому, что под решение. Задачи, для которого бы хватило одного хорошего сервера, вам придется собирать ферму из нескольких серверов, или же к тому, что ваше приложение будет требовать мощной аппаратной части на задачи, для которых хватило бы составляющих выпущенных 7-10 лет назад.

Если говорить о совокупности субъективных «простоты разработки», «красоты кода» и объективной производительности, то используя C# легче написать код, ублаготворяющий этим мерам одновременно.

2.4 Библиотеки

Отличие ассортимента C++ и C# библиотек в том, что C++ библиотек больше, они имеют большую историю, за которую стали неплохо отлажены и оптимизированы, часто кроссплатформенные, многие с открытым кодом. При всем том, при всех положительных сторонах C++ библиотеки как имеют очень разную, часто даже архаичную структуру, часто не объектный, а структурно-

процедурный интерфейс. Связано это с тем, что многое в C++ - это C библиотеки.

Однако не все так страшно, для C++ есть немало и «красивых» библиотек. Да и основные библиотеки постоянно расширяются.

В C# перечисленных выше проблем существенно меньше. Немалое количество библиотек с .net идет в базе, плюс к ним масса свободно доступных библиотек, это покрывает практически все основные задачи разработки под Windows. Наличие немалого количества шаблонных типов почти избавляет от библиотек, где базовые типы переопределены. И в силу того, что библиотеки C# сравнительно молодые, интерфейсы библиотек, как правило, лучше вписываются в те или иные шаблоны проектирования, что часто упрощает их изучение.

Однако же, при ближайшем разборе высок шанс, что под вашу специфическую задачу C# библиотеки не окажется, более того, может оказаться, что и решать такую задачу на C# достаточно не эффективно, поэтому подобной библиотеки не появится и в будущем, а если и появится, то будет работать недостаточно быстро.

Другая неприятная особенность C++ библиотек — это создание и переопределение своих базовых типов. Многие C++ библиотеки заводят свои типы строк, контейнеров, переопределяют некоторые базовые типы. Этому есть логичные объяснения (лучшая производительность, поддержка кроссплатформенности, отсутствие подходящих типов на момент написание библиотеки), однако все это не добавляет удобства использования и красоты коду. Базовые же C++ библиотеки дают не так много, как дают стандартные библиотеки C#, поэтому подбор правильных библиотек для проекта C++ - это задача, необходимая даже в сравнительно простых проектах.

Вторая неприятная особенность библиотек C# в том, что многие из них являются просто оберткой над неуправляемыми библиотеками, что будет всегда приводить к потерям производительности на конверсиях типов, и создавать дополнительные проблемы отладки и распространения.

2.5 Удобство отладки

Под Windows C# в большей степени удобней отлаживать, в отличие от C++.

Однако если по какой-то причине у вас на ряду с управляемым кодом из C# сборки используется неуправляемый, то его отладка станет более сложная по сравнению с обычной отладкой неуправляемого кода из C++.

В C#, как и в других появившихся до .NET языках, главная методика по отладке состоит в добавлении точек останова и изучении того, что происходит в коде в конкретные моменты во время его выполнения.

2.6 Язык и Синтаксис

Синтаксис C#, пожалуй, можно назвать упрощенной версией C++, таким образом C#, как и любое упрощение, одновременно несет и позитивный и негативный эффекты.

С первого взгляда код C++ и C# очень похож внешне. Но многообразие кода на C++ больше, ведь C++ является одновременно и C и C++ и C++0x и все это вы можете использовать одновременно.

C# же, это только C#, хотя его синтаксис постоянно расширяется. Код на C#, как правило, выглядит проще и лаконичнее, чем код C++. Языковые конструкции C++ и C# очень схожи, однако существенные различия можно найти в деталях.

Стоит сказать, что более сложный код часто легче пишется и анализируется, если написан более простым языком. С этой позиции, используя C#, меньше шансов допустить ошибку в принципиально сложном коде и больше шансов написать чистый код, обладая теми же ресурсами. Это может быть полезно при решении достаточно сложных, но не требовательных к производительности задач. Однако при этом большее количество «синтетики» в C# делает меньше оценку производительности кода по его «внешнему виду».

Если C++ можно упрекнуть за отсутствие сборки мусора. То C# нужно упрекнуть за отсутствие полноценных деструкторов, отсутствие полноценных макросов, достаточно грубую настройку наследования, отсутствие константных методов и членов, отсутствие глобальных методов (процедур), очень ограниченную поддержку шаблонов, список можно продолжать… Однако жить без всего этого вполне можно как в случае C++, так и в случае C#.

2.7 Стоимость поддержки

В поддержке приложений большой разницы между C++ и C# нет. Хотя стоит понимать, что некоторые баги в приложениях, написанных на C#, средствами .net, исправить невозможно и при необходимости их исправить стоимость поддержки может существенно возрасти. Однако, если говорить о переструктурировании, то зачастую приложения, написанные на C#, реструктурировать несколько дешевле.

2.8 Самодостаточность приложений

Полной самодостаточности приложений нет ни у C++ ни у C#. Для C++ так или иначе нужен runtime, а для C# .net framework.

Однако хотелось бы отметить, что runtime C++, как и любая другая библиотека, может быть статически линкована в исполняемый модуль, таким образом исполняемый модуль может содержать все необходимое для работы, и за счет чего станет самодостаточным, в случае C# такое, стандартными средствами не реализуемо.

2.9 Удобство сборки

Сборка C++ проектов заметно сложнее сборки проектов C#. Однако стоит понимать, что большая сложность предоставляет и дополнительную гибкость, которая рано или поздно может стать полезной вам. Однако до этого момента будет лишь увеличивать расходы вашего времени.

2.10 Выводы

C#, являясь последним из широко распространенных языков программирования, должен впитать в себя весь имеющийся опыт и вобрать лучшие стороны существующих языков программирования, при этом являясь специально созданным для работы в .NET. Сама архитектура .NET продиктовала ему (как и многим другим языкам, на которых можно писать под .NET) объектно-ориентированную направленность. Конечно, это не является правилом, возможно создание компиляторов даже функциональных языков по .NET, на эту тему существуют специальные работы.

Свой синтаксис C# во многом унаследовал от C++ и Java. Разработчики, имеющие опыт написания приложений на этих языках, найдут в C# много знакомых черт. Но вместе с тем он является во многом новаторским - атрибуты, делегаты и события, прекрасно вписанные в общую идеологию языка, прочно заняли место в сердцах .NET - разработчиков. Их введение позволило применять принципиально новые приемы программирования.

Из вещей, включенных в спецификацию языка, но не являющихся чисто «программистскими» необходимо отметить возможность использование комментариев в формате XML. Если комментарии отвечают специально описанной структуре, компилятор по ним может сгенерировать единый XML-файл документации.

Но C# внес и свои уникальные черты, которые уже были упомянуты - это события, индексаторы, атрибуты и делегаты. Все эти элементы будут

обсуждены в следующих частях, сейчас лишь отмечу, что они предоставляют собой очень полезные возможности, которые не останутся невостребованными.

Конечно, излюбленным объектом для сравнения с C# у мировой коммьюнити является Java. Также разработанный для работы в виртуальной среде выполнения, имеющей объектно-ориентированную архитектуру и сборщик мусора, основанный на механизме ссылок. При сравнении с этим языком сразу выделаются такие особенности, как возможность объявлять несколько классов в одном файле, из чего следует синтаксическая поддержка иерархической системы пространств имен. Из реализации ООП-концепций сходство в механизме наследования и реализации (и в Java и в C# возможно единичное наследование, но множественная реализация интерфейсов, в отличие от C++). Но в Java отсутствуют свойства и индексаторы (а также делегаты и события, но они отсутствуют еще много где). Также есть возможность перечисления контейнеров.

Разрабатывать информационную систему планируется под платформу Windows, следовательно кроссплатформенность для нас не важна. Необходимо создать приложение, которое будет иметь дружелюбный для пользователя интерфейс, средствами C# этого гораздо легче и быстрее добиться. Синтаксис C# на много понятнее человеку, не имевшему отношение к разработке программы, следовательно, легче найти человека на доработку и поддержку программного обеспечения. Так же библиотеки на C#, считаю более адаптированными для работы с базами данных, следовательно, легче и быстрее будет связать базу данных и разрабатываемое приложение. Итогом данного раздела считаю выбор языка программирования C# и среды разработки Microsoft Visual Studio 2012.

Ввиду очень удобного объектно-ориентированного дизайна, C# является хорошим выбором для быстрого конструирования различных компонентов - от высокоуровневой бизнес логики до системных приложений, использующих низкоуровневый код. Также следует отметить, что C# является и Web ориентированным - используя простые встроенные конструкции языка ваши

компоненты могут быть легко превращены в Web сервисы, к которым можно будет обращаться из Internet посредством любого языка на любой операционной системе. Дополнительные возможности и преимущества перед другими языками приносит в C# использование передовых Web технологий, таких как: XML (Extensible Markup Language) и SOAP (Simple Object Access Protocol). Среда разработки Web сервисов позволяет программисту смотреть на существующие сегодня Web приложения, как на родные C# объекты, что дает возможность разработчикам соотнести имеющиеся Web сервисы с их познаниями в объектно-ориентированном программировании.

3 Выбор среды разработки для базы данных

Для взаимодействия пользователя с БД используются системы управления базами данных (СУБД), которые обеспечивают:

– набор средств, для поддержки таблиц и соотношений между ними;

– развитый пользовательский интерфейс, позволяющий вводить и модифицировать информацию, производить поиск и представлять результаты;

– средства программирования высокого уровня, позволяющие создавать собственные приложения.

Подход к построению СУБД значительно видоизменялся на протяжении почти 40 лет. На смену вычислительным центрам предприятия и АСУП на их основе пришли персональные компьютеры и настольные (персональные) системы управления базами данных, затем с развитием коммуникаций появились распределенные системы и концепции управления крупными предприятиями - корпорациями на основе бизнес-процессов.

При этом в условиях динамичного изменения бизнес-процессов в последние годы сформулировался ряд определенных требований к функциональным возможностям тех СУБД, производители которых стремятся поддерживать свои продукты на высоком, конкурентоспособном уровне.

Создаваемые средствами СУБД приложения должны обладать высокой степенью мобильности и легко переноситься на разные компьютерные и сетевые платформы.

Перечисленные требований к СУБД, как к интегрирующим звеньям информационных систем новой архитектуры, позволяют взглянуть на существующие ныне на рынке продукты разных производителей под соответствующим углом зрения. Адекватность предлагаемых сегодня СУБД новым требованиям определит для их будущих владельцев и клиентов преимущества, создаваемых ИС, их гибкость, мобильность, возможность легкой перестройки и, в конечном счете, способность к выживанию.

Средства СУБД должны допускать возможность гибкого варьирования

архитектуры различных ИС для соблюдения разумного компромисса при разделении функциональных возможностей системы между рабочими станциями клиентов и серверами.

Создание «менеджеров процессов» может быть эффективным только в таких условиях, когда средства программирования СУБД объектно-ориентированы и возможно создание стабильных приложений при динамичном изменении маршрутизации сквозь эти задачи.

Производителям СУБД следует обеспечить соответствие поставляемых ими продуктов открытым стандартам взаимодействия.

Расширение бизнес-процессов за пределы одной компании и необходимость создания глобальных информационных связей выдвигает серьезную задачу поддержки высокой степени готовности систем, работающих 24 часа в сутки все 365 дней в году.

Коммуникационный обмен данными становится асинхронным, а информационные процессы длительными, и поэтому возникает необходимость журнализации состояния баз данных и проведения возможного отката/восстановления для расширенных временных рамок (дни, недели).

Адекватность предлагаемых сегодня СУБД новым требованиям определит для их будущих владельцев и клиентов преимущества, создаваемых ИС, их гибкость, мобильность, возможность легкой перестройки и, в конечном счете, способность к выживанию.

Противоположностью настольной системе является распределенная база данных, т. е. такая архитектура, при которой ядро БД работает на выделенном сервере; там же обычно хранятся и данные. Через локальную или глобальную сеть пользователь посредством установленного на своем компьютере программного обеспечения посылает запросы и получает ответы. Такие системы предназначены для работы с большим количеством клиентов, и зачастую в качестве серверов в них функционируют компьютеры более сложные и мощные, чем PC. В целом БД MS Access представляет собой многокомпонентный объект, который может включать кроме таблиц отчеты,

запросы, формы, макросы и модули. Все эти объекты, как правило, хранятся в одном файле, имеющем расширение *.mdb или *.mde. На практике это значительно упрощает резервное копирование данных. Такая технология носит название «клиент-сервер».

Так как на первоначальном этапе планируется внедрение информационной системы на единичный объект, логично использовать локальную базу данных. Возможно, придется переносить программу на другие ПК, следовательно, необходимо использовать СУБД, которая располагает все объекты, которыми она оперирует в одном файле. Необходим удобный визуальный редактор, который позволит быстро выполнять SQL запросы, а так же создания данных SQL запросов по средствам конструктора, для внесения их на клиентскую часть БД.

Программное обеспечение для работы с базами данных используется на персональных компьютерах уже довольно давно. К сожалению, эти программы либо были элементарными диспетчерами хранения данных и не имели средств разработки приложений, либо были настолько сложны и трудны, что даже хорошо разбирающиеся в компьютерах люди избегали работать с ними до тех пор, пока не получали полных, ориентированных на пользователя приложений.

Вышесказанное не означает, что MS Access способен выполнять операции только в однопользовательском режиме. Напротив, программа успешно работает и в сетях, правда, с ограниченным числом клиентов, и при этом не используется технология «клиент-сервер». Сама компания Microsoft настоятельно рекомендует инсталлировать MS Access в рамках небольших рабочих групп. Что касается легкости использования, то Microsoft Access совершил здесь настоящий переворот, и многие для создания своих собственных баз данных и приложений обращаются именно к нему.

Microsoft Access - это функционально полная СУБД. Программный продукт MS Access представляет собой настольную систему управления базами данных. Понятие «настольная» СУБД указывает на то, что все операции с базой данных осуществляются на локальном компьютере пользователя.

В ней предусмотрены все нужные вам средства для определения и обработки данных, а также для управления ими при работе с большим количеством информации. Система управления базами данных предоставляет вам возможность контролировать задание структуры и описание своих данных, работу с ними и организацию коллективного пользования этой информацией. СУБД также существенно увеличивает возможности и облегчает упорядочивание и ведение большого количества хранящейся в многочисленных таблицах информации. Именно здесь находится физическое место хранения информации, а также работают средства управления и организации запросов.

Система управления базами данных включает в себя три основных типа функций: определение (задание структуры и описание) данных, обработка данных и управление данными. Все эти функциональные возможности в полной мере в Microsoft Access. В практике, обычно, необходимо решать и задачи с использованием электронных таблиц и текстовых процессоров. Например, после подсчета или анализа данных необходимо их представить в определенном виде или шаблоне. В итоге пользователю приходится использовать несколько программных продуктов для достижения необходимого результата. В этом смысле все существенно упростят возможности, предоставляемые Microsoft Access.

Вышеприведенным требованиям соответствует СУБД Microsoft Access. В данной работе решено использовать Microsoft Access 2010.

4 Разработка базы данных

Разработка базы данных была начата с общей концепции. Для информационной системы гостиницы необходимо иметь в базе данных следующую информацию:

– информацию о посетителях;

– информацию о сотрудниках;

– информацию о комнатах;

– информацию об учетных записях пользователей ИС;

– информацию о ценовых категориях на виды услуг;

– информацию о когда-либо забронированных номерах (Архив).

Информацию о посетителях содержит в себе таблица «Клиенты» и «Контакты_Клиента», их содержимое представлено на рисунке 1 и рисунке 2, соответственно.

Код	Фамилия	Имя	Отчество	Дата рожде	Пол	Вид докуме	Серия	Номер	Комментарий
1	Булавочкина	Александра	Вительевна	23.10.1939	жен	Паспорт	2546	999999	Любит выпить
2	Асташкин	Роман	Владимирови	16.10.1991	муж	Паспорт	1232	214785	Оставляет хорошие
3	Аверкина	Наталья	Викторовна	16.11.1992	жен	Паспорт	3698	563214	Оставляет чистоту в
4	Дряхлов	Евгений	Евгеньевич	07.01.1992	муж	Паспорт	6547	963258	Общительный челов
5	Орехова	Людмила	Витальевна	09.10.1992	жен	Паспорт	6547	123654	Любит покушать
6	Журавлев	Игорь	Евгеньевич	02.04.1992	муж	Паспорт	4569	369872	Оставляет хорошие
7	Данилин	Василий	Владимирови	30.07.1992	муж	Паспорт	3654	569234	Любит покушать
8	Крючкова	Ольга	Генадьевна	18.11.1991	жен	Паспорт	5698	218975	Любит выпить
9	Гарина	Маргарита	Юрьевна	18.05.1992	жен	Паспорт	2589	357951	Общительный челов
(№)									

Рисунок 1 – Таблица «Клиенты»

ID	Номер телефона	Адрес прож	Щелкните для добавления
1	+79232140201	Сатурная 3-45	
2	+79232140285	Вавилова 6-21	
3	+79232140286	Кравченко 7-8	
4	+79232140287	Крылова 7-12	
5	+79232140288	Калинина 185-	
6	+79232140289	Малиновского	
7	+79232140290	Маяковского 6	
8	+79232140214	Ломоносова 9	
9	+79232140224	Чернышева 8-	
(№)			

Рисунок 2 – Таблица «Контакты_Клиента»

В столбцах данной таблицы содержится информация, которая необходима для идентификации каждого клиента. Таблица «Клиенты» и «Контакты_клиента» связаны по идентификационному коду.

Информацию о сотрудниках гостиницы содержит таблица «Сотрудники», которая представлена на рисунке 3.

Код	Фамилия	Имя	Отчество	Дата рождения	Образовани	Должность	Адрес	Телефон	Заработная плата
1	Антонова	Анна	Игоревна	22.02.1999	высшее	старший admini	ул.Ленина 52а/1	+792321402	2500
2	Епифанов	Антон	Викторович	03.12.1968	высшее	завхоз	ул.Воронина 32	25698	1500
3	Суриков	Евгений	Арсеньевич	17.11.1954	среднее	охранник	ул.Острякова 16	24790	1000
4	Савельев	Алексей	Олегович	08.04.1976	среднее	охранник	ул.Толстого 12	25841	1000
5	Партнов	Иван	Петрович	15.10.1974	высшее	повар	ул.Аверина	26947	1500
6	Терлеев	Константин	Юрьевич	09.01.1969	высшее	повар	ул.Гоголя 45	25897	1500
7	Пивоварова	Ольга	Антоновна	24.08.1975	высшее	горничная	ул.Королева 23	26985	1500
8	Мамонова	Юлия	Владиславовн	04.11.1970	среднее	горничная	ул.Вишневского 54	21336	1500
9	Ильина	Мария	Александровн	12.11.1968	среднее	уборщица	ул.Кирова 14	21199	1250
10	Тулупов	Игнатий	Савельевич	05.07.1978	высшее	официант	ул.Миронова 15	21965	1250
11	Иващенко	Людмила	Степановна	14.08.1973	высшее	горничная	ул.Степаняна 12	24567	1500
12	Самойлова	Инесса	Васильевна	01.10.1969	среднее	горничная	ул.Хрюкина 39	20574	1500
13	Агафонова	Мария	Ивановна	06.05.1967	среднее	горничная	ул.Мира	23695	1500
14	Проскуров	Иван	Сергеевич	11.07.1971	высшее	официант	ул.Юмашеева	25794	1250
15	Тонгоног	Полина	Григорьевна	13.02.1972	высшее	официант	ул.Строительная 25	25436	1250
16	Розанова	Светлана	Самуиловна	04.04.1974	среднее	уборщица	ул.Кесаева 11	23366	1250
17	Борисов	Анатолий	Сергеевич	06.07.1978	высшее	бармен	ул.Гагарина 16	25889	1300
18	Любимвцев	Денис	Сидорович	29.01.1979	высшее	бармен	ул.Борисова 36	25698	1300
19	Костылев	Авдей	Муратович	18.06.1958	высшее	сантехник	ул.Ракетная 2	21145	1350
20	Архипов	Алексей	Петрович	17.10.1969	высшее	экспедитор	ул.Орехова 54	25986	1500
(№)									

Рисунок 3 – Таблица «Сотрудники»

В столбцах выше представленной таблицы, хранится информация о сотрудниках организации. Каждый сотрудник идентифицируется по персональному коду.

Информацию о комнатах содержит в себе таблица, представленная на рисунке 4.

Таблица «Номера» содержит в себе информацию с номером комнаты, датой заезда и выезда, а также маркером, который указывает, занят ли в данный момент номер.

Информация об учетных записях пользователей информационной системы, которая в дальнейшем будет использоваться для аутентификации и авторизации, содержится в таблице «Таблица Аутентификации», она, в свою очередь, содержит такие поля как:

– код – отображает личный идентификатор;

– логин;

– пароль;

– имя;

– должность;

– права, которые выдаются при авторизации.

Код	Категория	дата заселе	дата высел	Статус	Щелкните для добавления
1	одноместный			☐	
2	одноместный			☐	
3	одноместный			☐	
4	одноместный			☐	
5	одноместный			☐	
6	двухместный			☐	
7	двухместный			☐	
8	двухместный			☐	
9	двухместный			☐	
10	двухместный			☐	
11	полулюкс			☐	
12	полулюкс			☐	
13	полулюкс			☐	
14	полулюкс			☐	
15	полулюкс			☐	
16	люкс			☐	
17	люкс			☐	
18	люкс			☐	
19	люкс			☐	
20	люкс			☐	
21	Люкс			☐	
22	Люкс			☐	
(№)				☐	

Рисунок 4 – Таблица «Номера»

Данные о ценовых категориях на виды услуг содержатся в таблицах «Заказ завтраков в номер» и «Категории номеров», которые представлены на рисунке 5 и рисунке 6, соответственно.

Код клиент	Код номера	Количество персон	Время подачи	Стоимость завтрака на 1 персону
1	1	2	10:15	35,00€
2	2			
3	7	3	10:00	35,00€
4	9	2	10:30	35,00€
5	11	2	10:00	35,00€
6	13	2	11:05	35,00€
7	15	2	11:10	35,00€
8	17	2	11:30	35,00€
9	19	2	10:00	35,00€

Рисунок 5 – Таблица «Заказ завтраков в номер»

Код категор	Категория	Количество мест	Количнство комнат	Стоимость в сутки
1	Одноместный	2	1	350,00€
2	Двухместный	2	3	500,00€
3	Полулюкс	4	4	700,00€
4	Люкс	5	6	1 000,00€
(№)				

Рисунок 6 – Таблица «Категории номеров»

Так же в базе данных присутствует таблица, которая отображает архив заказанных номеров. Данная таблица носит название «Архив». В ней отображены такие поля как:

– номер снятой комнаты;
– имя забронировавшего человека;
– контактный телефон;
– дата заезда;
– дата выселения;
– сотрудник, принявший заявку.

Считаю, что данной информации достаточно, чтобы реализовать первый этап информационной системы.

5 Разработка структурной схемы информационной системы и ее дальнейшая реализация

Первым этапом разработки клиентской части программного обеспечения стало создание структурной схемы, которая представлена на рисунке 7.

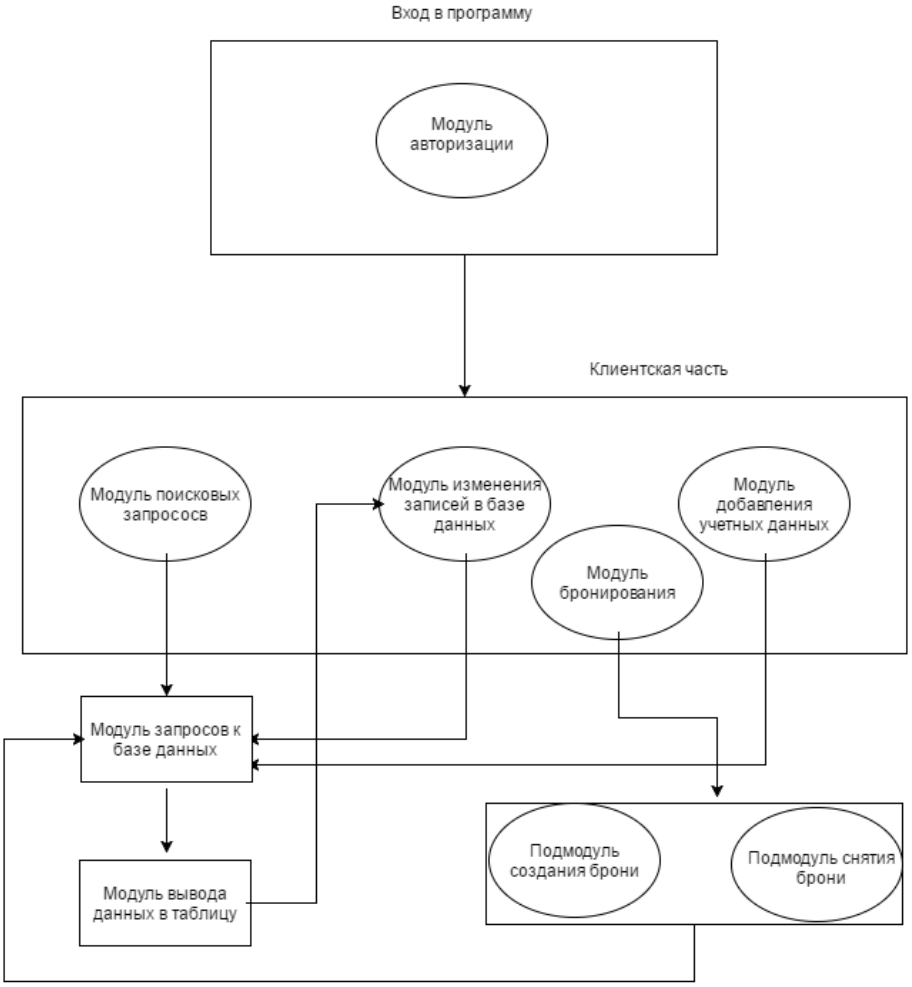

Рисунок 7 - Структурная схема клиентской части информационной системы

Модуль аутентификации – предназначен для сопоставления идентификационных данных и ключа аутентификации с последующей авторизацией. Реализовано это посредством сравнения данных вводимых пользователем с записями, имеющимися в базе данных, таблице «Таблица Аутентификации». Если находится хоть одно совпадение связки логин плюс пароль в базе данных и связки, вводимой пользователем, sql запрос возвращает

строку из таблицы «Таблица Аутентификации», содержащую логин, пароль, должность и права, соответствующие конкретному пользователю.

Участок кода, отвечающий за аутентификацию и авторизацию пользователя, представлен ниже:

```
private void button5_Click(object sender, EventArgs e)
        {   //строка запроса в базу данных
            string cmdtxt = "SELECT * FROM [ТаблицаАутентификации] WHERE [Логин] =" +
"\"" + textBox3.Text + "\"" + " AND [Пароль] =" + "\"" + textBox2.Text + "\"";
            //адаптер данных для запроса, где
connString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Гостинница1.mdb";
            OleDbDataAdapter dta3 = new OleDbDataAdapter(cmdtxt, connString);
           //создаем дата сет, куда запишем данные из запроса
            DataSet dt = new DataSet();
            //записываем данные по запросу
            dta3.Fill(dt);
//если есть хоть одна запись с в бд, соответствующая введеной связке логин+пароль
            if (dt.Tables[0].Rows.Count == 1)
            {   //Выводим сообщение добро пожаловать, ИМЯ_ПОЛЬЗОВАТЕЛЯ
MessageBox.Show("Добро пожаловать, " + dt.Tables[0].Rows[0][3].ToString() + "!");
                //Изменяем заголовок программы
                string str = this.Text = "Управление гостинницей, " +
dt.Tables[0].Rows[0][4].ToString() + " " + dt.Tables[0].Rows[0][3].ToString();
                //выдаем права в соответствии с записью в базе данных:
                //Добавление записей
this.добавитьЗаписьToolStripMenuItem.Enabled =
Convert.ToBoolean(dt.Tables[0].Rows[0][6].ToString());
                //Изменение записей
                ChangeFlag = Convert.ToBoolean(dt.Tables[0].Rows[0][5].ToString());
                //Управление учетными записями пользователей
                this.управлениеУчетнымиЗаписямиToolStripMenuItem.Visible =
Convert.ToBoolean(dt.Tables[0].Rows[0][8].ToString());
                //Удаление записей
                DeleteFlag = Convert.ToBoolean(dt.Tables[0].Rows[0][10].ToString());
                //Добавление комнат
                this.добавитьНомерToolStripMenuItem.Enabled =
Convert.ToBoolean(dt.Tables[0].Rows[0][11].ToString());
                //Добавление сотрудников
                this.добавитьСотрудникаToolStripMenuItem.Enabled =
Convert.ToBoolean(dt.Tables[0].Rows[0][9].ToString());
                //Бронь
```

```
                        this.броньToolStripMenuItem.Enabled =
Convert.ToBoolean(dt.Tables[0].Rows[0][7].ToString());
                        //Скрыть панель аутентификации
                        panel1.Visible = false;
                        this.textBox1.Select();
                        this.textBox1.ScrollToCaret();
                        this.AcceptButton = button2;     }
                    //Иначе вывести ошибку   }
```

Схема выполнения данного кода представлена на рисунке 8.

Выше представленный участок кода выполняется при нажатии кнопки «Вход», представленной на рисунке 9.

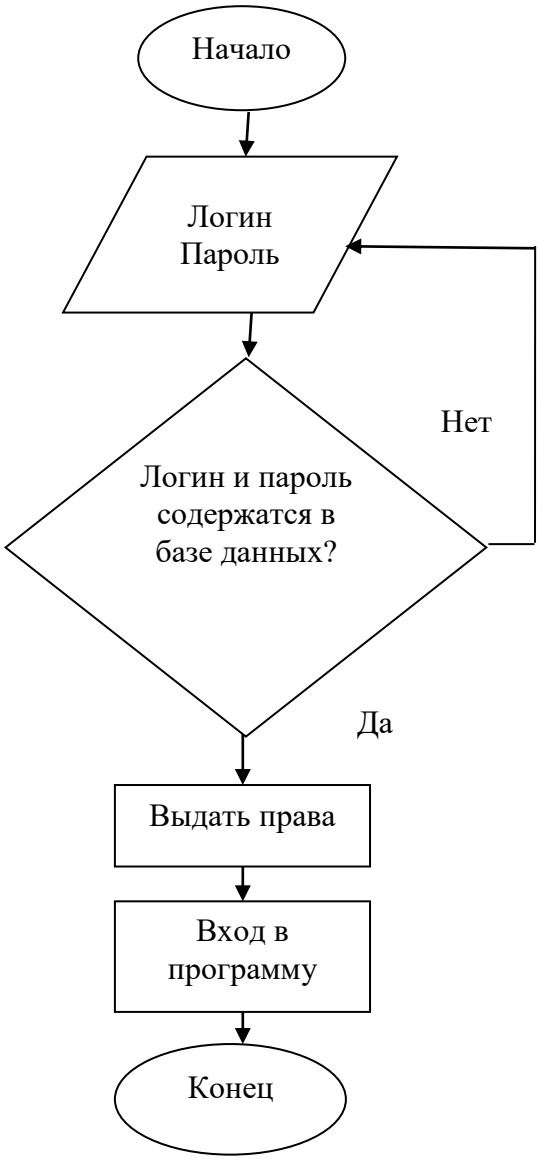

Рисунок 8 – Блок схема порядка выполнения входа в приложение

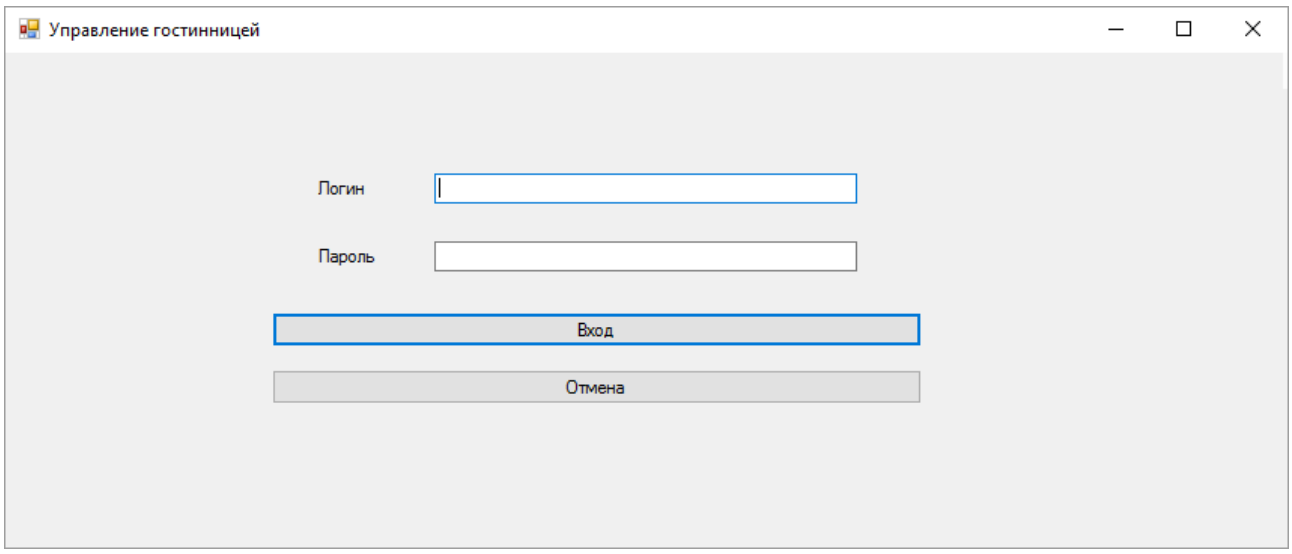

Рисунок 9 – Окно аутентификации

При нажатии кнопки «Отмена», представленной на рисунке 8, выполняется участок кода, который закрывает программу:

```
private void button4_Click(object sender, EventArgs e)
{            Application.Exit();        }
```

После входа в программу перед пользователем представляется основное окно программы, представленное на рисунке 10.

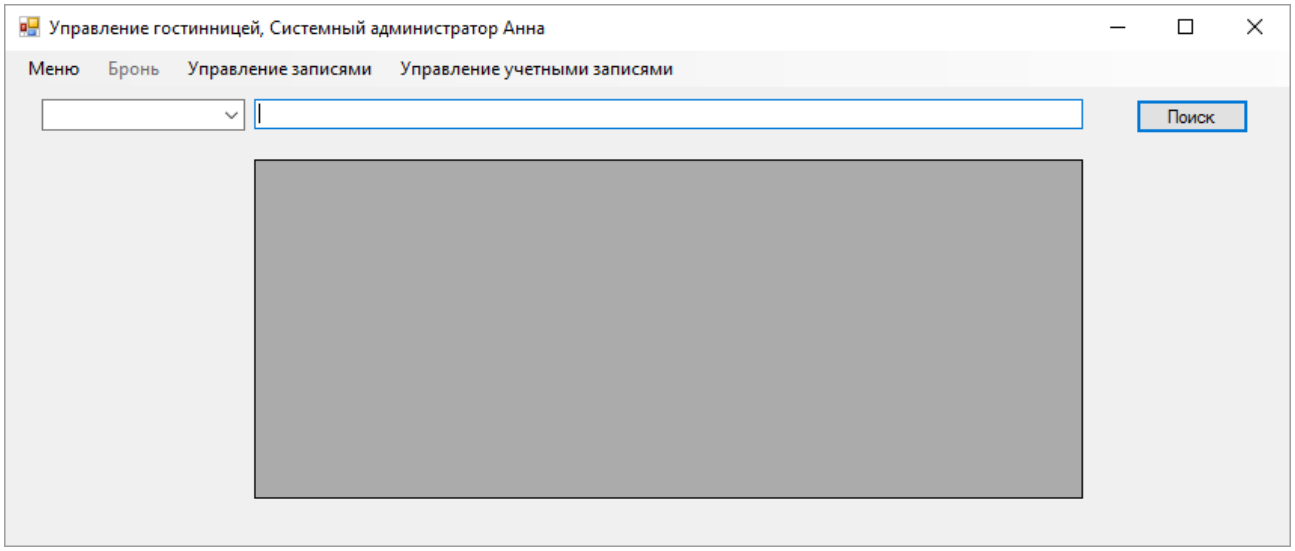

Рисунок 10 – Основное окно программы

В зависимости от прав, выданных для учетной записи пользователя, в окне, изображенном на рисунке 10, имеются такие пункты меню как:

– меню;

– бронь;

– управление записями;

– управление учетными записями.

Подменю, пункта меню содержит следующие функции:

– переподключиться;

– выход.

С помощью пункта «Переподключиться» имеется возможность сменить пользователя в информационной системе; за данную функцию отвечает следующий участок кода:

```
private void переподключитьсяToolStripMenuItem_Click(object sender, EventArgs e)
        {
        textBox2.Text = "";
            this.AcceptButton = button5;
            panel1.Visible = true;
            this.textBox3.Select();
            this.textBox3.ScrollToCaret();        }
```

Закрывается программа с помощью представленного ниже участка кода:

```
private void выходToolStripMenuItem_Click_1(object sender, EventArgs e)
        {            Application.Exit();        }
```

Пункт «бронь» позволяет перейти в подменю «Создание брони» и «Удаление брони».

Выбрав пункт «Создание брони» перед пользователем открывается диалоговое окно, которое позволяет забронировать номер. Пример такого окна представлен на рисунке 11.

Рисунок 11 – Диалоговое окно для создания брони

При нажатии кнопки «Сформировать» отправляется запрос к базе данных, результатом которого будет являться таблица свободных в данный момент номеров. За это отвечает участок кода, представленный ниже:

```
private void button1_Click(object sender, EventArgs e)
      {   string cmd = "SELECT * FROM [Номера] WHERE Статус=false";
          OleDbDataAdapter dta = new OleDbDataAdapter(cmd,connString);
          DataSet ds = new DataSet();
          dta.Fill(ds);
          dataGridView1.DataSource = ds.Tables[0].DefaultView;        }
```

После выполнения данного кода перед пользователем представляется окно, показанное на рисунке 12.

Схема выполнения данного участка кода представлена на рисунке 12.

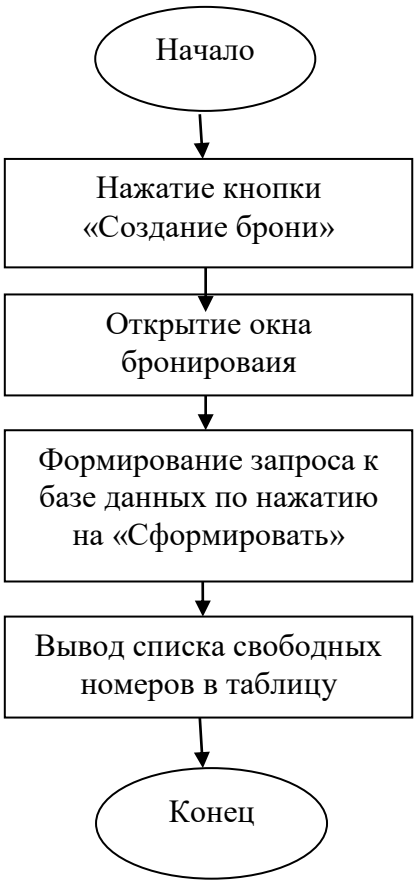

Рисунок 12 – Схема формирования списка свободных номеров

Для того, чтобы забронировать номер, необходимо сделать щелчок левой кнопкой мыши на строку с нужным номером, после этого перед пользователем представится диалоговое окно, изображенное на рисунке 13.

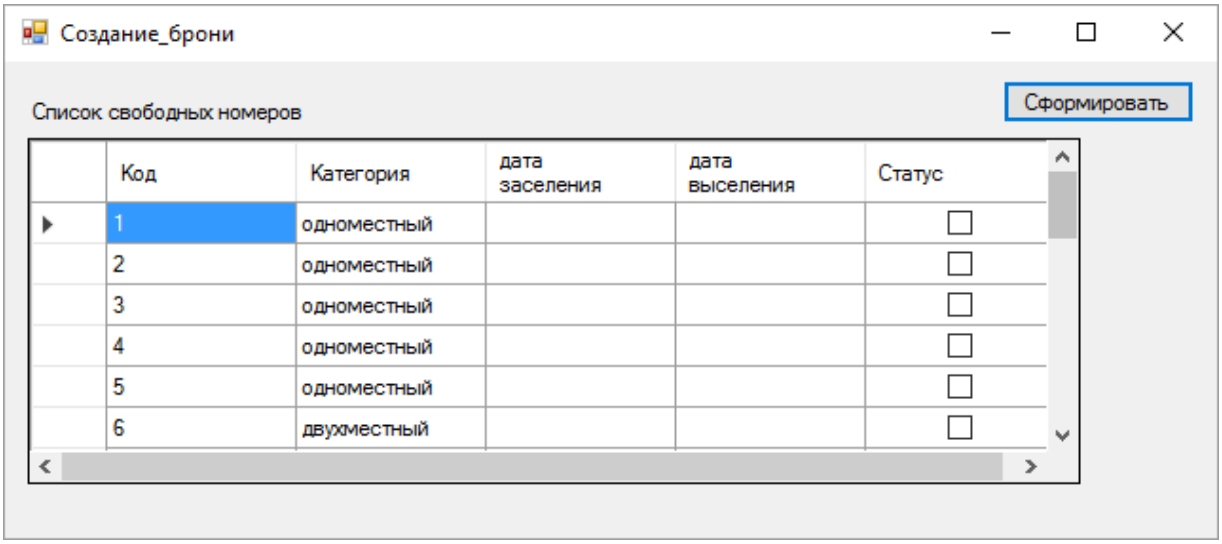

Рисунок 13 – Сформированная база свободных номеров

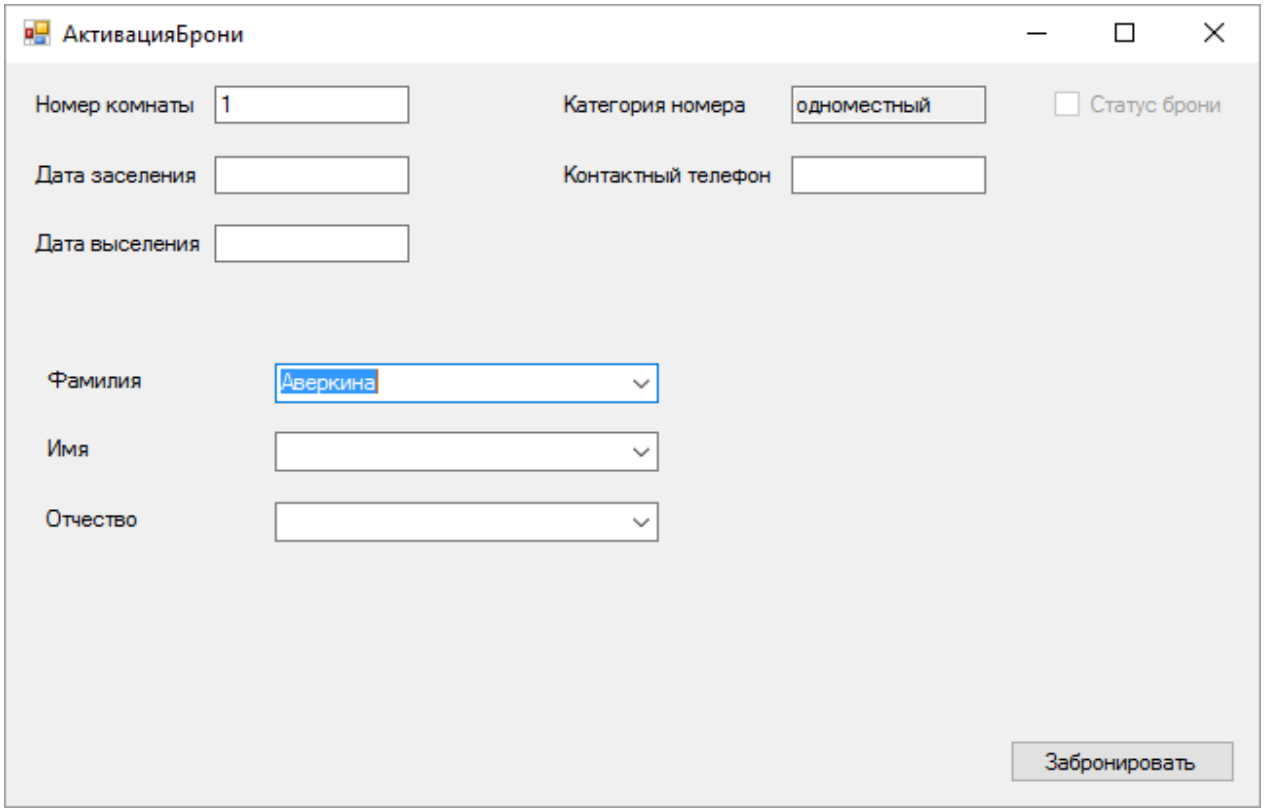

Рисунок 14 – Диалоговое окно создания брони

В процессе создания брони, необходимо указать данные, представленные на рисунке 14. Создать бронь возможно только для клиента, имеющегося в базе данных. По совпадению фамилии клиента составляется список имен, соответствующих фамилии, далее по фамилии и имени, составляется список возможных отчеств. Таким образом, введенные данные в любом случае соответствуют одному из посетителей. Автоматически заполняется поле «Контактный телефон», в соответствии с Ф.И.О. клиента. После нажатия кнопки «Забронировать», в базе данных, таблице «Номера», изменяется статус брони номера, появляется так дата заселения и выселения, а так же заносится запись в таблицу архив.

В архив заносятся данные о брони и пользователе, который сделал бронь. За вышеперечисленный функционал отвечает следующий участок кода:

```
string connString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Гостинница1.mdb";
        public АктивацияБрони()
        {  InitializeComponent();
```

```
                if (checkBox1.Checked == true)
                { button1.Enabled = false;            }
        }
                public string creator = "";

                private void button1_Click(object sender, EventArgs e)
                {
        try
                {//строка запроса на добавление записи в архив
        string qry = "INSERT INTO [Архив] ([Код номера],Имя, [Номер телефона], [Дата заезда],[Дата выезда],Сотрудник) Values (" + "\"" + RoomsCode.Text + "\"" + "," + "\"" + comboBox1.Text + "\"" + "," + "\"" + Phone.Text + "\"" + "," + "\"" + InDate.Text + "\"" + "," + "\"" + OutData.Text + "\"" + "," + "\"" + creator + "\"" + ")";

                        OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);
                        DataSet ds = new DataSet();
        //добавление записи в архив
                        dta.Fill(ds);

        //запрос на обновление таблицы комнат
        qry = "UPDATE [Номера] SET Статус=true ,[дата заселения]=" + "\"" + InDate.Text + "\"" + ",[дата выселения]=" + "\"" + OutData.Text + "\"" + " WHERE Код=" + RoomsCode.Text;

                        dta = new OleDbDataAdapter(qry, connString);

                        DataSet ds1 = new DataSet();

                        dta.Fill(ds1);

                        checkBox1.Checked = true;

                        if (checkBox1.Checked == true)

                        {       button1.Enabled = false;  }
                    }   catch
                    { MessageBox.Show("Проверьте вводимые данные");    } }
                private void comboBox3_TextChanged(object sender, EventArgs e)
                {   try       {  if (comboBox1.Text != "" && comboBox2.Text != "")
                {//Выборка номера телефона в соответствии с Ф.И.О.
string qry = "SELECT [Номер телефона] FROM [Контакты_Клиента] WHERE ID IN (SELECT [Код] FROM [Клиенты] WHERE ((Фамилия = " + "\"" + comboBox1.Text + "\"" + "AND Имя=" + "\"" + comboBox2.Text + "\"" + ")AND Отчество=" + "\"" + comboBox3.Text + "\"" + "))";
```

```
                        OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

                        DataSet ds = new DataSet();

                        dta.Fill(ds);

                        Phone.Text = ds.Tables[0].Rows[0][0].ToString();  }
        }
                    catch            {  } }
                private void comboBox1_TextChanged(object sender, EventArgs e)
                {  DataTable dt1 = new DataTable();

        //Составление списка возможных Имен в соответствии с выбранной фамилией
string qry = " SELECT [Имя] FROM [Клиенты] WHERE Фамилия="+"\""+comboBox1.Text+ "\"";
                    OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

                    dta.Fill(dt1);

                    comboBox2.DataSource = dt1;

                    comboBox2.DisplayMember = "Имя";
        }
                private void comboBox2_SelectedValueChanged(object sender, EventArgs e)

                {//Составление списка возможных отчеств в соответствии с именем и фамилией
                    DataTable dt2 = new DataTable();
        string qry = "SELECT [Отчество] FROM [Клиенты] WHERE Имя=" + "\"" + comboBox2.Text +
"\"" + "AND Фамилия=" + "\"" + comboBox1.Text + "\"";

                    OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);
                    dta.Fill(dt2);

                    comboBox3.DataSource = dt2;

                    comboBox3.DisplayMember = "Отчество";
        }
```

Блок схема выполнения данного участка кода представлена на рисунке 15.

При выборе пункта «Удаление брони», пользователю представляется следующее диалоговое окно, с которым можно ознакомиться на рисунке 16.

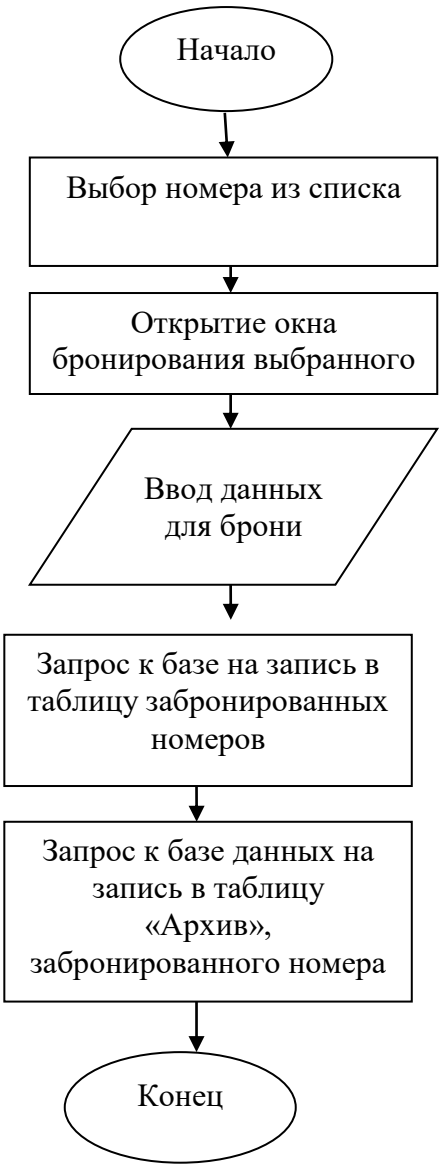

Рисунок 15 – Блок схема выполнения бронирования номеров

Рисунок 16 – Окно снятия брони

При нажатии кнопки «сформировать», выводится таблица забронированных в настоящее время номеров. После выделения конкретной занятой комнаты и последующего нажатия кнопки «Снять бронь», номер освобождается, меняя свой статус в базе данных. За представленные функции отвечает следующий участок кода:

```
public partial class reserve_delete : Form
    {//строка подключения к бд
        string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\\Гостинница1.mdb";
        int changedindex = 0;
        public reserve_delete()
        {        InitializeComponent();        }
        private void button1_Click(object sender, EventArgs e)
        {
// выборка номеров занятых номеров
            string cmd = "SELECT * FROM [Номера] WHERE Статус=true";
            OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);
            DataSet ds = new DataSet();
            dta.Fill(ds);
            dataGridView1.DataSource = ds.Tables[0].DefaultView;        }
        private void dataGridView1_SelectionChanged(object sender, EventArgs e)
        {   button2.Enabled = true;       }
        private void button2_Click(object sender, EventArgs e)
        {string qry = "UPDATE [Номера] SET Статус=false ,[дата заселения]=\"\""+",
[дата выселения]=\"\"" + " WHERE Код=" +
dataGridView1.Rows[changedindex].Cells[0].Value.ToString();
           OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);
           DataSet ds = new DataSet();
           dta.Fill(ds);
           button1_Click(sender, e);        }
    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {    changedindex = e.RowIndex;       }
```

Блок схема выполнения данного участка кода представлена на рисунке 17.

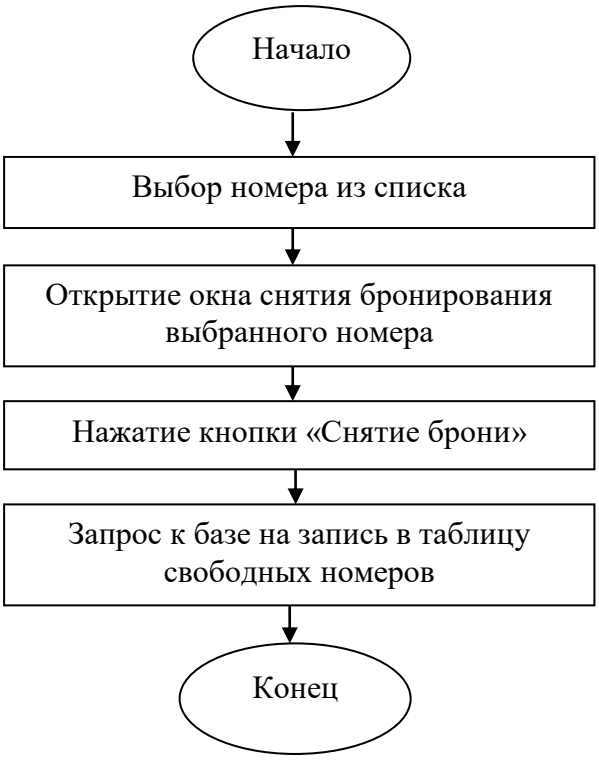

Рисунок 17 – Блок схема выполнения снятия брони с номера

Пункт «Управление записями» позволяет добавить: клиента; сотрудника; комнату;

При выборе каждого из пунктов открывается соответствующее диалоговое окно, которое предоставляет поля для ввода необходимой информации с целью создания выбранного объекта.

Диалоговое окно для добавления клиента представлено на рисунке 18, для добавления сотрудника - на рисунке 19 и для добавления комнаты - на рисунке 20.

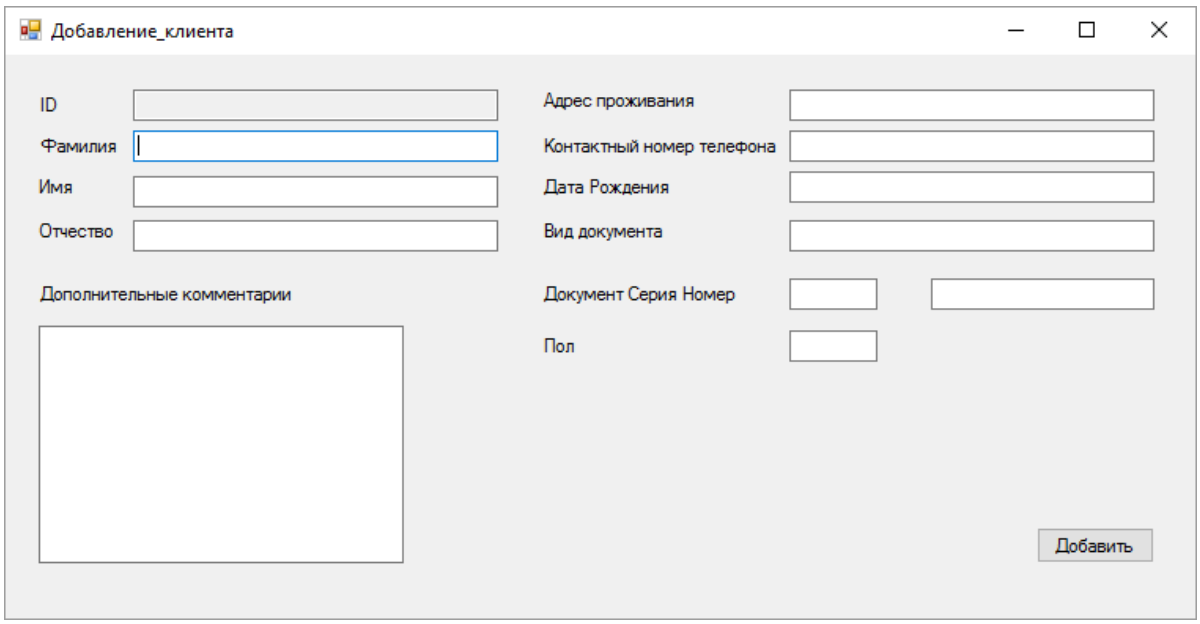

Рисунок 18 – Окно добавления клиента

Данные, которые вводятся в данную форму, добавляются в базу данных в таблицу «Клиенты» и таблицу «Контакты_клиента». Код, отвечающий за добавление клиента:

```
public partial class Добавление_клиента : Form
{string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Гостинница1.mdb";
              public Добавление_клиента()
              { InitializeComponent();        }
              private void button1_Click(object sender, EventArgs e)
              {//Запрос в базу данных для добавления клиента
string qry = "INSERT INTO [Клиенты] (Фамилия,Имя,Отчество,[Дата рождения],Пол,[Вид документа],Серия,Номер,Комментарий) Values (" + "\"" + SoName.Text + "\"" + "," + "\"" + Name1.Text + "\"" + "," + "\"" + SecondName.Text + "\"" + "," + "\"" + BirthDate.Text + "\"" + "," + "\"" + Male.Text + "\"" + "," + "\"" + Kind_of_document.Text + "\"" + "," + "\"" + Serial.Text + "\"" + "," + "\"" + Number.Text + "\""+"," + "\"" +Comments.Text + "\"" + ")";
       //Второй запрос для добавления телефона и адреса в другую таблицу
string qry2 = "INSERT INTO [Контакты_Клиента]([Номер телефона], [Адрес проживания]) Values (" + "\"" + Phone.Text + "\"" + "," + "\"" + Adress.Text + "\"" + ")";
                  OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);
                  OleDbDataAdapter dta2 = new OleDbDataAdapter(qry2, connString);
                  DataSet ds = new DataSet();
                  DataSet ds2 = new DataSet();
                  dta.Fill(ds);
                  dta2.Fill(ds2);
                  this.Close();        }    }
```

Схема выполнения вышеуказанного участка кода представлена на рисунке 19.

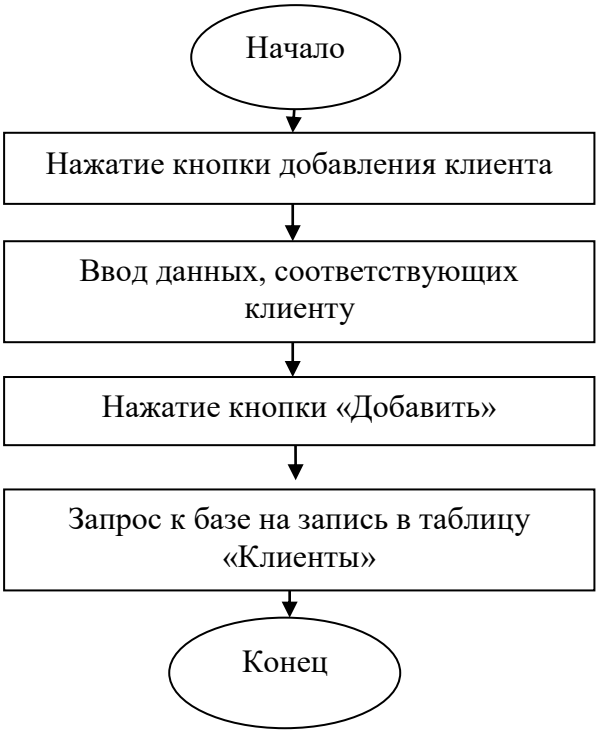

Рисунок 19 – Блок схема выполнения добавления клиента в базу данных

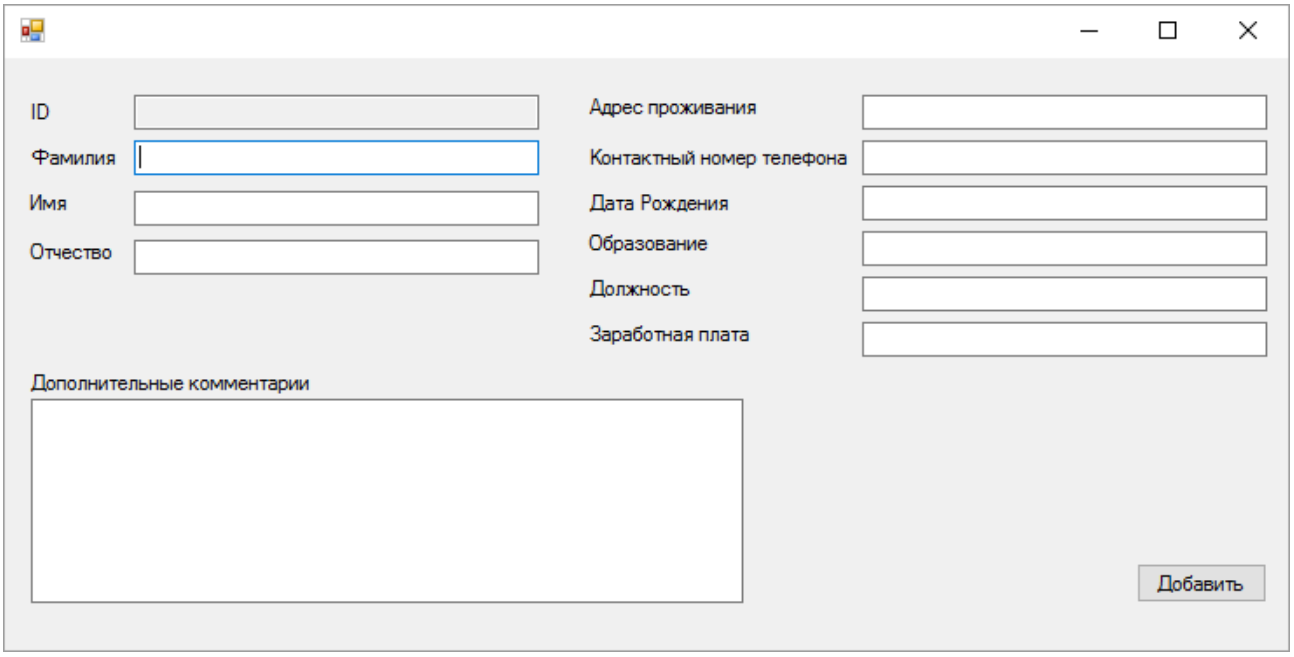

Рисунок 20 – Окно добавления сотрудника

Информация из данной формы заносится в таблицу «Сотрудники» базы данных по нажатию кнопки «добавить» Код, который отвечает за добавление сотрудника:

```
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Гостинница1.mdb";

                public Действия_с_записями_бд()

        {
         InitializeComponent();
        }
                private void button1_Click(object sender, EventArgs e)

                {
        //Запрос на добавление сотрудника

string qry = "INSERT INTO [Сотрудники] (Фамилия,Имя,Отчество,[Дата рождения],Образование,Должность,Адрес,Телефон,[Заработная плата],Комментарии) Values ("+"\""+SoName.Text+"\""+","+"\""+Name.Text+"\""+","+"\""+SecondName.Text+"\""+","+"\""+BirthDate.Text+"\""+","+"\""+
Education.Text+"\""+","+"\""+Status.Text+"\""+","+"\""+Adress.Text+"\""+","+"\""+Phone.Text+"\""+","+"\""+Grand.Text+"\""+","+"\""+Comments.Text+"\""+")";

                    OleDbDataAdapter dta = new OleDbDataAdapter(qry,connString);

                    DataSet ds = new DataSet();

                    dta.Fill(ds);

                    this.Close();
        }
```

Схема выполнение вышеописанного кода представлена на рисунке 21.

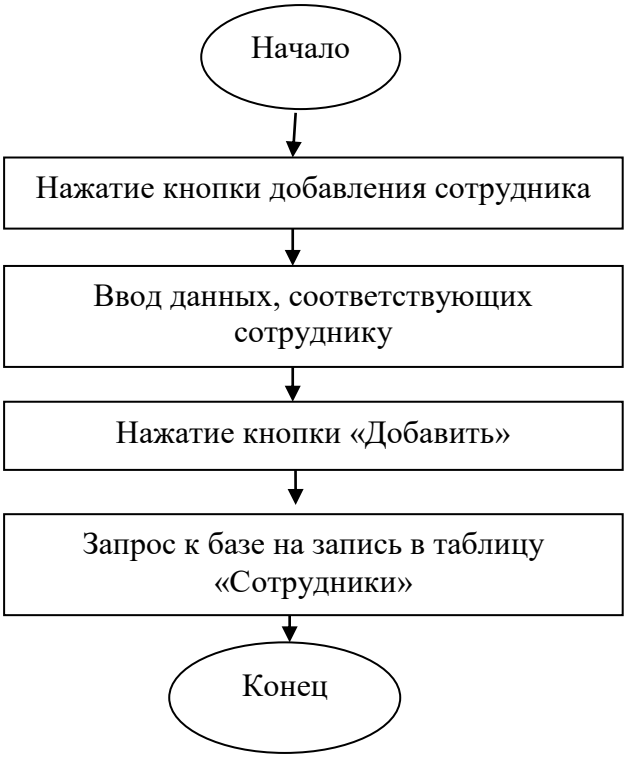

Рисунок 21 – Схема добавления сотрудника в базу данных

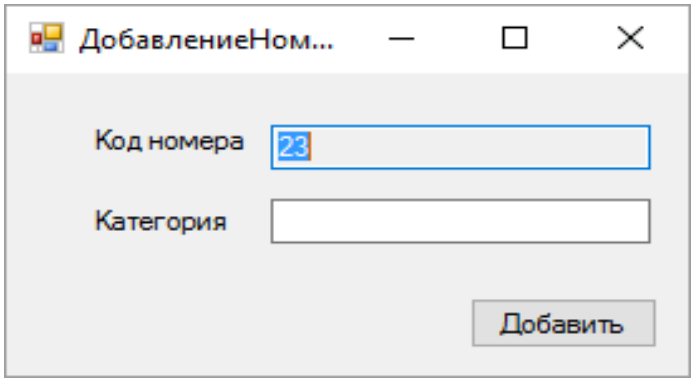

Рисунок 22 – Окно добавления номера

В соответствии с выбором пункта «Добавить номер» можно создать записи дополнительных помещений, которые появились в гостинице. По нажатию кнопки добавить, формируется запись в бд. Код:

```
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Гостинница1.mdb";

                private void button1_Click(object sender, EventArgs e)
                {
        //запрос на добавление номера в бд
```

```
string qry = "INSERT INTO [Номера] (Категория) Values (" + "\"" + Category.Text + "\"" +
")";
                    OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

                    DataSet ds = new DataSet();

                    dta.Fill(ds);

                    this.Close();
          }
```

Схема выполнения данного участка кода представлена на рисунке 23.

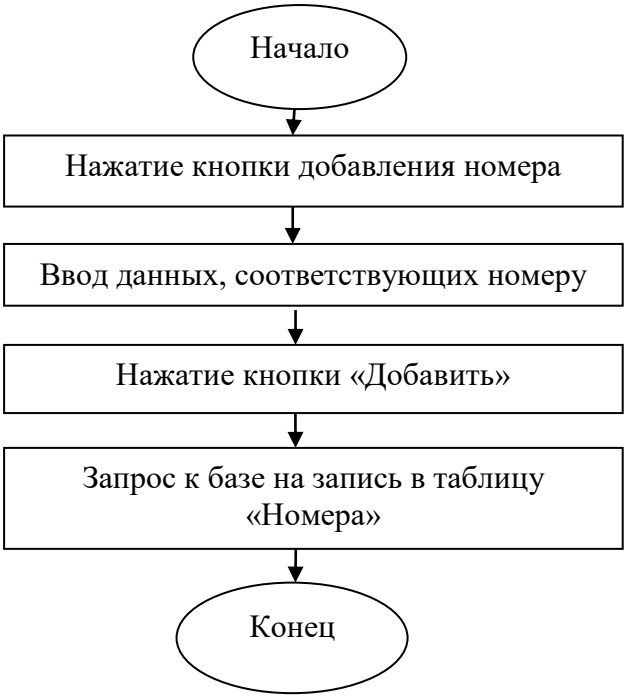

Рисунок 23 – Схема добавления номера в базу данных

Если пользователь имеет права на добавление учетных записей и выбирает пункт «Управление учетными записями», ему предоставляется диалоговое окно, которое позволяет добавить пользователя информационной системы. Данное диалоговое окно представлено на рисунке 24.

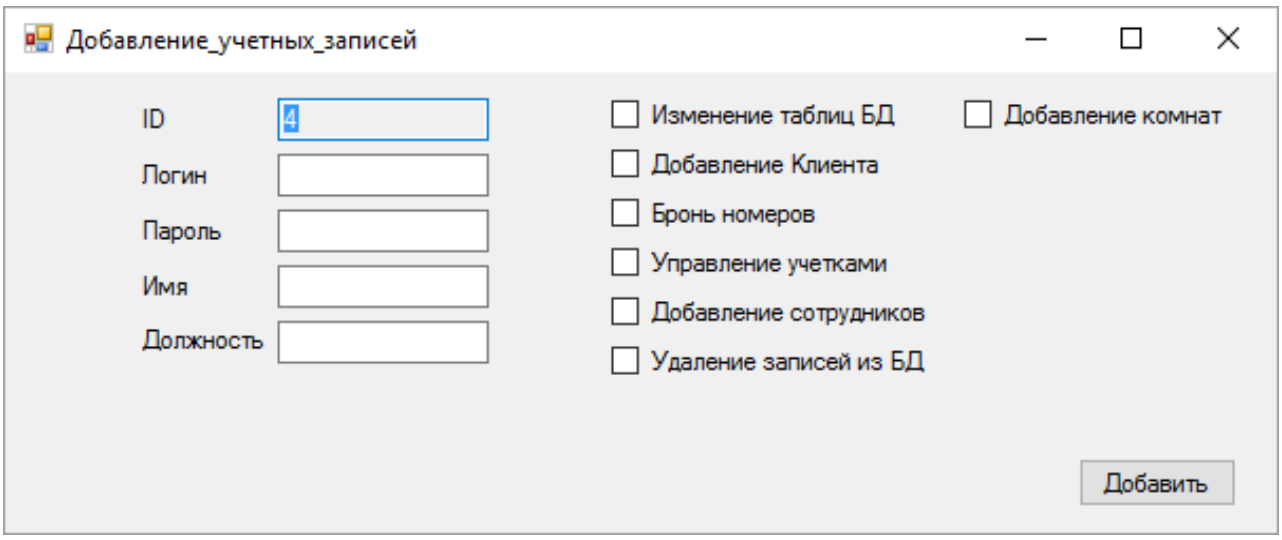

Рисунок 24 – Окно добавления учетных записей пользователей

После заполнения формы, выдачи прав и нажатия кнопки «Добавить», создается новый пользователь информационной системы. Код:

```
private void button1_Click(object sender, EventArgs e)
{
//запрос на добавление учетной записи
string qry = "INSERT INTO [ТаблицаАутентификации] (Логин,Пароль,Имя,Должность,Изменение,[Добавление клиентов],[Бронь номера],[Управление аутентификационными Данными],[Добавление Сотрудников],Удаление,[Добавить номер]) Values (" + "\"" + Login.Text + "\"" + "," + "\"" + Psswd.Text + "\"" + "," + "\"" + Name.Text + "\"" + "," + "\"" + State.Text + "\"" + "," + ChangeDB.Checked + "," + AddClient.Checked + "," + ReserveRoom.Checked + "," + AuthControl.Checked + "," + AddWorkers.Checked + "," + DeleteDBElements.Checked + ","+ AddRooms.Checked + ")";

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

this.Close();  }
```

Добавление учетной записи осуществляется по схеме представленной на рисунке 25.

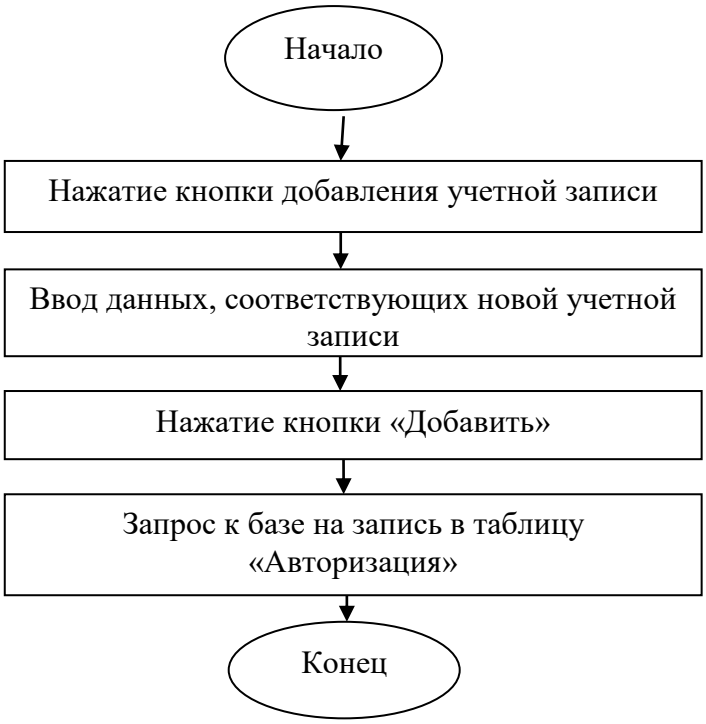

Рисунок 25 – Схема добавления учетной записи пользователя в базу данных

Так же на главном окне программы имеется поисковая система по базе данных с учетом определенных параметров, данная система представлена на рисунке 26.

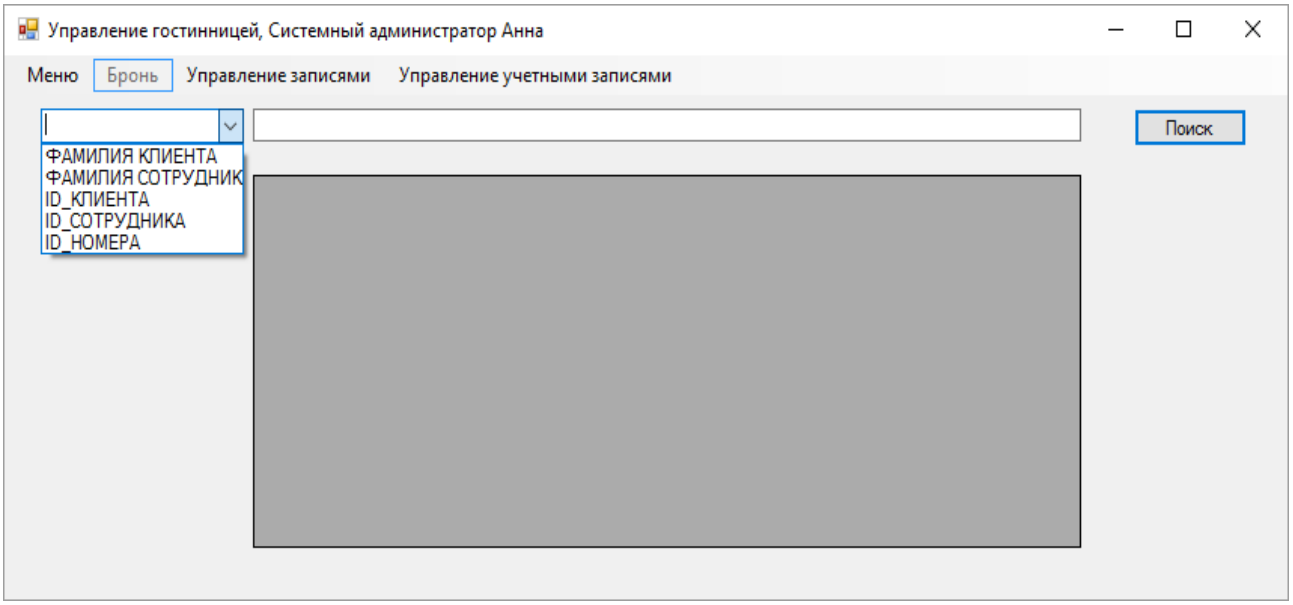

Рисунок 26 – Поисковая система по параметрам

После нажатия кнопки поиск окна, изображенного на рисунке 26, происходит запрос к базе данных. Пример запроса представлен на рисунке 28.

Схема выполнения запроса представлена на рисунке 27.

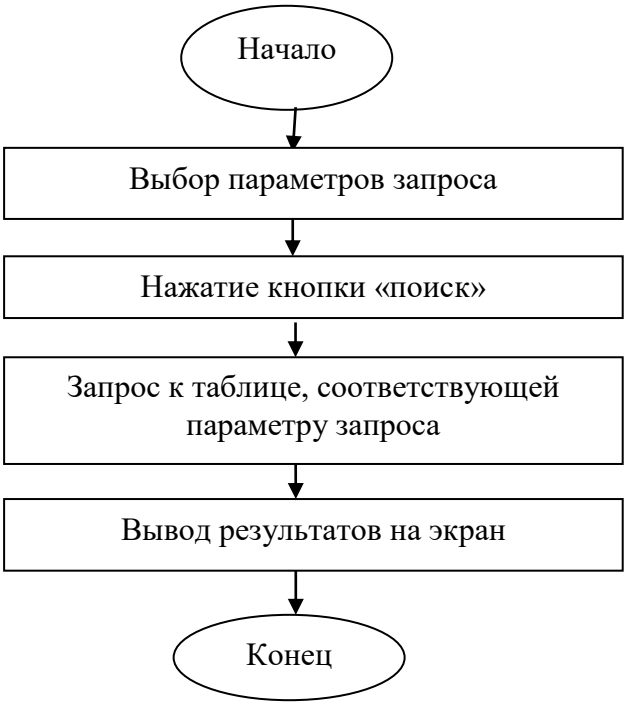

Рисунок 27 – Схема выполнения запроса в поисковой системе

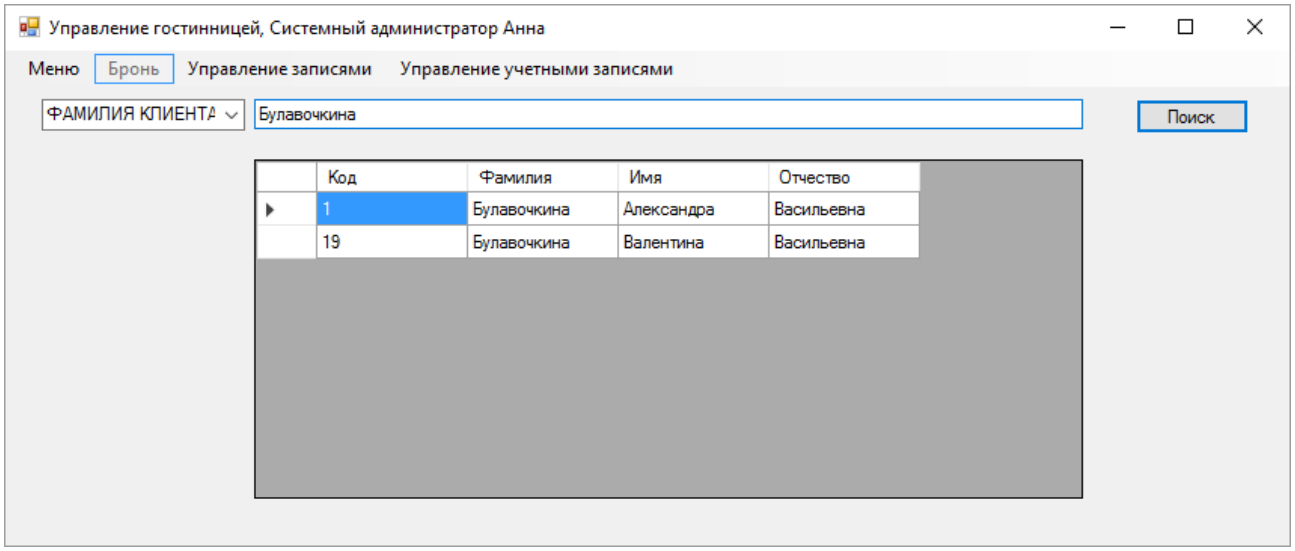

Рисунок 28 – Пример работы поисковой системы

По нажатию любой строки таблицы, открывается окно с подробной информацией о том или ином клиенте. В нем же есть возможность ее изменения или удаления записи вовсе. Пример такого окна можно видеть на рисунке 29.

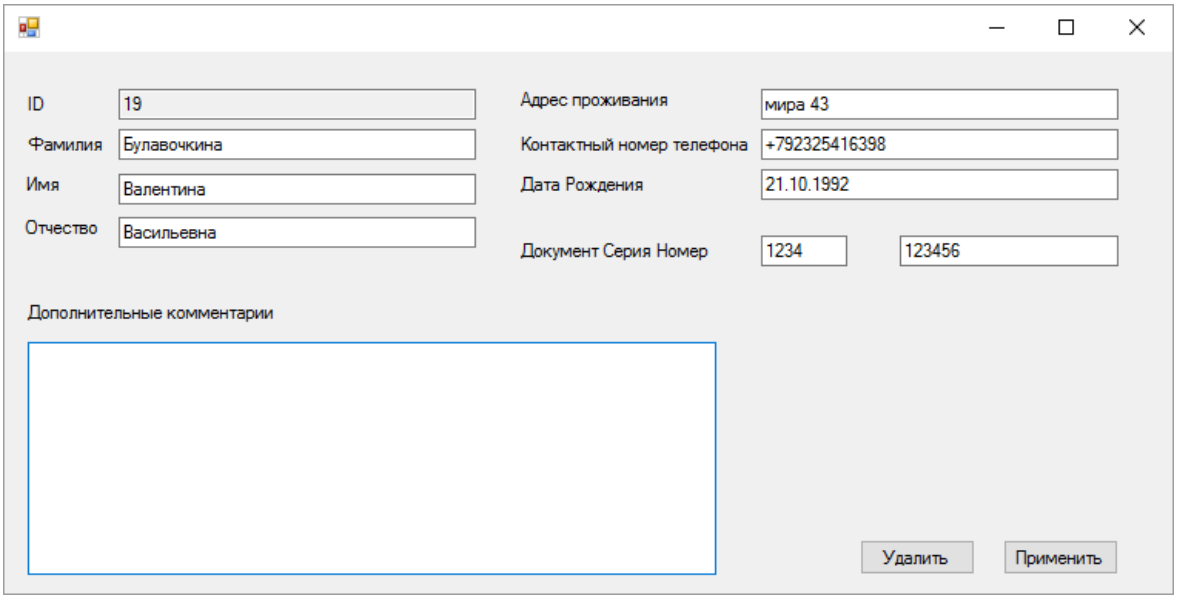

Рисунок 29 – Подробная информация об объекте

За этот функционал отвечает следующий участок кода:

```
private void button2_Click(object sender, EventArgs e)
        {   if (comboBox1.Text != "")
            {  string Search_variable = "";
                Search_variable = Search_parametr(comboBox1.Text);
                string CmdText = "";  string Cmd = "";
               // string CmdText = "SELECT * FROM " + Search_variable
               //+" WHERE [Код клиента] = " +"\""+ textBox1.Text+"\"";
          if (Search_variable == "[Клиенты]" || Search_variable == "[Сотрудники]")
      { CmdText = "SELECT [Код],[Фамилия],[Имя],[Отчество] FROM " + Search_variable;
                    //+ " WHERE Фамилия = " + textBox1.Text;
                    Cmd = "SELECT * FROM " + Search_variable;
                if (comboBox1.Text == "ID_СОТРУДНИКА" || comboBox1.Text == "ID_КЛИЕНТА")
                    {   CmdText = CmdText + "WHERE [Код]=" + textBox1.Text;
                        Cmd = Cmd + "WHERE [Код]=" + textBox1.Text; }
else if (comboBox1.Text == "ФАМИЛИЯ СОТРУДНИКА" || comboBox1.Text == "ФАМИЛИЯ КЛИЕНТА")
                  { CmdText = CmdText + "WHERE [Фамилия]=" + "\"" + textBox1.Text + "\"";
                Cmd = Cmd + "WHERE [Фамилия]=" +"\"" + textBox1.Text + "\"";      }
```

```
OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CmdText, connString);
 OleDbDataAdapter dataAdapter2 = new OleDbDataAdapter(Cmd, connString);
                    // создаем объект DataSet
                    DataSet ds = new DataSet();DataSet ds2 = new DataSet();
                    // заполняем таблицу Order    // данными с базы данных
                    try   { dataAdapter.Fill(ds);
                        dataAdapter2.Fill(ds2);
                        Dt = ds2.Tables[0];
                        dataGridView1.DataSource = ds.Tables[0].DefaultView;  }
                  catch {  MessageBox.Show("Запрашиваемые данные отсутствуют");}   }
                else if (Search_variable == "[Номера]")
{ CmdText = "SELECT * FROM " + Search_variable+"WHERE [Код]="+ textBox1.Text;
     OleDbDataAdapter dt_clients = new OleDbDataAdapter(CmdText, connString);
                    DataSet ds_clients = new DataSet();
                    dt_clients.Fill(ds_clients);
                    Dt = ds_clients.Tables[0];
                    dataGridView1.DataSource = ds_clients.Tables[0].DefaultView;
                }  }   }
```

Вышепоказанный код соответствует выполнению по блок схеме представленной на рисунке 30.

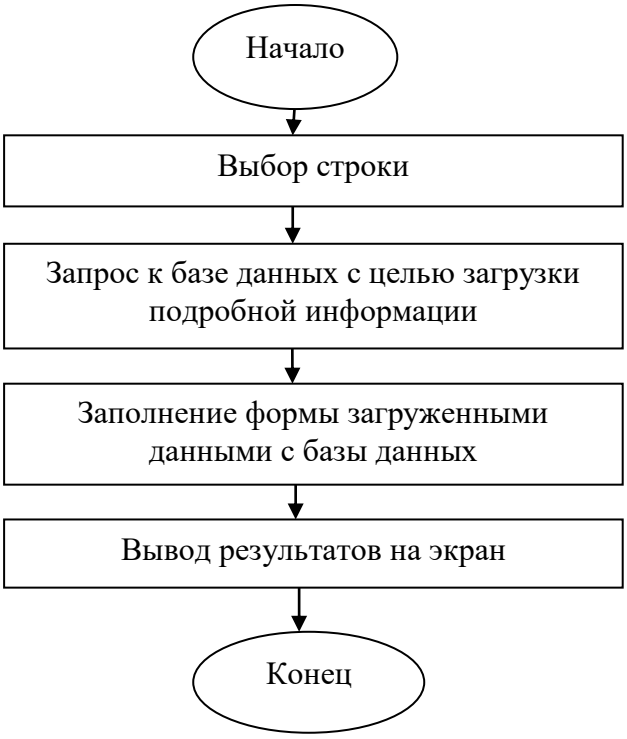

Рисунок 30 – Схема отображение подробной информации по объекту запроса

Так же в окне просмотра подробной информации по конкретному объекту, представленном на рисунке 29, есть возможность его изменения или удаления записи о нем из базы данных. За это отвечают кнопки «Применить» и «Удалить».

Данная информационная система отвечает минимальному требуемому функционалу, который необходим для работы гостиничного комплекса. Реализованы функции авторизации и аутентификации, для ограничения доступа рабочего персонала к данным посетителей, что обеспечивает защиту персональных данных клиентов. Внедрена возможность добавления, изменения и удаления записей о посетителях, работников и их учетных записей в информационной системе из базы данных, что позволяет поддерживать ее в актуальном виде. Так же была выполнена функция бронирования / снятия брони номеров, она является одной из основополагающих в данной работе, так как напрямую связана с работой гостиничного комплекса.

ЗАКЛЮЧЕНИЕ

В процессе работы были получены весомые знания в области гостиничного дела, навыки анализа рабочих процессов гостиницы. В процессе анализа был определен функционал, который был автоматизирован в представленной выше информационной системе. Так же на практике были отработаны навыки проектирования и создания баз данных, программирования и автоматизации рабочих процессов, необходимых для стабильной работы гостиничного комплекса. В дальнейшем планируется разработка более сложных модулей для данной информационной системы. Например, использования мобильных средств связи для передачи задач различным отделам гостиничного комплекса и последующего отчета по их выполнению, фиксации данных процессов в информационной системе. Это позволит ускорить работу обслуживающего персонала, повысить качество взаимодействия между отделами и улучшить контроль над обслуживающим персоналом. Так же планируется привести данную информационную систему к распределенному виду, централизовать базу данных на выделенном сервере, тем самым разделив клиентскую и серверную часть. Поставленные задачи в работе считаю выполненными, а цель достигнутой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СУБД Cache. Объектно-ориентрированная разработка приложений: учебный курс / В. Кирстен, М. Ирингер, П. Шульпе, Б. Рериг. – Санкт-Петербург: Питер, 2001. – 384 с.

2. Хомоненко, А. Д. Базы данных: учебник для высших учебных заведений. – 4-е изд., доп. и перераб. / под ред. проф. А. Д. Хомоненко. – Санкт-Петербург: КОРОНА принт, 2004. – 736-с.

3. Когаловский, М. Р. Энциклопедия технологий баз данных: учебное пособие / М. Р. Когаловский. – Москва: Финансы и статистика, 2002. – 800 с.

4. Ham Trans [Электронный ресурс]: электронный журнал // Специфика гостиничного бизнеса. - 2013. - Режим доступа: http://hamtrans.ru/allarti.aspx?id=952.

5. Professor web [Электронный ресурс]: учебник // Отладка кода, язык программирования C#. – 2014. - Режим доступа: http://professorweb.ru/my/csharp/charp_theory/level2/2_8.php.

6. Life Prog [Электронный ресурс]: электронный курс // Объектно-ориентированное программирование C#. - 2014. - Режим доступа: http://life-prog.ru/view_infosystem.php?id=29.

7. Кречетов, Н. Посреляционная технология Cache для реализации объектных приложений/ Н. Кречетов. – Москва: МИФИ, 2001. – 152 с.

8. СТО 4.2-07-2014. Система менеджмента качества Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 30 декабря 2013. – Красноярск: ИПК СФУ, 2013. – 55.

9. Developer Network [Электронный ресурс]: электронный журнал // Пошаговое руководство. Подключение к данным в базе данных Access (Windows Forms). - 2014. - Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms171893.aspx.

10. Easy prog [Электронный ресурс]: электронный курс // Работа с базами данных на C#. Урок 5. Создание таблицы в Access. - 2014. - Режим

доступа: http://easyprog.ru/index.php?option=com=view&id=902&Itemid=54.

11. Redov [Электронный ресурс]: электронный курс // Маркетинг в области гостиничного бизнеса. – 2015. - Режим доступа: http://www.redov.ru/nauchnaja_literatura_prochee/osnovy_industrii_gostepriimstva/p12.php.

12. Professor web [Электронный ресурс]: электронный курс // Отладка кода, язык программирования C#. – 2014. - Режим доступа: http://professorweb.ru/my/csharp/charp_theory/level2/2_8.php.

13. Кузнецов С. Д. Основы баз данных: учебное пособие / С. Д. Кузнецов - 2-е изд., испр. – Москва: Интернет-Университет Информационных Технологий; БИНОМ Лаборатория знаний, 2007. - 484 с.

14. Грей П. Логика, алгебра и базы данных. – Москва: Машиностроение, 1989. - С. 188-213. - 368 с.

15. Особенности разработки информационных систем / Бурмистров А.С., Данилова Т.С., Сальшин В.И., Умаров А.С., Зелепухина В.А., Тарасевич Ю.Ю. - Ижевск: РХД, 2013.- С. 205.

16. Мартин, Г. SQL / Мартин Грабер. - Казань : Издательство «ЛОРИ», 2003. – 644 с.

17. Бойко, В.В. Проектирование баз данных информационных систем./ Бойко В.В., Савинков В.М. – Москва : Финансы и статистика, 1989. - 455 с.

18. Базы данных. Проектирование, реализация, сопровождение. Теория и практика. / Томас М. Конноли, Каролин Е. Бегг – Москва-Санкт-Петербург-Киев, 2007. – 1111 с.

19. Microsoft developer network [Электронный ресурс]: учебное пособие // Отладка управляемого кода, C#. – 2016. - Режим доступа: https://msdn.microsoft.com/ru-ru/library/awtaffxb.aspx.

20. Csharpcoderr [Электронный ресурс]: электронный журнал // Справочник по C#. – 2015. - Режим доступа: www.csharpcoderr.com/2012/03/access.html.