Содержание

[Введение 4](#_Toc11839896)

[1 Объектно-ориентированный анализ и проектирование системы 5](#_Toc11839897)

[1.1 Сущность задачи 5](#_Toc11839898)

[1.2 Проектирование модели 5](#_Toc11839899)

[2 Вычислительная система 10](#_Toc11839900)

[2.1 Требования к аппаратным и операционным ресурсам 10](#_Toc11839901)

[2.2 Инструменты разработки 10](#_Toc11839902)

[3 Проектирование задачи 12](#_Toc11839903)

[3.1 Требования к приложению 12](#_Toc11839904)

[3.2 Концептуальный прототип 12](#_Toc11839905)

[3.3 Организация данных 15](#_Toc11839906)

[3.4 Функции: логическая и физическая организация 17](#_Toc11839907)

[3.5 Проектирование справочной системы приложения 19](#_Toc11839908)

[4 Описание программного средства 20](#_Toc11839909)

[4.1 Общее сведения 20](#_Toc11839910)

[4.2 Функциональное назначение 20](#_Toc11839911)

[4.3 Входные и выходные данные 20](#_Toc11839912)

[5 Методика испытаний 22](#_Toc11839913)

[5.1 Технические требования 22](#_Toc11839914)

[5.2 Функциональное тестирование 22](#_Toc11839915)

[6 Применение 25](#_Toc11839916)

[6.1 Назначение программы 25](#_Toc11839917)

[6.2 Условия применения 25](#_Toc11839918)

[6.3 Справочная система 25](#_Toc11839919)

[Заключение 27](#_Toc11839920)

[Список информационных источников 28](#_Toc11839921)

[Приложение А Текст программы 29](#_Toc11839922)

[Приложение Б Результаты тестирования 51](#_Toc11839923)

# Введение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

КП Т.691015.401 ПЗ

Разраб.

Кочмарик Н.С.

Провер.

Багласова Е.В.

Т. Контр.

Н. Контр.

Утверд.

Систепа поддержки бронирования и заселения номеров гостиницы

Лит.

Листов

55

КБП

Black sun – небольшой двухэтажный отель. В работе отеля важно грамотно использовать доступные номера для заселения и зачастую это не простая задачи. Для упрощения этой задачи было разработано одноименное программное средство. Программное средство способно гибко обрабатывать входной потока заявок и контролировать заселение и бронирование номеров. В программном средстве также имеется возможность ручного редактирования данных чтобы иметь возможность обработать заявку человека вручную.

Пояснительная записка к курсовому проекту состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации программного приложения.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть автоматизированы. Описывается задача, перечисляются основные функции программы. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи.

Во втором разделе «Вычислительная система» перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются новые возможности программы, а также ее отличия от предыдущих версий.

В третьем разделе «Проектирование задачи» проводится объектно-ориентированный анализ задачи, строится концептуальный прототип системы меню, диалоговых окон и элементов управления.

В четвертом разделе «Описание программного средства» представлены общие сведения о программном средстве и его функциональном назначении.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Представляются результаты функционального тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы, а также методика ее использования.

В заключении будет проанализировано созданное программное приложение, определена степень соответствия поставленной задачи и выполненной работы.

Приложение будет содержать текст программы.

В графической части будут представлены диаграммы вариантов использования, классов, деятельности, последовательности и компонентов.

# 1 Объектно-ориентированный анализ и проектирование системы

## 1.1 Сущность задачи

В рамках курсового проектирования будет создано программное средство, предоставляющее возможность пользователю контролировать заселение и бронирование номеров в отеле, просмотр текущих заказов, клиентов, и статус каждого номера. Рассмотрим предметную область задачи:

* номер – помещение, в котором проживают клиенты;
* клиент – человек, который купил время проживания в определенном номере.

Программное средство должно обеспечивать следующие функции:

* создание и ведение базы данных;
* обработка базы данных;
* вывод информации на экран;
* симулировать работу;
* формировать отчетную документацию;
* редактировать данные.

Небольшая гостиница содержит К номеров (20 ≤ К ≤ 30). Различающихся по степени комфорта и стоимости: «люкс». «полулюкс», одноместные. Простые двухместные, двухместные с раскладным диваном (например, 70 у.е. за день проживания в одноместном номере, 120 у‚е. – за номер «люкс›>). Требуется создать компьютерную систему. Автоматизирующую управление занятостью номеров гостиницы. Система обрабатывает входной поток заявок двух видов:

* заявки бронирующие определенные типы номеров на определенный срок;
* заявки на заселение в текущий момент.

Система хранит информацию о фактической занятости всех номеров и о их занятости в ближайшие дни (учитываются уже оплаченные вперед дни), а также сведения о произведенной брони номеров. и использует все эти данные при обработке заявок.

Стратегия обработки заявок строится так, чтобы добиться максимальной занятости гостиницы с целью увеличения ее прибыли. Для этого система гибко распоряжается номерным фондом: в частности. при нехватке нужных номеров можно использовать пустующие номера большей комфортности (по меньшей цене). например, при нехватке одноместных номеров можно поселить одного человека в двухместный номер (за 70% его стоимости).

Для тестирования построенной системы необходимо смоделировать входной поток заявок на бронирование и поселение. Вид и параметры каждой заявки определяются случайным образом. Интервал между появлением двух заявок следует моделировать как случайную величину из определенною диапазона (например, от 1 до 5 часов).

Период моделирования – М дней (12 ≤ М ≤ 30), шаг – несколько часов. Цель моделирования - изучение стратегий обработки заявок на заселение. В параметры моделирования следует включить: числа К и М. количество номеров каждой категории, характеристики используемых случайных величин.

В ходе моделирования система должна представлять всю необходимую информацию о занятости номеров гостиницы. По окончании моделирования выводится статистика заселения номеров, выполненных заявок.

.

## 1.2 Проектирование модели

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получила дальнейшее развитие в работах Баркера.

Исходя из исследования предметной области, можно выделить следующие сущности разработки: заказ, клиент, комната, состояние, тип комнаты.

Для сущности «заказ» можно выделить следующие атрибуты:

* номер;
* дата регистрации;
* дата выезда;
* полная стоимость;
* активен.

Для сущности «клиент» можно выделить следующие атрибуты:

* номер паспорта;
* фамилия;
* имя;
* отчество;
* дата рождения;
* возраст;
* активен.

Для сущности «Комната» можно выделить атрибут номер комнаты.

Для сущности «Состояние» можно выделить атрибут название.

Для сущности «Тип комнаты» можно выделить следующие атрибуты:

* название;
* цена.

Диаграмма «сущность-связь» представлена на рисунке 1.1.

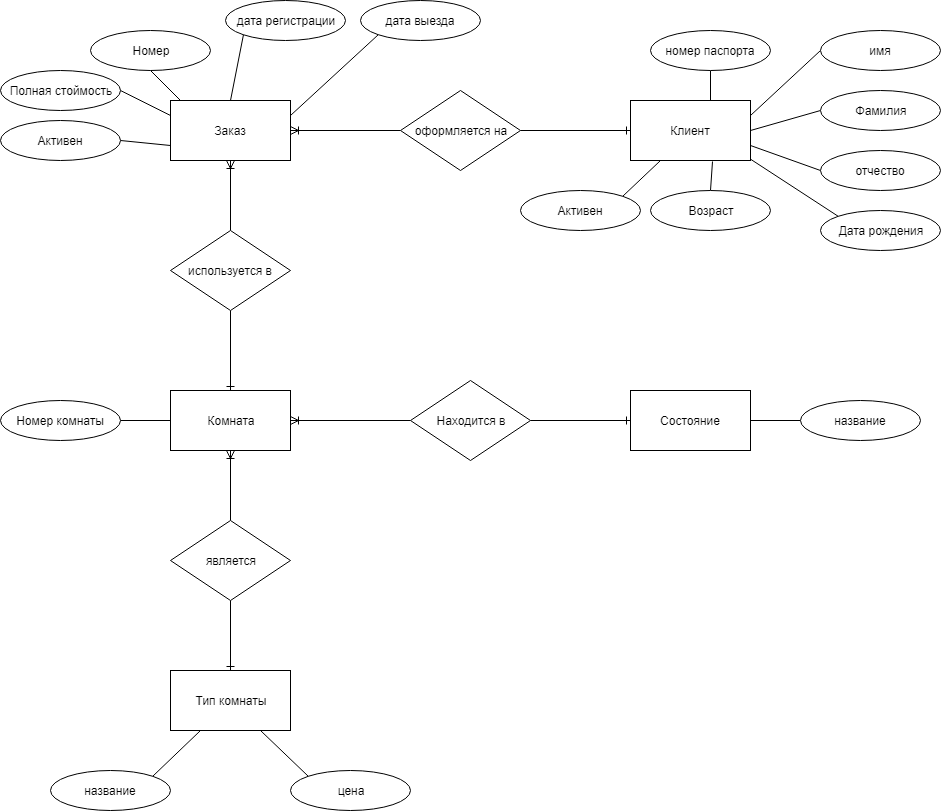


Рисунок 1.1 – Диаграмма «Сущность-связь»

Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

Актёр представляет собой внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует её функциональные возможности для достижения определённых целей или решения частных задач. При этом актёры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актёр может рассматриваться как некоторая отдельная роль относительно конкретного варианта использования.

Данное программное средство имеет следующие основные (Include) функции:

* вывод результата на печать;
* ручное редактирование;
* добавление в базу данных;
* удаление из базы данных;
* создание клиентов;
* создание заказов;
* отображение данных.

К вспомогательным (Extend) функциям, расширяющим возможности системы, относится функция симуляции.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы. Диаграмма классов представлена в графической части на листе 2.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности - визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

Диаграмма деятельности для функции добавления заказа представлена в графической части на листе 3.

Для моделирования взаимодействия объектов в UML используются соответствующие диаграммы взаимодействия. Если рассматривать взаимодействия объектов во времени, тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно - слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе - вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы.

Диаграмма последовательности для функции добавления клиента представлена в графической части на листе 4.

Рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо, некоторым образом, реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами.

Вид диаграммы компонентов для данной проектируемой системы представлен в графической части на листе 5 и содержит следующие компоненты:

* файлы базы данных «Hotel.sql» и «Hotel\_log.mdf»;
* файлы отчетов «\*.docx»;
* файл ресурсов «\*.resx»;
* файлы форм «\*.cs»;
* библиотеки «\*.dll».

# 2 Вычислительная система

## 2.1 Требования к аппаратным и операционным ресурсам

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

## 2.2 Инструменты разработки

Инструментами разработки для будущего программного приложения будут являться:

* операционная система Widows 10;
* программная среда разработки Microsoft Visual Studio 2017;
* язык программирования C#;
* ядро SQL Server 2017;
* система управления базами данных MS SQL Server Management Studio 2017;
* пакет MaterialSkin для создания графической части;
* программа для разработки справочной системы dr.Explain.

Для разработки данного проекта была использована операционная система Microsoft Windows 10 Максимальная.

Что касается среды разработки, то был выбран так же продукт компании Microsoft. Microsoft Visual Studio — линейка бесплатных интегрированных сред разработки, облегчённая версия Microsoft Visual Studio.

C# — На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в 7 марта 2017 года вместе с Visual Studio 2017.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Согласно корпорации Майкрософт, SQL Server 2017 является безопасной, масштабируемой и управляемой платформой, обеспечивающей работу ответственных бизнес-приложений. Он включает массу новых возможностей по работе с базами данных, в том числе регулятор ресурсов, сжатие данных, прозрачное шифрование, комплексный аудит и улучшенное зеркалирование баз данных. SQL Server 2017 повышает производительность труда администраторов баз данных, позволяя им управлять многими БД с помощью политик. Он также способствует разработке приложений, управляемых данными, за счет более тесной интеграции с Visual Studio и .NET Framework, а также поддержки всех типов данных.

Dr. Explain — программа для быстрого создания файлов справки (help-файлов), справочных систем, on-line руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам. Dr.Explain поддерживает вывод документации в такие форматы, как HTML (on-line руководства), CHM (файлы справки для ОС MS Windows®), RTF, и PDF. И все это на основе единого проекта-источника. Встроенная технология анализа структуры пользовательского интерфейса позволяет документировать экраны программных приложений почти автоматически.

# 3 Проектирование задачи

## 3.1 Требования к приложению

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

Кроме этого, при разработке форм необходимо соблюдать определённые требования: формы в приложении должны иметь стандартные элементы управления данными, не должны быть перегружены информацией, понятны простому пользователю. При конструировании форм в необходимых случаях нужно предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных:

* добавление записей должно осуществляется только через специальную форму;
* удаление записей должно осуществятся через специальную форму;
* длинна номера паспорта человека равняется четырнадцати символам;
* дата выезда из номера не может быть меньше даты регистрации.

## 3.2 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему меню и форм.

При запуске приложения будет открыта форма авторизации, представленная на рисунке 3.1. После входа открывается главная форма Рисунок 3.2. С помощью главной формы можно открыть форму для редактирования данных и запустить симуляцию.

При нажатии на кнопку «EDIT DATA» откроется форма редактирования данных. Эта форма разделена на 3 вкладки: вкладка с клиентами Рисунок 3.3, вкладка с заказами Рисунок 3.4 и вкладка с комнатами Рисунок 3.5. На вкладке с заказами можно осуществлять следующие действия: обновить содержимое, добавить нового клиента, удалить клиента, удалить клиентов без заказов. При нажатии на кнопку «добавить нового клиента» открывается форма добавления нового клиента Рисунок 3.6. На вкладке с заказами можно осуществлять следующие действия: обновить содержимое, добавить новый заказ, удалить заказ, удалить заказы с истекшим сроком. При нажатии на кнопку «добавить новый заказ» открывается форма добавления нового заказа Рисунок 3.7. На вкладке с комнатами можно получить информацию о комнатах.

При нажатии на кнопку «START SIMULATION» запускается симуляция работы отеля. Она также откроет форму для визуального отображения процесса работы Рисунок 3.8. По окончанию работы формируется отчет в Microsoft word по шаблону. Симуляция все делает автоматически, дополнительных действий пользователя не требуется.

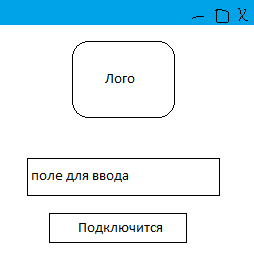


Рисунок 3.1 – Форма авторизации

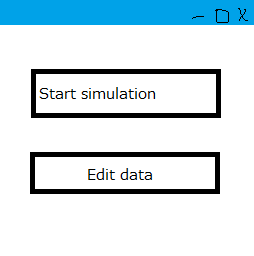


Рисунок 3.2 – Главная форма

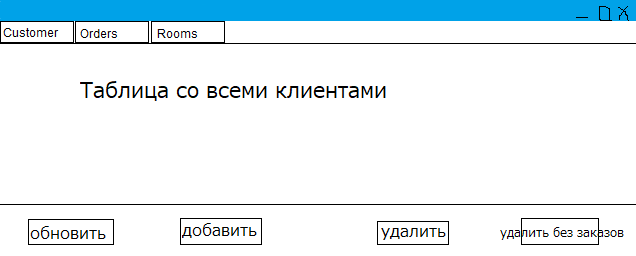


Рисунок 3.3 – Форма редактирования данных о клиентах

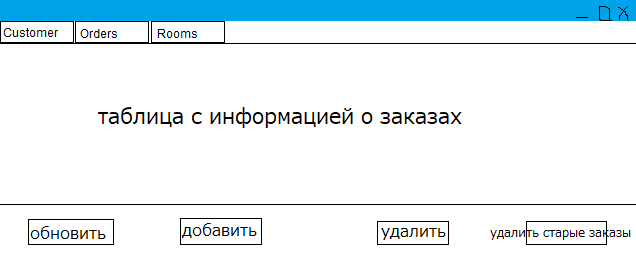


Рисунок 3.4 – Страница редактирования данных о заказах

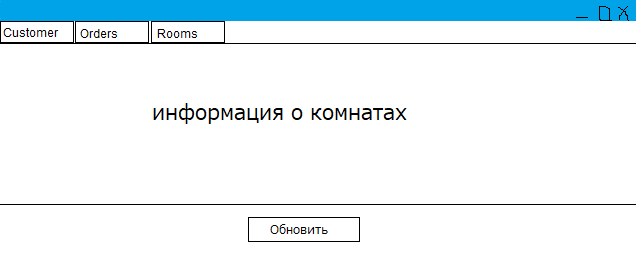


Рисунок 3.5 – страница с информацией о комнатах

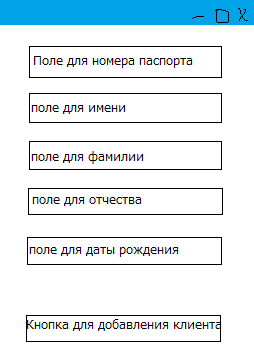


Рисунок 3.6 – форма добавления клиента

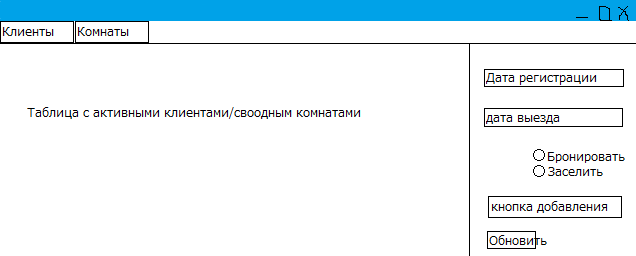


Рисунок 3.7 – форма добавления заказа

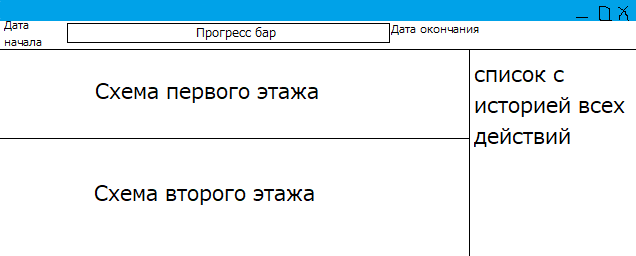


Рисунок 3.8 – Форма симуляции

## 3.3 Организация данных

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы - атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

В разделе 1.2 на рисунке 1.1 отображены главные сущности задачи: «Заказ», «Клиент», «Комната», «Состояние», «Тип\_комнаты». Каждой сущности ставится в соответствие таблица базы данных.

Структура базы данных разрабатываемого программного средства включает пять таблиц. Их краткое описание приводится в таблицах 3.1-3.5.

Таблица «Заказ» хранит информацию о заказах, ее структура представлена в таблице 3.1.

Таблица 3.1 – Структура таблицы «Заказ»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля | Описание поля |
| registration\_date | date | 3 | Дата регистрации |
| exit\_date | date | 3 | Дата выезда |
| room | int | 4 | Комната |
| fullPrice | money | 8 | Стоимость проживания |
| isActive | varchar | 1 | Является ли данный заказ активным |

Таблица «Клиент» хранит информацию о клиентах, она представлена в таблице 3.2

Таблица 3.2 – Структура таблицы «Клиент»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  Поля | Размер поля | Описание поля |
| passport\_number | varchar | 14 | Паспортный номер |
| name | varchar | 50 | Имя |
| sername | varchar | 50 | фамилия |
| patronymic | varchar | 50 | Отчество |
| date\_of\_birth | date | 3 | Дата рождения |
| isActive | varchar | 1 | Является ли данный клиент активным |

Таблица «Комната» хранит информацию о комнатах, она представлена в таблице 3.3.

Таблица 3.3 – Структура таблицы «Комната»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  Поля | Размер поля | Описание поля |
| Room\_number | Int | 4 | Номер комнаты |

Таблица «Состояние» хранит информацию о возможных состояниях комнаты, она представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Состояние»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  Поля | Размер поля | Описание поля |
| name | varchar | 30 | Название состояния |

Таблица «Тип\_комнаты» хранит информацию о типах комнаты, она представлена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Тип\_комнаты»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  Поля | Размер поля | Описание поля |
| name | varchar | 30 | Название типа комнаты |
| price | money | 8 | Цена за один день |

Структура базы данных представлена на схеме на рисунке 3.3.

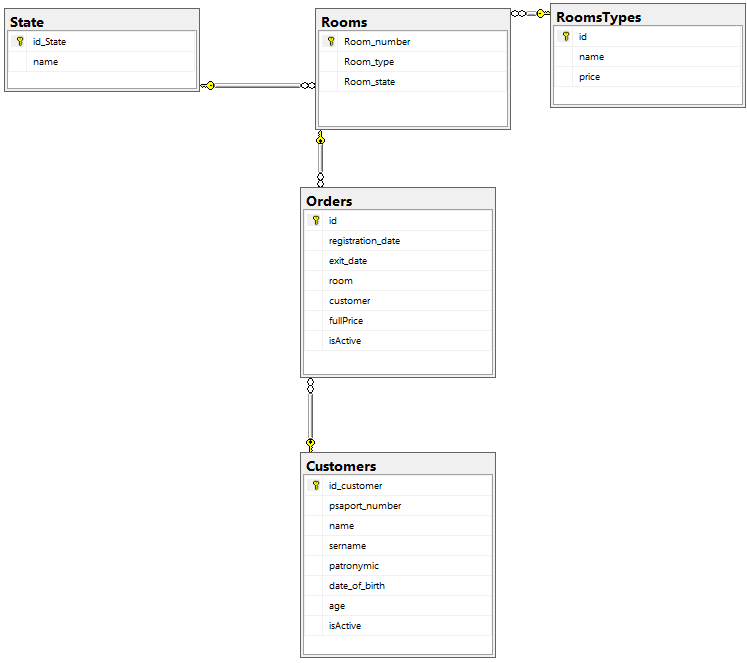


Рисунок 3.3 – Схема данных

## 3.4 Функции: логическая и физическая организация

Рассмотрим основные функции программы.

При нажатии на кнопку «ADD CUSTOMER» отвечает метод «AddCustomerButton\_Click», позволяющий добавить нового клиента в базу данных. Программный код реализации данного метода представлен ниже.

private void AddCustomerButton\_Click(object sender, EventArgs e)

{

if (this.Pasport.TextLength == 0 || this.textBox1.TextLength == 0 || (this.textBox2.TextLength == 0 || this.textBox3.TextLength == 0))

{

int num = (int) MessageBox.Show("all field mast'n be blunk");

}

else

{

using (SqlConnection connection = new SqlConnection(Program.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@name", (object) this.textBox3.Text));

sqlCommand.Parameters.Add(new SqlParameter("@sername", (object) this.textBox2.Text));

sqlCommand.Parameters.Add(new SqlParameter("@patronymic", (object) this.textBox1.Text));

sqlCommand.Parameters.Add(new SqlParameter("@dateOfBirth", (object) this.DateOfBirth.Value.Date));

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) this.Pasport.Text));

sqlCommand.ExecuteNonQuery();

}

this.Close();

}

}

При нажатии на кнопку «START SIMULATION» на главной форме будет вызван метод «StartSimulationButton\_Click» Программный код реализации данного метода представлен ниже.

private void StartSimulationButton\_Click(object sender, EventArgs e)

{

if (this.Logik != null)

return;

DateTime end = DateTime.Now.AddDays(15.0);

Form1.Ui = new Timer(end);

Form1.Ui.FormClosed += (FormClosedEventHandler) ((x, y) =>

{

this.Logik = (LogickForm) null;

Form1.Ui = (Timer) null;

if (!this.thread.IsAlive)

return;

this.thread.Abort();

});

Form1.Ui.Show();

this.Logik = new LogickForm(ref Form1.Ui, end);

this.thread = new Thread(new ThreadStart(this.Logik.StartSimulation));

this.thread.IsBackground = true;

this.thread.Start();

}

Весь текст программы представлен в приложении А.

## 3.5 Проектирование справочной системы приложения

Для работы с приложением начинающего пользователя необходимо обеспечить качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе пункта меню; сведения о том, какую информацию и в каком виде следует вводить в соответствующие поля, каким образом можно вносить изменения в базу данных.

Пользователь должен иметь доступ к справочной информации на протяжении всей работы с программой для этого пользователь может воспользоваться, открыв справочную систему из папки с приложением. Справочная система состоит из нескольких разделов, и предоставляет полную информацию о функциях приложения.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может пригодиться пользователю: о правилах пользования приложением, о его возможностях и о создателе программы.

Система справки данного программного средства будет содержать следующие разделы:

* «Начальная станица»;
* «Оглавление»;
* «Введение»;
* «Авторизация»;
* «Главное окно»;
* «Окно редактирования данных»;
* «Симуляция»

Справочная система будет создана в программе Dr.Explain.

# 4 Описание программного средства

## 4.1 Общее сведения

Из всех задач, которые будет решать разрабатываемое программное средство, можно выделить ряд основных:

* предоставление информации о комнатах, заказах и клиентах;
* добавление, удаление, редактирование информации о заказах и клиентах;
* вывод информации в файлы Microsoft Office;
* симуляция потока заявок и автоматическое управление заселением.

Разрабатываемое программное средство предназначено для управления заселенностью отеля.

Для применения данного программного средства необходимы следующие технические требования:

* процессор Pentium 800 или выше;
* минимальный объем оперативной памяти — 512 Мбайт;
* операционная система Windows10;
* рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
* клавиатура;
* мышь;
* ядро SQL Server;
* среда SQL Server Management Studio.

Программное средство создано в среде разработки Visual Studio 2017 на языке программирования C# в операционной системе Windows. Программное средство может работать в средах операционных систем семейства Microsoft Windows начиная с Windows 7. Программа не требовательна к системным ресурсам, также проста в использовании и не требует специальных навыков при работе.

Требуется инсталляция данного программного средства, а также необходимо присоединить базу данных к Microsoft SQL Server.

## 4.2 Функциональное назначение

Предметной областью решаемой задачи является управление заселенностью отеля.

Основными задачами приложения является максимально упростить работу администратора в отеле.

Работа с приложением осуществляется на локальном компьютере, сетевая поддержка присутствует.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобную систему для управления заселенностью.

## 4.3 Входные и выходные данные

Входными данными при добавлении клиента в базу данных являются данные о клиенте, а именно:

* номер паспорта;
* имя;
* фамилия;
* отчество;
* дата рождения.

Входными данными при добавлении клиента в базу данных являются данные о клиенте, а именно:

* клиент;
* комната;
* дата регистрации;
* дата выезда;
* тип заказа.

Выходные данные – выходными данными информация о т комнатах, заказах и клиентах.

# 5 Методика испытаний

## 5.1 Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* ядро SQL Server;
* среда SQL Server Management Studio.

Компьютер должен работать под управлением операционной системы, начиная с Windows7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 7, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

## 5.2 Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Таблица 5.1 – Тест-кейсы для проведения функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/  Функция | Шаги воспроизведения | Результат |
| 001 | Открыть главную форму | 1. Запустить базу данных; 2. В поле ввести адрес базы данных: DESKTOP-MI4SP6N\SQLEXPRESS; 3. Нажать на кнопку «CONNECT» | Ожидаемый результат:   1. Запускается приложение; 2. Данные введены; 3. Подключение успешно, открывается главная форма. |
| Фактический результат: совпал с ожидаемым.  Результаты тестирование предоставлены в приложении Б на рисунках: Б.1, Б.2. |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/  Функция | Шаги воспроизведения | Результат |
| 002 | Переключение вкладок на форме редактирования данных | 1. Открыть главную форму; 2. нажать на кнопку «EDIT DATA»; 3. нажать на вкладку «ORDERS»; 4. нажать на вкладку «ROOMS». | Ожидаемый результат:   1. Открывается главная форма; 2. открывается форма редактирования данных; 3. форма редактирования данных переключается на вкладку с заказами; 4. форма отображения данных переключается на вкладку с комнатами. |
| Фактический результат: совпал с ожидаемым.  Результаты тестирование предоставлены в приложении Б на рисунках: Б.3, Б.4, Б.5. |
| 003 | Ввод неправильных данных при авторизации | 1. Запустить приложение; 2. в поле ввести: «флтфщзцщцззцфцаф»; 3. нажать на кнопку «CONNECT». | Ожидаемый результат:   1. Запускается приложение; 2. Данные введены; 3. Происходит попытка подключения, после отображается окно с сообщением об ошибке. |
| Фактический результат: совпал с ожидаемым.  Результат тестирования предоставлен в приложении Б на рисунке Б.6. |
| 004 | Добавление нового пользователя | 1. Открыть форму редактирования данных; 2. Нажать на кнопку «ADD NEW CUSTOMER» 3. Заполнить поля: passport: «BSE6AEN12DAB3H»; name: «Петр»; surname: «Горохов»; patronymic: «Иванович»; date of birth: «вторник, 11 июня 1991 г.». 4. Нажать на кнопку «ADD CUSTOMER» | Ожидаемый результат:   1. Открывается форма редактирования данных; 2. открывается форма добавления клиента; 3. все поля заполнены верно; 4. происходит добавление клиента в базу данных. |
| Фактический результат: совпал с ожидаемым.  Результаты тестирования предоставлены в приложении Б на рисунках Б.7, Б.8 |
| 005 | Добавление нового заказа | 1. открыть форму редактирования данных; 2. кликнуть на вкладку с «ORDERS» 3. нажать на кнопку «ADD NEW ORDER» | Ожидаемый результат:   1. Открывается форма редактирования данных; 2. Осуществляется переключение на вкладку с заказами; |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/  Функция | Шаги воспроизведения | Результат |
|  |  | 1. Заполнить поля: registration date: «среда 12 июня 2019 г.» exit date: «среда 19 июня 2019 г.» Order Type: «Booked» 2. Выбрать клиента 3. Выбрать свободную комнату 4. Нажать на кнопку «ADD ORDER» | 1. Открывается форма добавления нового заказа; 2. Все поля заполнены верно; 3. Клиент выбран; 4. Комната выбрана; 5. Осуществляется добавление заказа в базу данных изменение статуса комнаты на «Booked». |
| Фактический результат: совпал с ожидаемым  Результаты тестирования предоставлены в приложении Б на рисунках Б.9, Б.10, Б.11 |

# 6 Применение

## 6.1 Назначение программы

Главным назначением программы является автоматизация и упрощение работы администратора в отеле.

Разработанное приложение имеет понятный интерфейс, довольно просто в применении, но в тоже время позволяет уменьшить затраты времени, используемые при вычислении за счет автоматизации вычислительных процессов

## 6.2 Условия применения

Для работы с приложением требуется установка дополнительных компонентов и специальных технических и программных ресурсов.

Для применения данного программного средства необходимы следующие технически требования:

* процессор Intel Core 2 Duo или выше;
* минимальный объем оперативной памяти — 512 Мбайт;
* операционная система Windows 10;
* Framework v4.5;
* рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
* устройство для чтения дисков;
* клавиатура;
* мышь;
* среда SQL Server Management Studio.

## 6.3 Справочная система

Справочная система необходима для ознакомления с программным средством. В ней будет изложена информация, которая может пригодиться пользователю. Содержание справочной системы должно быть просто и ясно изложено.

Справочная система разработана с помощью программы Dr.Explain, программа для быстрого создания файлов справки (help-файлов), справочных систем, on-line руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам.

Справочная система программного средства представляет собой отдельный файл «help.chm» с полным описанием основных функций программы в формате \*.chm. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справочная система данного программного средства содержит следующие разделы:

* «Начальная страница»;
* «Оглавление»;
* «Введение»;
* «Авторизация»;
* «Главное окно»;
* «Окно редактирования данных»;
* «Симуляция».

Справочная система по работе с программным средством будет представлена при открытии файла в корневом каталоге.

Структура справочной системы представлена на рисунке 6.1.

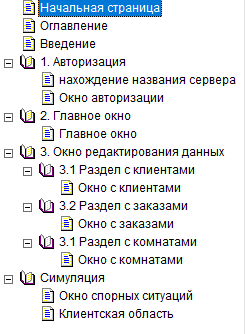


Рисунок 6.1 – Справочная система

# Заключение

В рамках курсового проектирования по предмету «Конструирование программ и языки программирования» было разработано программное средство, предоставляющее возможность редактирования, добавления и удаления информации, просмотр и печать информации.

Во время курсового проектирования удалось накопить неоценимый опыт в разработке приложений в интегрированной среде визуального программирования Microsoft Visual Studio 2017, развить умения самостоятельно находить оптимальный метод решения задачи. Кроме того, изучить принципы организации и технологии реализации программного обеспечения, развить умения разработки программ в соответствии с требованиями технического задания, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности. Получить навыки оформления комплекта документации на созданное программное обеспечение.

Целями курсового проектирования по предмету «Конструирование программ и языки программирования» являлось:

* закрепление знаний, связанных с технологией обработки информации;
* приобретение навыков разработки программ с использованием современных средств обработки экономической и деловой информации;
* выработка умений применять средства стандартных библиотек в практических задачах.
* Задачами курсового проектирования являлось:
* углубленное изучение принципов организации программного обеспечения и технологии его проектирования;
* владение методами надежного программирования;
* умение разрабатывать программы в соответствии с промышленными требованиями, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности;
* изучение способов определения экономической себестоимости и эффективности разработки программного обеспечения;
* оформление комплекта документации на созданное программное обеспечение.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2019. – 24 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
3. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник / А.М. Вендров. – 2-е изд., перераб. и доп.– М. : Финансы и статистика, 2006. — 544 с.
4. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. — 816 с.
5. Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2019. – Режим доступа : http://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/. – Дата доступа : 25.04.2019.
6. Dr.Explain [Электронный ресурс]. - Indigo Byte Systems, 2019. – Режим доступа : <https://www.drexplain.ru> – Дата доступа 25.04.2019.
7. Visual Studio 2017 [Электронный ресурс]. – Microsoft, 2019. – Режим доступа : [https://visualstudio.microsoft.com/ru/vs. – Дата доступа 22.04.2019](https://visualstudio.microsoft.com/ru/vs.%20–%20Дата%20доступа%2022.04.2019).

# Приложение А (обязательное) Текст программы

// Login Form

public LoginForm()

{

this.InitializeComponent();

MaterialSkinManager instance = MaterialSkinManager.Instance;

instance.AddFormToManage((MaterialForm) this);

instance.Theme = MaterialSkinManager.Themes.LIGHT;

instance.ColorScheme = new ColorScheme(Primary.Blue500, Primary.Blue700, Primary.Blue100, Accent.Red400, TextShade.WHITE);

}

private void TryToConnect()

{

string connectionString = "Data Source=" + this.materialSingleLineTextField1.Text + ";Initial Catalog=HOTEL;Integrated Security=True";

SqlConnection sqlConnection = new SqlConnection(connectionString);

try

{

sqlConnection.Open();

using (BinaryWriter binaryWriter = new BinaryWriter((Stream) File.Open("login", FileMode.OpenOrCreate)))

binaryWriter.Write(this.materialSingleLineTextField1.Text);

Program.connectionString = connectionString;

sqlConnection.Close();

this.Invoke((Delegate) (() => this.Close()));

}

catch (SqlException ex)

{

Console.WriteLine(ex.Message);

}

finally

{

if (Program.connectionString == null)

{

this.LoadingImage.Invoke((Delegate) (() => this.LoadingImage.Visible = false));

sqlConnection.Close();

int num = (int) MessageBox.Show("failed to connect to database");

}

}

}

private void materialFlatButton1\_Click(object sender, EventArgs e)

{

if (this.loadingThread == null)

{

this.LoadingImage.Visible = true;

this.loadingThread = new Thread(new ThreadStart(this.TryToConnect));

this.loadingThread.Start();

}

if (this.loadingThread == null || this.loadingThread.IsAlive)

return;

this.LoadingImage.Visible = true;

this.loadingThread = new Thread(new ThreadStart(this.TryToConnect));

this.loadingThread.Start();

}

private void LoginForm\_Load(object sender, EventArgs e)

{

this.materialFlatButton1.AutoSize = false;

this.materialFlatButton1.Size = new Size(207, 36);

this.materialLabel1.Font = new Font("Elephant", 16f, FontStyle.Italic);

using (BinaryReader binaryReader = new BinaryReader((Stream) File.Open("login", FileMode.OpenOrCreate)))

{

if (binaryReader.PeekChar() <= -1)

return;

this.materialSingleLineTextField1.Text = binaryReader.ReadString();

}

}

private void LoginForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!this.loadingThread.IsAlive)

return;

this.loadingThread.Abort();

}

//Program

internal static class Program

{

public static string connectionString;

public static void FillList(DataTable dataTable, ref MaterialListView listView)

{

foreach (DataColumn column in (InternalDataCollectionBase) dataTable.Columns)

listView.Columns.Add(column.ColumnName);

foreach (DataRow row in (InternalDataCollectionBase) dataTable.Rows)

{

ListViewItem listViewItem = new ListViewItem(row[0].ToString());

for (int index = 1; index < dataTable.Columns.Count; ++index)

listViewItem.SubItems.Add(row[index].ToString());

listView.Items.Add(listViewItem);

}

}

[STAThread]

private static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run((Form) new LoginForm());

if (Program.connectionString == null)

return;

Application.Run((Form) new Form1());

}

}

//Form1

public Form1()

{

this.InitializeComponent();

MaterialSkinManager instance = MaterialSkinManager.Instance;

instance.AddFormToManage((MaterialForm) this);

instance.Theme = MaterialSkinManager.Themes.LIGHT;

instance.ColorScheme = new ColorScheme(Primary.Blue500, Primary.Blue700, Primary.Blue100, Accent.Red400, TextShade.WHITE);

}

private void EditDataButton\_Click(object sender, EventArgs e)

{

if (this.dataForm != null)

return;

this.dataForm = new DataForm();

this.dataForm.FormClosed += (FormClosedEventHandler) ((x, y) => this.dataForm = (DataForm) null);

this.dataForm.Show();

}

private void StartSimulationButton\_Click(object sender, EventArgs e)

{

if (this.Logik != null)

return;

DateTime end = DateTime.Now.AddDays(15.0);

Form1.Ui = new Timer(end);

Form1.Ui.FormClosed += (FormClosedEventHandler) ((x, y) =>

{

this.Logik = (LogickForm) null;

Form1.Ui = (Timer) null;

if (!this.thread.IsAlive)

return;

this.thread.Abort();

});

Form1.Ui.Show();

this.Logik = new LogickForm(ref Form1.Ui, end);

this.thread = new Thread(new ThreadStart(this.Logik.StartSimulation));

this.thread.IsBackground = true;

this.thread.Start();

}

//Data form

public DataForm()

{

this.InitializeComponent();

MaterialSkinManager.Instance.AddFormToManage((MaterialForm) this);

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

this.updateTabelValue(connection);

}

}

private void updateTabelValue(SqlConnection connection)

{

if (connection == null)

return;

this.materialListViewCustomers.Clear();

this.materialListViewOrders.Clear();

this.materialListViewRooms.Clear();

this.command = new SqlCommand("SelectAllCustomers", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewCustomers);

this.command = new SqlCommand("SelectALLOrders", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewOrders);

this.command = new SqlCommand("SelectAllRooms", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewRooms);

this.materialListViewCustomers.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

this.materialListViewOrders.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

this.materialListViewRooms.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

this.materialListViewCustomers.Columns[2].Width /= 2;

this.materialListViewCustomers.Columns[5].Width /= 2;

}

private void button9\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

this.materialListViewRooms.Clear();

connection.Open();

this.command = new SqlCommand("SelectAllRooms", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewRooms);

this.materialListViewRooms.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

}

}

private void DeleteCustomersWithoutOrders\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

this.command = new SqlCommand("RemoveCustomersWithoutOrders", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.command.ExecuteNonQuery();

this.updateTabelValue(connection);

}

}

private void RefreshCustomers\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

this.materialListViewCustomers.Clear();

connection.Open();

this.command = new SqlCommand("SelectAllCustomers", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewCustomers);

this.materialListViewCustomers.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

}

}

private void DeleteCustomer\_Click(object sender, EventArgs e)

{

if (this.materialListViewCustomers.SelectedItems.Count == 0)

return;

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

this.command = new SqlCommand("RemoveCustomer", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.command.Parameters.Add(new SqlParameter("@Customer\_id", (object) int.Parse(this.materialListViewCustomers.SelectedItems[0].Text)));

this.command.ExecuteNonQuery();

this.updateTabelValue(connection);

}

}

private void RefreshOrders\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

this.materialListViewOrders.Clear();

connection.Open();

this.command = new SqlCommand("SelectALLOrders", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.adapter = new SqlDataAdapter(this.command);

this.ds = new DataSet();

this.adapter.Fill(this.ds);

Program.FillList(this.ds.Tables[0], ref this.materialListViewOrders);

this.materialListViewOrders.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

}

}

private void button8\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

this.command = new SqlCommand("RemoveOldOrders", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.command.ExecuteNonQuery();

this.updateTabelValue(connection);

}

}

private void RemoveOrder\_Click(object sender, EventArgs e)

{

if (this.materialListViewOrders.SelectedItems.Count == 0)

return;

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

this.command = new SqlCommand("RemoveOrder", connection);

this.command.CommandType = CommandType.StoredProcedure;

this.command.Parameters.Add(new SqlParameter("@OrderId", (object) int.Parse(this.materialListViewOrders.SelectedItems[0].Text)));

this.command.ExecuteNonQuery();

this.updateTabelValue(connection);

}

}

private void AddNewCustomer\_Click(object sender, EventArgs e)

{

AddCustomerForm addCustomerForm = new AddCustomerForm();

addCustomerForm.Deactivate += (EventHandler) ((x, y) =>

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

this.updateTabelValue(connection);

});

int num = (int) addCustomerForm.ShowDialog();

}

private void AddNewOrder\_Click(object sender, EventArgs e)

{

AddOrderForm addOrderForm = new AddOrderForm();

addOrderForm.Deactivate += (EventHandler) ((x, y) =>

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

this.updateTabelValue(connection);

});

int num = (int) addOrderForm.ShowDialog();

}

//Add Customer Form

public AddCustomerForm()

{

this.InitializeComponent();

MaterialSkinManager.Instance.AddFormToManage((MaterialForm) this);

}

private void Pasport\_Leave(object sender, EventArgs e)

{

if (this.Pasport.Text == "" || this.Pasport.TextLength == 14)

return;

this.Pasport.Text = "";

int num = (int) MessageBox.Show("length of pasport number must be 14");

}

private void materialRaisedButton1\_Click(object sender, EventArgs e)

{

if (this.Pasport.TextLength == 0 || this.textBox1.TextLength == 0 || (this.textBox2.TextLength == 0 || this.textBox3.TextLength == 0))

{

int num = (int) MessageBox.Show("all field mast'n be blunk");

}

else

{

using (SqlConnection connection = new SqlConnection(Program.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@name", (object) this.textBox3.Text));

sqlCommand.Parameters.Add(new SqlParameter("@sername", (object) this.textBox2.Text));

sqlCommand.Parameters.Add(new SqlParameter("@patronymic", (object) this.textBox1.Text));

sqlCommand.Parameters.Add(new SqlParameter("@dateOfBirth", (object) this.DateOfBirth.Value.Date));

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) this.Pasport.Text));

sqlCommand.ExecuteNonQuery();

}

this.Close();

}

}

// Add Order form

public AddOrderForm()

{

this.InitializeComponent();

this.Booked.Checked = true;

MaterialSkinManager.Instance.AddFormToManage((MaterialForm) this);

using (SqlConnection connection = new SqlConnection(this.connectionString))

this.updateValue(connection);

}

private void updateValue(SqlConnection connection)

{

SqlCommand selectCommand1 = new SqlCommand("SelectAllCustomers", connection);

selectCommand1.CommandType = CommandType.StoredProcedure;

SqlDataAdapter sqlDataAdapter1 = new SqlDataAdapter(selectCommand1);

DataSet dataSet1 = new DataSet();

DataSet dataSet2 = dataSet1;

sqlDataAdapter1.Fill(dataSet2);

Program.FillList(dataSet1.Tables[0], ref this.materialListView1);

SqlCommand selectCommand2 = new SqlCommand("SelectAllFreeRooms", connection);

selectCommand2.CommandType = CommandType.StoredProcedure;

SqlDataAdapter sqlDataAdapter2 = new SqlDataAdapter(selectCommand2);

DataSet dataSet3 = new DataSet();

DataSet dataSet4 = dataSet3;

sqlDataAdapter2.Fill(dataSet4);

Program.FillList(dataSet3.Tables[0], ref this.materialListView2);

this.materialListView1.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

this.materialListView2.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

}

private void button1\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

this.materialListView1.Clear();

this.materialListView2.Clear();

this.updateValue(connection);

}

}

private void AddNewOrder\_Click(object sender, EventArgs e)

{

if (this.materialListView1.SelectedItems.Count == 0)

{

int num1 = (int) MessageBox.Show("not selected customer");

}

else if (this.materialListView2.SelectedItems.Count == 0)

{

int num2 = (int) MessageBox.Show("not selected any room");

}

else if (this.dateTimePicker1.Value > this.dateTimePicker2.Value)

{

int num3 = (int) MessageBox.Show("exit date mast be greater than registration date");

}

else

{

using (SqlConnection connection = new SqlConnection(this.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddOrder", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@registrationDate", (object) this.dateTimePicker1.Value.Date));

sqlCommand.Parameters.Add(new SqlParameter("@exitDate", (object) this.dateTimePicker2.Value.Date));

sqlCommand.Parameters.Add(new SqlParameter("@state", (object) (this.Booked.Checked ? 2 : 3)));

sqlCommand.Parameters.Add(new SqlParameter("@room", (object) this.materialListView2.SelectedItems[0].Text));

sqlCommand.Parameters.Add(new SqlParameter("@customer", (object) this.materialListView1.SelectedItems[0].Text));

sqlCommand.ExecuteNonQuery();

this.Close();

}

}

}

//Logic Form

public LogickForm(ref Timer timer, DateTime end)

{

this.InitializeComponent();

this.ui = timer;

this.end = end;

this.problemsOrders = new ProblemsOrdersForm(ref timer.getLogList());

this.problemsOrders.Show();

this.problemsOrders.Visible = false;

}

private void Simulation()

{

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("ActiveOrdersCount", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

this.counts[0] = (int) sqlCommand.ExecuteScalar();

}

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("ActiveCustomersCount", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

this.counts[1] = (int) sqlCommand.ExecuteScalar();

}

DateTime now = DateTime.Now;

for (DateTime dateTime = DateTime.Now; dateTime < this.end; dateTime = dateTime.AddHours(4.0))

{

int percent = 100 \* (int) dateTime.Subtract(now).TotalHours / (int) this.end.Subtract(now).TotalHours;

lock (this)

{

this.ui.UpdateProgressBar(percent);

this.ui.UpdateRooms();

this.ui.UpdateDate(dateTime);

}

++this.totalOrders;

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("ClearRoomAtDate", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@CurentDate", (object) string.Format("{0}/{1}/{2}", (object) dateTime.Month, (object) dateTime.Day, (object) dateTime.Year)));

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

if (sqlDataReader.HasRows)

{

while (sqlDataReader.Read())

this.ui.AddLog(new ListViewItem(sqlDataReader.GetString(0))

{

SubItems = {

sqlDataReader.GetInt32(1).ToString(),

"Leave"

}

});

}

}

this.MakeOrder(dateTime);

do

;

while (this.problemsOrders.Visible);

}

this.ui.UpdateProgressBar(100);

this.counts[4] = LogickForm.newOrdersCount;

this.counts[5] = this.totalOrders;

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("ActiveOrdersCount", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

this.counts[2] = (int) sqlCommand.ExecuteScalar();

}

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("ActiveCustomersCount", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

this.counts[3] = (int) sqlCommand.ExecuteScalar();

}

this.MakeWord();

}

private bool UpdateData()

{

LogickForm.Order problemOrder = this.problemOrder;

while (problemOrder.roomType != LogickForm.Order.RoomType.Suite)

{

problemOrder.roomType = problemOrder.roomType != LogickForm.Order.RoomType.DoubleWhithSofa ? (problemOrder.roomType != LogickForm.Order.RoomType.JuniorSuite ? (problemOrder.roomType == LogickForm.Order.RoomType.single ? problemOrder.roomType + 2 : problemOrder.roomType + 1) : LogickForm.Order.RoomType.Suite) : LogickForm.Order.RoomType.JuniorSuite;

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("GetAllFreeRoomsOfType", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@TypeOfRoom", (object) (int) problemOrder.roomType));

SqlDataReader reader = sqlCommand.ExecuteReader();

if (reader.HasRows)

{

problemOrder.room = this.GetRandomRoom(reader);

reader.Close();

connection.Close();

problemOrder.Price = (float) ((double) (problemOrder.End - problemOrder.Begin).Days \* (double) LogickForm.GetRoomPrice((int) problemOrder.roomType) \* 0.850000023841858);

this.problemOrder = problemOrder;

return true;

}

}

}

return false;

}

private LogickForm.Customer MakeCustomer()

{

LogickForm.Customer customer = new LogickForm.Customer();

Random random = new Random();

if (random.Next(0, 1) == 0)

{

customer.name = this.nameMale[random.Next(0, this.nameMale.Length)];

customer.sername = this.sername[random.Next(0, this.sername.Length)];

}

else

{

customer.name = this.nameFemail[random.Next(0, this.nameFemail.Length)];

customer.sername = this.sername[random.Next(0, this.sername.Length)] + "a";

}

customer.patronymic = this.patronymics[random.Next(0, this.patronymics.Length)];

customer.dateOfBirth = DateTime.Now;

customer.dateOfBirth = customer.dateOfBirth.AddDays((double) -random.Next(0, 30));

customer.dateOfBirth = customer.dateOfBirth.AddMonths(random.Next(0, 12));

customer.dateOfBirth = customer.dateOfBirth.AddYears(-random.Next(1, 100));

for (int index = 0; index < 14; ++index)

customer.pasportNumber += Convert.ToChar(random.Next(65, 90)).ToString();

return customer;

}

private void MakeOrder(DateTime curDate)

{

LogickForm.Order order = new LogickForm.Order(curDate);

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("GetAllFreeRoomsOfType", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@TypeOfRoom", (object) (int) order.roomType));

SqlDataReader reader = sqlCommand.ExecuteReader();

if (!reader.HasRows)

{

reader.Close();

connection.Close();

if (order.roomType == LogickForm.Order.RoomType.Suite)

return;

LogickForm.Order oldOrder = new LogickForm.Order(order);

this.problemOrder = order;

if (this.UpdateData())

{

this.problemCustomer = this.MakeCustomer();

lock (this.problemsOrders)

{

this.problemsOrders.Customer = this.problemCustomer;

this.problemsOrders.Order = this.problemOrder;

this.problemsOrders.AddToForm(oldOrder);

this.problemsOrders.Visible = true;

}

this.problemCustomer = (LogickForm.Customer) null;

this.problemOrder = (LogickForm.Order) null;

}

else

this.problemOrder = (LogickForm.Order) null;

}

else

{

order.room = this.GetRandomRoom(reader);

reader.Close();

connection.Close();

this.AddToDB(order, this.MakeCustomer());

}

}

}

private int GetRandomRoom(SqlDataReader reader)

{

if (reader == null || !reader.HasRows || reader.VisibleFieldCount == 0)

return 0;

reader.Read();

return reader.GetInt32(0);

}

private void AddToDB(LogickForm.Order order, LogickForm.Customer customer)

{

this.ui.AddLog(new ListViewItem(customer.sername + " " + customer.name)

{

SubItems = {

order.room.ToString(),

order.ordersType.ToString()

}

});

++LogickForm.newOrdersCount;

LogickForm.CustomerString = LogickForm.CustomerString + (object) customer + "\r";

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@name", (object) customer.name));

sqlCommand.Parameters.Add(new SqlParameter("@sername", (object) customer.sername));

sqlCommand.Parameters.Add(new SqlParameter("@patronymic", (object) customer.patronymic));

sqlCommand.Parameters.Add(new SqlParameter("@dateOfBirth", (object) customer.dateOfBirth));

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) customer.pasportNumber));

sqlCommand.ExecuteNonQuery();

}

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("FindCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) customer.pasportNumber));

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

sqlDataReader.Read();

order.customer = int.Parse(sqlDataReader.GetValue(0).ToString());

sqlDataReader.Close();

}

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddOrder", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Clear();

sqlCommand.Parameters.Add(new SqlParameter("@registrationDate", (object) order.Begin));

sqlCommand.Parameters.Add(new SqlParameter("@exitDate", (object) order.End));

sqlCommand.Parameters.Add(new SqlParameter("@state", (object) (int) order.ordersType));

sqlCommand.Parameters.Add(new SqlParameter("@room", (object) order.room));

sqlCommand.Parameters.Add(new SqlParameter("@customer", (object) order.customer));

sqlCommand.ExecuteNonQuery();

}

}

public static float GetRoomPrice(int roomType)

{

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand(nameof (GetRoomPrice), connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@type", (object) roomType));

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

sqlDataReader.Read();

return (float) (int) (Decimal) sqlDataReader.GetSqlMoney(0);

}

}

private void MakeWord()

{

// ISSUE: variable of a compiler-generated type

Microsoft.Office.Interop.Word.Application instance = (Microsoft.Office.Interop.Word.Application) Activator.CreateInstance(Marshal.GetTypeFromCLSID(new Guid("000209FF-0000-0000-C000-000000000046")));

instance.Visible = false;

// ISSUE: variable of a compiler-generated type

Documents documents = instance.Documents;

object obj1 = (object) (System.Windows.Forms.Application.StartupPath.ToString() + "/Template.docx");

ref object local1 = ref obj1;

object missing1 = System.Type.Missing;

ref object local2 = ref missing1;

object missing2 = System.Type.Missing;

ref object local3 = ref missing2;

object missing3 = System.Type.Missing;

ref object local4 = ref missing3;

object missing4 = System.Type.Missing;

ref object local5 = ref missing4;

object missing5 = System.Type.Missing;

ref object local6 = ref missing5;

object missing6 = System.Type.Missing;

ref object local7 = ref missing6;

object missing7 = System.Type.Missing;

ref object local8 = ref missing7;

object missing8 = System.Type.Missing;

ref object local9 = ref missing8;

object missing9 = System.Type.Missing;

ref object local10 = ref missing9;

object missing10 = System.Type.Missing;

ref object local11 = ref missing10;

object missing11 = System.Type.Missing;

ref object local12 = ref missing11;

object missing12 = System.Type.Missing;

ref object local13 = ref missing12;

object missing13 = System.Type.Missing;

ref object local14 = ref missing13;

object missing14 = System.Type.Missing;

ref object local15 = ref missing14;

object missing15 = System.Type.Missing;

ref object local16 = ref missing15;

// ISSUE: reference to a compiler-generated method

// ISSUE: variable of a compiler-generated type

Document document1 = documents.Open(ref local1, ref local2, ref local3, ref local4, ref local5, ref local6, ref local7, ref local8, ref local9, ref local10, ref local11, ref local12, ref local13, ref local14, ref local15, ref local16);

this.ReplaceWordStub("{Заказы до}", this.counts[0].ToString(), document1);

this.ReplaceWordStub("{Клиенты до}", this.counts[1].ToString(), document1);

this.ReplaceWordStub("{новые клиенты}", LogickForm.CustomerString, document1);

ListView.ListViewItemCollection items = this.ui.getLogList().Items;

string text = "";

foreach (ListViewItem listViewItem in items)

{

text = text + "клиент: " + listViewItem.Text + "\r";

text = text + "комната: " + listViewItem.SubItems[1].Text + "\r";

text = text + "Действие: " + listViewItem.SubItems[2].Text + "\r\r";

}

Console.WriteLine(text);

this.ReplaceWordStub("{история заказов}", text, document1);

this.ReplaceWordStub("{заказов после}", this.counts[2].ToString(), document1);

this.ReplaceWordStub("{клиентов после}", this.counts[3].ToString(), document1);

this.ReplaceWordStub("{добавлено заказов}", this.counts[4].ToString(), document1);

this.ReplaceWordStub("{обработано заказов}", this.counts[5].ToString(), document1);

// ISSUE: variable of a compiler-generated type

Document document2 = document1;

object obj2 = (object) "Result.docx";

ref object local17 = ref obj2;

object missing16 = System.Type.Missing;

ref object local18 = ref missing16;

object missing17 = System.Type.Missing;

ref object local19 = ref missing17;

object missing18 = System.Type.Missing;

ref object local20 = ref missing18;

object missing19 = System.Type.Missing;

ref object local21 = ref missing19;

object missing20 = System.Type.Missing;

ref object local22 = ref missing20;

object missing21 = System.Type.Missing;

ref object local23 = ref missing21;

object missing22 = System.Type.Missing;

ref object local24 = ref missing22;

object missing23 = System.Type.Missing;

ref object local25 = ref missing23;

object missing24 = System.Type.Missing;

ref object local26 = ref missing24;

object missing25 = System.Type.Missing;

ref object local27 = ref missing25;

object missing26 = System.Type.Missing;

ref object local28 = ref missing26;

object missing27 = System.Type.Missing;

ref object local29 = ref missing27;

object missing28 = System.Type.Missing;

ref object local30 = ref missing28;

object missing29 = System.Type.Missing;

ref object local31 = ref missing29;

object missing30 = System.Type.Missing;

ref object local32 = ref missing30;

object missing31 = System.Type.Missing;

ref object local33 = ref missing31;

// ISSUE: reference to a compiler-generated method

document2.SaveAs2(ref local17, ref local18, ref local19, ref local20, ref local21, ref local22, ref local23, ref local24, ref local25, ref local26, ref local27, ref local28, ref local29, ref local30, ref local31, ref local32, ref local33);

instance.Visible = true;

}

private void ReplaceWordStub(string ReplaceStub, string text, Document document)

{

// ISSUE: variable of a compiler-generated type

Microsoft.Office.Interop.Word.Range content = document.Content;

// ISSUE: reference to a compiler-generated method

content.Find.ClearFormatting();

if (text.Length < (int) byte.MaxValue)

{

// ISSUE: variable of a compiler-generated type

Find find = content.Find;

object obj1 = (object) ReplaceStub;

ref object local1 = ref obj1;

object missing1 = System.Type.Missing;

ref object local2 = ref missing1;

object missing2 = System.Type.Missing;

ref object local3 = ref missing2;

object missing3 = System.Type.Missing;

ref object local4 = ref missing3;

object missing4 = System.Type.Missing;

ref object local5 = ref missing4;

object missing5 = System.Type.Missing;

ref object local6 = ref missing5;

object missing6 = System.Type.Missing;

ref object local7 = ref missing6;

object missing7 = System.Type.Missing;

ref object local8 = ref missing7;

object missing8 = System.Type.Missing;

ref object local9 = ref missing8;

object obj2 = (object) text;

ref object local10 = ref obj2;

object missing9 = System.Type.Missing;

ref object local11 = ref missing9;

object missing10 = System.Type.Missing;

ref object local12 = ref missing10;

object missing11 = System.Type.Missing;

ref object local13 = ref missing11;

object missing12 = System.Type.Missing;

ref object local14 = ref missing12;

object missing13 = System.Type.Missing;

ref object local15 = ref missing13;

// ISSUE: reference to a compiler-generated method

find.Execute(ref local1, ref local2, ref local3, ref local4, ref local5, ref local6, ref local7, ref local8, ref local9, ref local10, ref local11, ref local12, ref local13, ref local14, ref local15);

}

else

{

int length = text.Length / (int) byte.MaxValue + 1;

string[] strArray = new string[length];

string str1 = "";

for (int index = 0; index < length; ++index)

{

strArray[index] = "{t" + (object) index + "}";

str1 += strArray[index];

}

// ISSUE: reference to a compiler-generated method

content.Find.ClearFormatting();

// ISSUE: variable of a compiler-generated type

Find find1 = content.Find;

object obj1 = (object) ReplaceStub;

ref object local1 = ref obj1;

object missing1 = System.Type.Missing;

ref object local2 = ref missing1;

object missing2 = System.Type.Missing;

ref object local3 = ref missing2;

object missing3 = System.Type.Missing;

ref object local4 = ref missing3;

object missing4 = System.Type.Missing;

ref object local5 = ref missing4;

object obj2 = System.Type.Missing;

ref object local6 = ref obj2;

object missing5 = System.Type.Missing;

ref object local7 = ref missing5;

object missing6 = System.Type.Missing;

ref object local8 = ref missing6;

object missing7 = System.Type.Missing;

ref object local9 = ref missing7;

object obj3 = (object) str1;

ref object local10 = ref obj3;

object missing8 = System.Type.Missing;

ref object local11 = ref missing8;

object missing9 = System.Type.Missing;

ref object local12 = ref missing9;

object missing10 = System.Type.Missing;

ref object local13 = ref missing10;

object missing11 = System.Type.Missing;

ref object local14 = ref missing11;

object missing12 = System.Type.Missing;

ref object local15 = ref missing12;

// ISSUE: reference to a compiler-generated method

find1.Execute(ref local1, ref local2, ref local3, ref local4, ref local5, ref local6, ref local7, ref local8, ref local9, ref local10, ref local11, ref local12, ref local13, ref local14, ref local15);

for (int index = 0; index < length; ++index)

{

string str2 = "";

if (text.Length != 0)

{

foreach (char ch in text.Take<char>(text.Length / (int) byte.MaxValue != 0 ? (int) byte.MaxValue : text.Length))

str2 += ch.ToString();

Console.Write(str2);

text = text.Remove(0, text.Length / (int) byte.MaxValue != 0 ? (int) byte.MaxValue : text.Length);

}

// ISSUE: reference to a compiler-generated method

content.Find.ClearFormatting();

// ISSUE: variable of a compiler-generated type

Find find2 = content.Find;

missing12 = (object) strArray[index];

ref object local16 = ref missing12;

missing11 = System.Type.Missing;

ref object local17 = ref missing11;

missing10 = System.Type.Missing;

ref object local18 = ref missing10;

missing9 = System.Type.Missing;

ref object local19 = ref missing9;

missing8 = System.Type.Missing;

ref object local20 = ref missing8;

obj3 = System.Type.Missing;

ref object local21 = ref obj3;

missing7 = System.Type.Missing;

ref object local22 = ref missing7;

missing6 = System.Type.Missing;

ref object local23 = ref missing6;

missing5 = System.Type.Missing;

ref object local24 = ref missing5;

obj2 = (object) str2;

ref object local25 = ref obj2;

missing4 = System.Type.Missing;

ref object local26 = ref missing4;

missing3 = System.Type.Missing;

ref object local27 = ref missing3;

missing2 = System.Type.Missing;

ref object local28 = ref missing2;

missing1 = System.Type.Missing;

ref object local29 = ref missing1;

object missing13 = System.Type.Missing;

ref object local30 = ref missing13;

// ISSUE: reference to a compiler-generated method

find2.Execute(ref local16, ref local17, ref local18, ref local19, ref local20, ref local21, ref local22, ref local23, ref local24, ref local25, ref local26, ref local27, ref local28, ref local29, ref local30);

content = document.Content;

}

}

}

public void StartSimulation()

{

this.Simulation();

}

//Problem orders Form

public ProblemsOrdersForm(

LogickForm.Customer customer,

LogickForm.Order order,

LogickForm.Order old,

ref MaterialListView LogList)

{

this.InitializeComponent();

MaterialSkinManager.Instance.AddFormToManage((MaterialForm) this);

Control.CheckForIllegalCrossThreadCalls = false;

this.Customer = customer;

this.Order = order;

this.logList = LogList;

this.AddToForm(old);

}

public ProblemsOrdersForm(ref MaterialListView LogList)

{

this.InitializeComponent();

MaterialSkinManager.Instance.AddFormToManage((MaterialForm) this);

this.logList = LogList;

}

public void AddToForm(LogickForm.Order oldOrder)

{

this.CustomerName.Text = this.Customer.name;

this.lastName.Text = this.Customer.sername;

this.patronymic.Text = this.Customer.patronymic;

this.OrderType.Text = oldOrder.ordersType.ToString();

this.InitinalRoomType.Text = oldOrder.roomType.ToString();

this.InitialPrice.Text = oldOrder.Price.ToString();

this.ProposedRoomType.Text = this.Order.roomType.ToString();

this.ProposedPrice.Text = this.Order.Price.ToString();

}

private void AddToDB(LogickForm.Order order, LogickForm.Customer customer)

{

this.logList.Items.Add(new ListViewItem(customer.sername + " " + customer.name)

{

SubItems = {

order.room.ToString(),

order.ordersType.ToString()

}

});

++LogickForm.newOrdersCount;

using (SqlConnection connection = new SqlConnection(ProblemsOrdersForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@name", (object) customer.name));

sqlCommand.Parameters.Add(new SqlParameter("@sername", (object) customer.sername));

sqlCommand.Parameters.Add(new SqlParameter("@patronymic", (object) customer.patronymic));

sqlCommand.Parameters.Add(new SqlParameter("@dateOfBirth", (object) customer.dateOfBirth));

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) customer.pasportNumber));

sqlCommand.ExecuteNonQuery();

}

using (SqlConnection connection = new SqlConnection(ProblemsOrdersForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("FindCustomer", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Add(new SqlParameter("@pasportNumber", (object) customer.pasportNumber));

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

sqlDataReader.Read();

order.customer = int.Parse(sqlDataReader.GetValue(0).ToString());

sqlDataReader.Close();

}

using (SqlConnection connection = new SqlConnection(ProblemsOrdersForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("AddOrder", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.Clear();

sqlCommand.Parameters.Add(new SqlParameter("@registrationDate", (object) order.Begin));

sqlCommand.Parameters.Add(new SqlParameter("@exitDate", (object) order.End));

sqlCommand.Parameters.Add(new SqlParameter("@state", (object) (int) order.ordersType));

sqlCommand.Parameters.Add(new SqlParameter("@room", (object) order.room));

sqlCommand.Parameters.Add(new SqlParameter("@customer", (object) order.customer));

sqlCommand.ExecuteNonQuery();

}

}

private void Accept\_Click(object sender, EventArgs e)

{

if (this.CustomerName.Text != "")

this.AddToDB(this.Order, this.Customer);

this.Visible = false;

}

private void Refuse\_Click(object sender, EventArgs e)

{

this.Visible = false;

}

// Timer

public void UpdateRooms()

{

using (SqlConnection connection = new SqlConnection(LogickForm.connectionString))

{

connection.Open();

SqlCommand sqlCommand = new SqlCommand("SelectAllRooms", connection);

sqlCommand.CommandType = CommandType.StoredProcedure;

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

int roomNumber = sqlDataReader.GetInt32(0);

string str = sqlDataReader.GetString(2);

Color color = !(str == "Free") ? (!(str == "Booked") ? Color.OrangeRed : Color.Yellow) : Color.Lime;

((IEnumerable<Label>) this.Rooms).First<Label>((Func<Label, bool>) (x => int.Parse(x.Text) == roomNumber)).Invoke((Delegate) (() => ((IEnumerable<Label>) this.Rooms).First<Label>((Func<Label, bool>) (x => int.Parse(x.Text) == roomNumber)).BackColor = color));

}

}

}

public void UpdateProgressBar(int percent)

{

this.materialProgressBar1.Value = percent;

}

public void UpdateDate(DateTime cur)

{

this.CurrentTime.Text = cur.ToString();

}

public void AddLog(ListViewItem item)

{

this.LogList.Items.Add(item);

}

public ref MaterialListView getLogList()

{

return ref this.LogList;

}

# Приложение Б (необязательное) Результаты тестирования

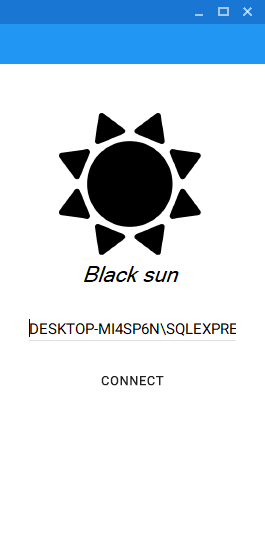


Рисунок Б.1 – форма авторизации

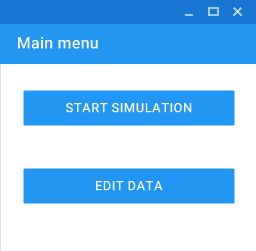


Рисунок Б.2 – главная форма

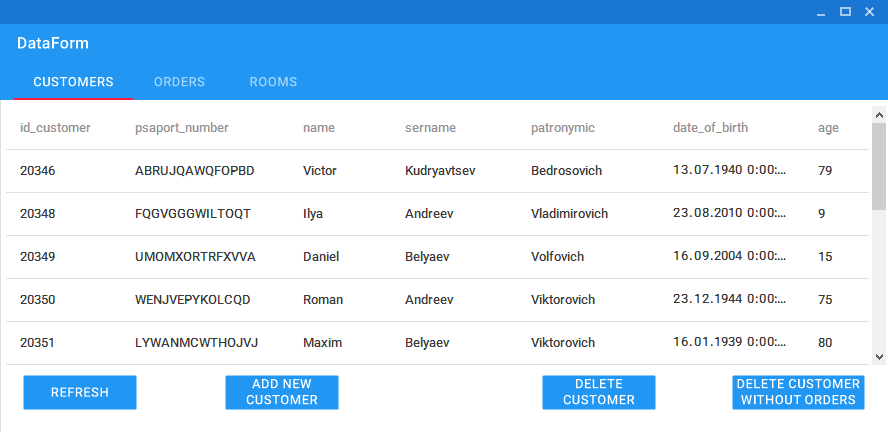


Рисунок Б.3 – форма редактирования данных на вкладке «CUSTOMERS»

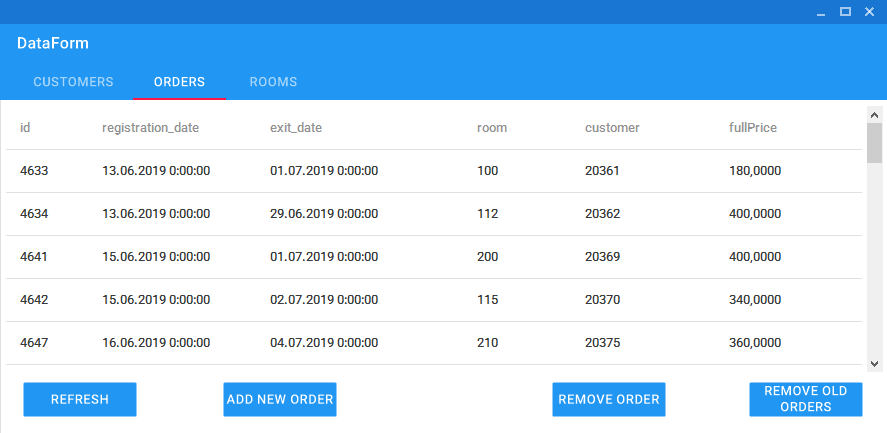


Рисунок Б.4 – форма редактирования данных на вкладке «ORDERS»

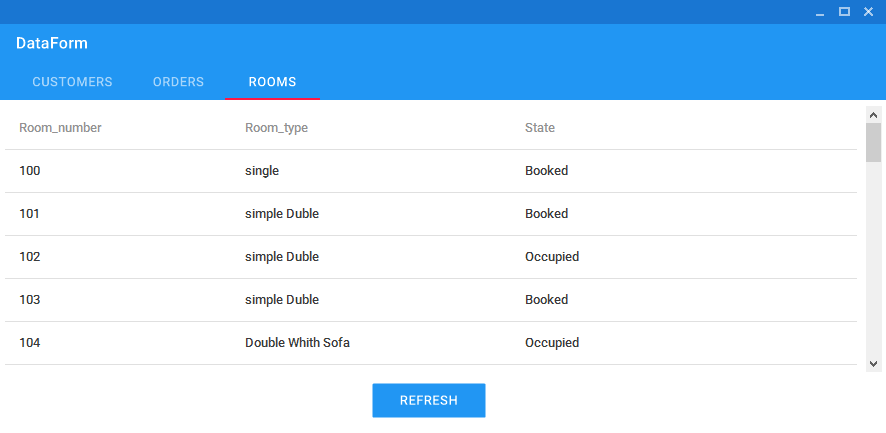


Рисунок Б.5 – форма редактирования данных на вкладке «ROOMS»

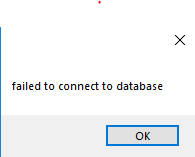


Рисунок Б.6 – сообщение об ошибке при неудачной попытке подключения к базе данных.

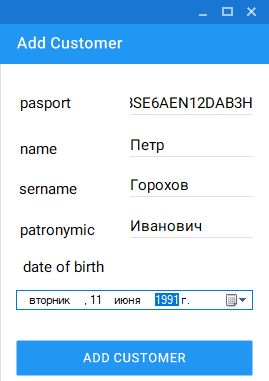


Рисунок Б.7 – Добавление нового клиента

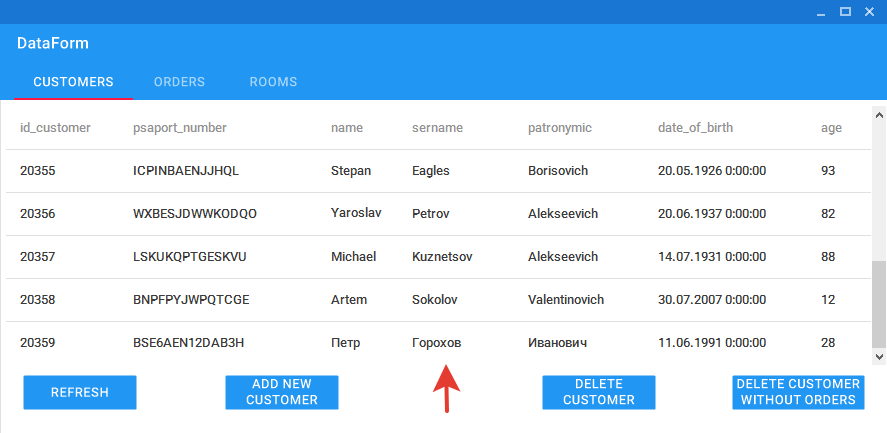


Рисунок Б.8 – результат работы добавления нового клиента

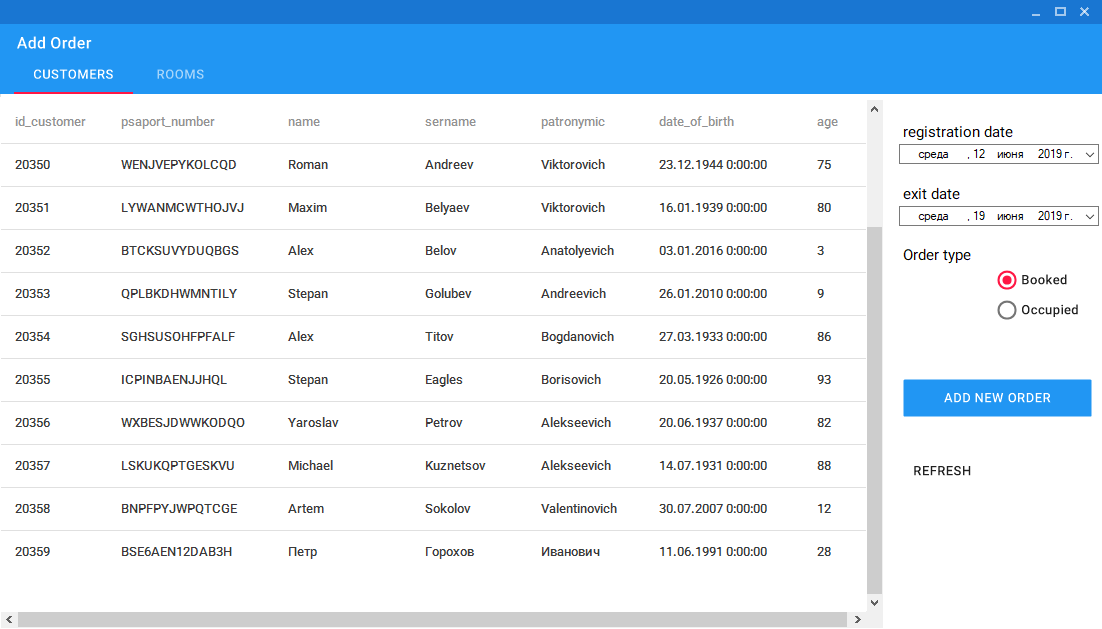


Рисунок Б.9 – форма добавления клиента

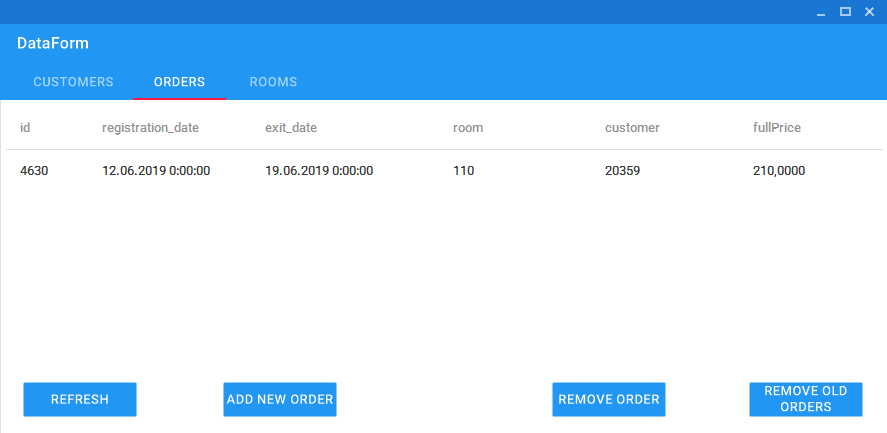


Рисунок Б.10 – добавленный заказ

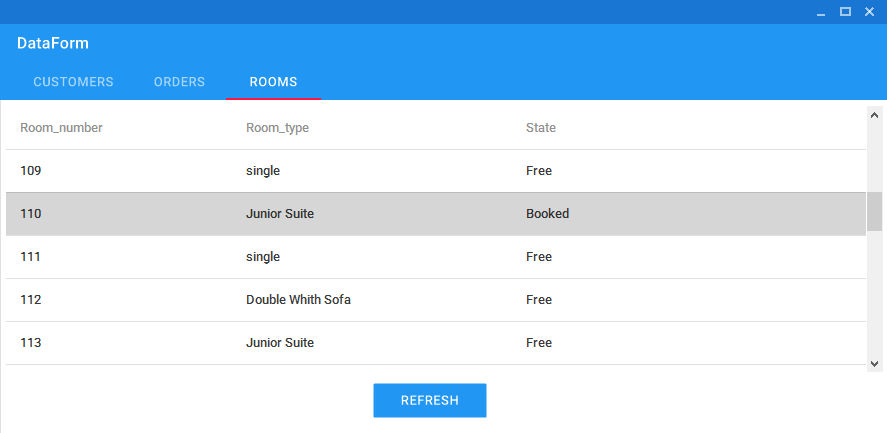


Рисунок Б.11 – измененное состояние комнаты в соответствии с заказом