|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***\_ИУ-КФ «Информатика и управление»\_\_\_\_\_\_******\_\_\_***

**КАФЕДРА** ***\_\_ИУ5-КФ «Системы обработки информации» \_\_\_\_***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

***«Корпоративный чат»***

по дисциплине ***Сетевые технологии***

Студент гр.СОИ.Б-51 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_Семин Т.С.\_)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Вершинин Е.В.)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2020

***Министерство науки и высшего образования Российской Федерации***

*Калужский филиал  
 федерального государственного бюджетного   
образовательного учреждения высшего образования*

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***

***(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУ5-КФ\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_(Вершинин Е.В.)

«\_ 8 \_»\_\_\_\_февраля\_\_\_\_2021г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине ***Сетевые технологии***

Студент **Семин Т.С. ИУК5-61Б**

(фамилия, инициалы, индекс группы)

Руководитель **Вершинин Е.В.** \_\_\_\_\_\_

(фамилия, инициалы)

График выполнения работы: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_14\_нед.

1. ***Тема курсовой работы***

***«Свободный чат»***

1. ***Техническое задание***

*«Написать клиент-серверное приложение на сокетах. Сервер и клиент – WPF приложения. Реализовать чат с поддержкой множества участников и возможностью передачи файлов. У сервера ведутся логи»*

1. ***Оформление курсовой работы***

3.1. Расчетно-пояснительная записка на\_\_\_\_\_\_\_\_\_\_ листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)\_*\_Презентация,\_*

*изображение интерфейса клиента и сервера, алгоритмы работы.* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_8\_\_\_»\_\_февраля\_2021г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_Вершинин Е.В.\_\_/

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_Семин Т.С.\_\_\_\_\_\_/ «\_\_8\_\_\_» \_\_февраля\_\_ 2021г.

(подпись) (Ф.И.О.)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc59022537)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc59022538)

[1.1. Общие сведения 5](#_Toc59022539)

[1.1.1. Наименование 5](#_Toc59022540)

[1.1.2. Исполнитель 5](#_Toc59022541)

[1.1.3. Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ: 5](#_Toc59022542)

[1.1.4. Плановые сроки по созданию приложения 5](#_Toc59022543)

[1.2. Назначение и цели создания системы 5](#_Toc59022544)

[1.2.1. Назначение системы 5](#_Toc59022545)

[1.2.2. Цели создания системы 5](#_Toc59022546)

[1.3. Требования к системе 5](#_Toc59022547)

[1.3.1. Требования к системе в целом 5](#_Toc59022548)

[1.3.2. Требования к надежности 6](#_Toc59022549)

[1.3.3. Требования к функциям, выполняемым системой 6](#_Toc59022550)

[1.3.4. Требования к информационному обеспечению системы 6](#_Toc59022551)

[1.3.5. Требования к методическому обеспечению 6](#_Toc59022552)

[1.4. Состав и содержание работ по созданию(развитию) системы 6](#_Toc59022553)

[1.5. Порядок контроля и приемки системы 7](#_Toc59022554)

[1.5.1. Методы испытания системы 7](#_Toc59022555)

[1.5.2. Общие требования к приемке работ по стадиям 7](#_Toc59022556)

[1.5.3. Статус приемочной комиссии 7](#_Toc59022557)

[1.6. Требования к документированию 7](#_Toc59022558)

[2. НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 8](#_Toc59022559)

[2.1. Описание предметной области 8](#_Toc59022560)

[2.1.1. Общие сведения 8](#_Toc59022561)

[2.1.2. Анализ аналогов 9](#_Toc59022562)

[2.1.3. Концептуальная модель данных 12](#_Toc59022563)

[2.2. Сценарии использования 12](#_Toc59022564)

[2.3. Задача автоматизации 13](#_Toc59022565)

[2.4. Перечень задач, подлежащих решению в процессе разработки 13](#_Toc59022566)

[2.5. Обоснование выбора платформы для разработки 14](#_Toc59022567)

[3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ 15](#_Toc59022568)

[3.1. Разработка структуры приложения 15](#_Toc59022569)

[3.2. Разработка алгоритмов обработки информации 16](#_Toc59022570)

[3.3. Логическая схема базы данных 17](#_Toc59022571)

[3.4. Разработка интерфейса взаимодействия пользователя с системой 20](#_Toc59022572)

[3.5. Разработка архитектуры приложения 25](#_Toc59022573)

[3.6. Реализация функционирующего приложения 27](#_Toc59022574)

[4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 29](#_Toc59022575)

[4.1. Порядок развертывания системы 29](#_Toc59022576)

[4.2. Уровень подготовки пользователя 32](#_Toc59022577)

[4.3. Руководство пользователя 32](#_Toc59022578)

[4.4. Возможности улучшения 33](#_Toc59022579)

[4.5. Тестирование приложения 33](#_Toc59022580)

[ЗАКЛЮЧЕНИЕ 35](#_Toc59022581)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc59022582)

# ВВЕДЕНИЕ

В данной курсовой работе разрабатывается чат-клиент и чат-сервер для возможности общения большого количества людей. Любой человек, знающий адрес сервера, может присоединиться к чату и начать общение. Приложение являются оконными и содержат в себе понятный интерфейс и минимум действий для обеспечения максимально комфортного общения.

В техническом задании описаны основные требования к разрабатываемому приложению.

В исследовательской части выполнен анализ предметной области, проанализированы аналоги, обоснован выбор среды и средств разработки.

В конструкторской части приведен интерфейс приложения. Дано краткое описание использованных структур данных и архитектуры приложения.

В технологической части объяснены основные принципы взаимодействия с разработанным приложением. Выполнено тестирование.

В заключении сформированы выводы о проделанной работе.

В списке использованных источников перечислены ссылки на адреса использованных Интернет-ресурсов.

Работа имеет Х страниц и Х рисунок.

**1.** ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## **1.1. Общие сведения**

### **1.1.1. Наименование**

«Корпоративный чат»

### **1.1.2. Исполнитель**

Студент КФ МГТУ Им. Н.Э. Баумана Семин Тимофей гр. ИУК5-61Б

### **1.1.3. Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ:**

-Методические указания по выполнению курсовой работы по дисциплине “Сетевые технологии”, автором которых является Е.В. Вершинин.

-А также на основании Технического задания.

### **1.1.4. Плановые сроки по созданию приложения**

Начало работ – 1 сентября 2020 года, окончание работ -1 декабря 2020 года.

По мере выполнения работы результаты загружается на сервис github по адресу <https://github.com/J0hannex/ServiceToFixAndSail>, где заказчик может следить за выполнением работ и вносить коррективы, а также презентуются в личных встречах с заказчиком.

## **1.2. Назначение и цели создания системы**

### **1.2.1. Назначение системы**

Обеспечение общения группы лиц на расстоянии по сети интернет.

### **1.2.2. Цели создания системы**

Удобный способ обмена информацией или общения.

## **1.3. Требования к системе**

### **1.3.1. Требования к системе в целом**

Структуру систему составляют:

* Чат-клиент
* Чат-сервер

### **1.3.2. Требования к надежности**

В случае отказа сервера должен быть выведен лог ошибки и реализована возможность его повторного запуска. У клиентов должно быть обеспечено безотказное соединение с сервером и отсутствие потерянных пакетов.

### **1.3.3. Требования к функциям, выполняемым системой**

Сервер должен:

1. Осуществлять работу по ip адресу локальной машины.
2. Вести лог работы и статистику присоединенных/отключённых пользователей.

Клиент должен:

1. Осуществить соединение к чату по указанному адресу.
2. Дать возможность пользователю выбрать никнейм.
3. Осуществить передачу сообщения пользователя на сервер.
4. Отображать всех пользователей в сети.

### **1.3.4. Требования к информационному обеспечению системы**

Чат не является хранилищем данных, вся передаваемая информация является разовой и не должна храниться на сервере. При отключении пользователя у него ничего не остается. При отключении сервера, информация удаляется у всех.

### **1.3.5. Требования к методическому обеспечению**

Разработку системы требуется сопровождать расчетно-пояснительной запиской и графической частью. Расчетно-пояснительная записка состоит из следующих частей:

1. Техническое задание

2. Научно-Исследовательская часть.

3.Проектно-конструкторская часть.

4. Проектно-технологическая часть.

## **1.4. Состав и содержание работ по созданию(развитию) системы**

Первый этап – изучение предметной области, определение состава и структуры данных. В результате прохождения первого этапа определяются: логическая структура данных, ТЗ, описание объекта автоматизации и целей разработки.

Второй этап – анализ пользовательских сценариев работы. В результате создается пользовательский интерфейс.

Третий этап – анализ взаимодействия веб-приложения с серверной компонентой. В результате определяются: архитектура приложения и структура классов.

## **1.5. Порядок контроля и приемки системы**

### **1.5.1. Методы испытания системы**

Испытания проектируемой системы проводятся заказчиком.

### **1.5.2. Общие требования к приемке работ по стадиям**

Место проведения приемки работ КФ МГТУ им. Н.Э. Баумана.

Сроки проведения приёмки работ – 8 марта, 23 марта, 1 апреля, 15 апреля, 1 мая и другие даты, согласованные с заказчиком и исполнителем

### **1.5.3. Статус приемочной комиссии**

Прием работы на всех этапах осуществляется непосредственным руководителем курсовой работы. По окончанию разработки работоспособное приложение демонстрируется на комиссии.

## **1.6. Требования к документированию**

-ТЗ по проектируемой системе

-Расчетно-пояснительная записка

-Презентация Microsoft PowerPoint

# **2. НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

## **2.1. Постановка задачи проектирования**

В рамках курсовой разработать клиент-серверное приложение. Разработать и написать клиент с понятным и удобным для пользователя интерфейсом. Разработать сервер с удобным интерфейсом и ведением логов работы. Осуществить связь между клиентом и сервером посредством потокового неблокирующего сокета на протоколе TCP.

## **2.2. Описание предметной области**

### **2.2.1. Общие сведения**

Чат в нынешнее время уже не является каким-то необычным и неизвестным явлением. Огромное количество людей общаются посредством различных чат-приложений.

Причина такой популярности чатов заключается в том, что они убирают рамки расстояний и люди могут общаться на больших дистанциях практически как вживую. Любая передаваемая информация доставляется быстро по интернету.

Чаты бывают как текстовые, так и голосовые или видео-чаты.

Также они могут существовать как отдельные приложения, либо быть встроенными в какое-либо приложение, игру или браузер.

В продвинутых чатах определенные участники могут иметь привилегии, недоступные большинству. Например модераторы, которые могут удалять сообщения, нарушающие какие-либо правила.

Модерация чатов является ключевой частью, поскольку наличие оскорбительных сообщений всегда мешает общению или может нанести вред определенным людям.

В большинстве существующих чатов есть возможность приватного общения пользователей. Если кто-либо хочет поделиться какой-то важной информацией с определенным пользователем, и она не должна быть доставлена больше никому, он может написать личное сообщение нужному пользователю.

Для удобства пользователей чаты всегда должны показывать всех людей, которые в данный момент находятся в сети.

Понятие “чат” подразумевает в себе 2 сущности: клиенты и сервер. Без сервера клиенты не смогут общаться друг с другом. Когда пользователь отправляет сообщение, его получает сервер и отправляет его всем подключенным участникам (либо определённым лицам в случае персонального сообщения).

Сервер чата должен иметь хорошую аппаратную составляющую. Недостаточные по мощности серверы при большом количестве пользователей могут передавать информацию с задержкой либо сильно зависать и или выходить из строя.

Также для видео-чатов крайне важно наличие у пользователей широкополосного интернет-соединения для обеспечения непрерывной и качественной передачи видеотрансляции голоса.

### **2.2.2. Сведения о разрабатываемом приложении**

Разрабатываемый в данной курсовой работе чат является способом практического применения и улучшения навыков разработки приложений с использованием сокета. Он будет поддерживать только текстовые сообщения.

Форма чата – отдельное оконное приложение.

Обязательно должен присутствовать сервер, к которому будут подключаться клиенты и сами клиенты, имеющие необходимый набор функций, строго необходимых для осуществления общения.

Любая передаваемая информация пользователями – не конфиденциальная, и в чате не будет реализована возможность приватных сообщений.

Хранение истории сообщений также не предусматривается. Чат рассчитан на общение в текущее время для простого досуга. При выходе пользователя вся информация удаляется.

Сервер также не хранит никакой информации о диалогах в логах, кроме ошибок, которые могут возникнуть в процессе работы.

Система регистраций и авторизаций по логину/паролям не будет использоваться, поскольку чат является свободным.

Для упрощения работы система полномочий и модерации чатов не будет использоваться.

Пользователи могут выбрать любой ник при выходе на сервер и в любое время менять цвет, которым будет отображаться его них для всех участников.

Данный чат особенно подойдет для общения тех людей, которые желают оставаться анонимными.

В интерфейсе каждого участника будут отображаться все активные пользователи, и в случае присоединения или отключения кого-либо у всех будет отображено соответствующее сообщение от имени сервера.

У клиента и сервера обязательно должен отображаться статус работы. Например, сервер обязательно должен показывать, что он отключен или подключен. Если произойдет ошибка во время работы или на стадии запуска, то появится всплывающее сообщение и в лог будет записана необходимая информация.

Клиент тоже должен быть проинформирован о какой-либо ошибке. В случае если ошибка возникает при подключении, то обязательно должна быть указана возможная проблема. Как и в случае с сервером, должно быть написан статус – отключен/подключен, а ошибки или незаполненные/заполненные неправильно поля должны “заявить о себе” всплывающим сообщением.

Как и у клиента, так и у сервера в интерфейсе не должно быть лишних кнопок и надписей, а каждое поле или кнопка должны быть понятно подписаны, чтобы не возникло проблем с использованием приложения.

### **2.2.3. Технологии обработки и передачи информации**

Протокол TCP осуществляет передачу данных от клиента к серверу посредством преобразования их в поток байтов. На основе TCP можно написать свой протокол, который сможет отправлять не только само сообщение, а пакет данных: сообщение, никнейм отправителя и цвет никнейма, чтобы у всех отображался никнейм данного пользователя таким цветом, который он выбрал.

Поскольку TCP передает поток байтов данных, будет необходима сериализация необходимых данных пакетов при отправке и их диссериализация у получателей.

Как было упомянуто в пункте выше, чат не используется в целях передачи какой-либо секретной и конфиденциальной информации, поэтому передаваемая информация не будет шифроваться.

### **2.2.4. Функции, подлежащие автоматизации**

Поскольку предназначение чата – общение, в клиентской части необходим ряд автоматизаций для обеспечения максимально простого и комфортного общения.

Прежде всего следует автоматизировать сам процесс подключения на сервер – в интерфейсе клиентского приложения должно присутствовать поле для ввода IP адреса и порта целевого сервера. Помимо них должно еще быть поле ввода никнейма, чтобы каждый пользователь был уникален на сервере, и селектор цвета никнейма – чтобы пользователь ставил какой понравится.

Затем по одному нажатию кнопки подключиться в случае правильно введенного IP адреса и порта, клиент выполнит соединение с сервером.

Важно также автоматизировать отправку сообщений. Пользователю всего лишь необходимо ввести текст и нажать кнопку отправки, после чего информация отправится на сервер и сообщение данного пользователя увидят остальные.

Сам сервер тоже должен быть автоматизирован – включение происходит по одному клику, IP адрес берется автоматически, порт вводится вручную в соответствующее поле.

Во время работы сервер автоматически пишет в специальное поле всю необходимую информацию.

Для отключения сервера нужно нажать всего одну кнопку.

### **2.3. Анализ аналогов**

Существует огромное количество различных чатов, поэтому аналогов данного чата тоже много. Невозможно уже создать что-либо уникальное, поэтому не будет присутствовать никаких особенностей, которые будут выделять данный чат среди других.

Однако аналоги можно будет использовать в качестве примера и использовать их наилучшие особенности.

1. WhatsApp - популярная бесплатная система мгновенного обмена текстовыми сообщениями для мобильных и иных платформ с поддержкой голосовой связи и видеосвязи. Позволяет пересылать текстовые сообщения, изображения, видео, аудио, электронные документы и даже программные установки через Интернет. Имеет как мобильную версию, так и десктопную.

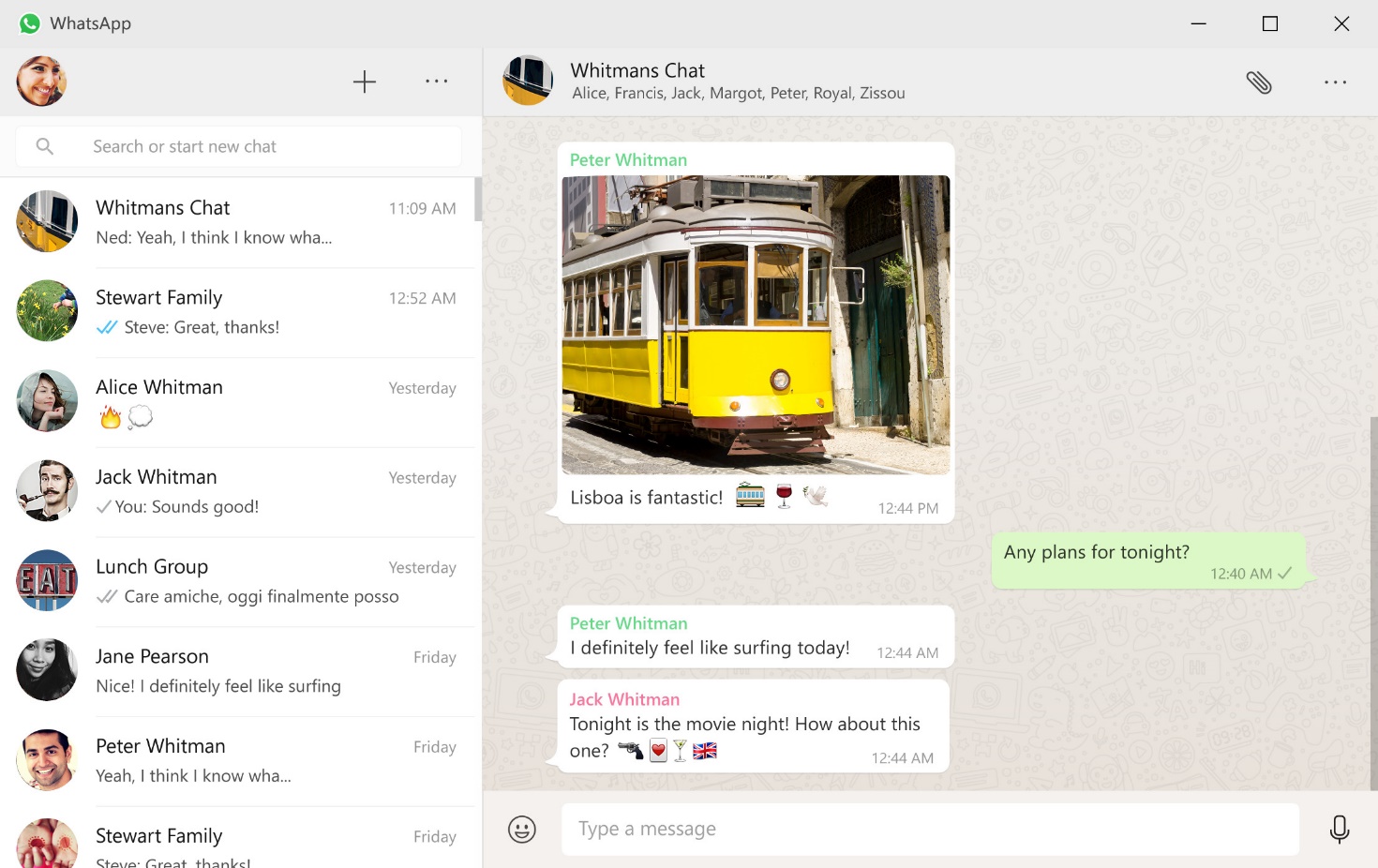


Рисунок 1 – Веб-версия WhatsApp

Мессенджер имеет очень простой и удобный дизайн. Чтобы пользоваться сервисом необходимо зарегистрироваться по номеру телефона. Сервис автоматически импортирует контакты. Добавление новых друзей так же по номеру телефона.

Имеется возможность персонализации профиля: выбор фотографии, статуса, имени.

Передаваемые сообщения имеют двойное шифрование. История диалогов хранится на устройстве пользователя в качестве базы данных.

Помимо личных сообщений можно создавать конференции на множество участников. Голосовые и аудиозвонки доступны как в личных сообщениях, так и конференциях.

1. Telegram – кроссплатформенный мессенджер с функциями VoIP, позволяющий обмениваться текстовыми, голосовыми и видеосообщениями, стикерами и фотографиями, файлами многих форматов. Также можно совершать видео и аудио-звонки, организовывать конференции и многопользовательские группы, и каналы.

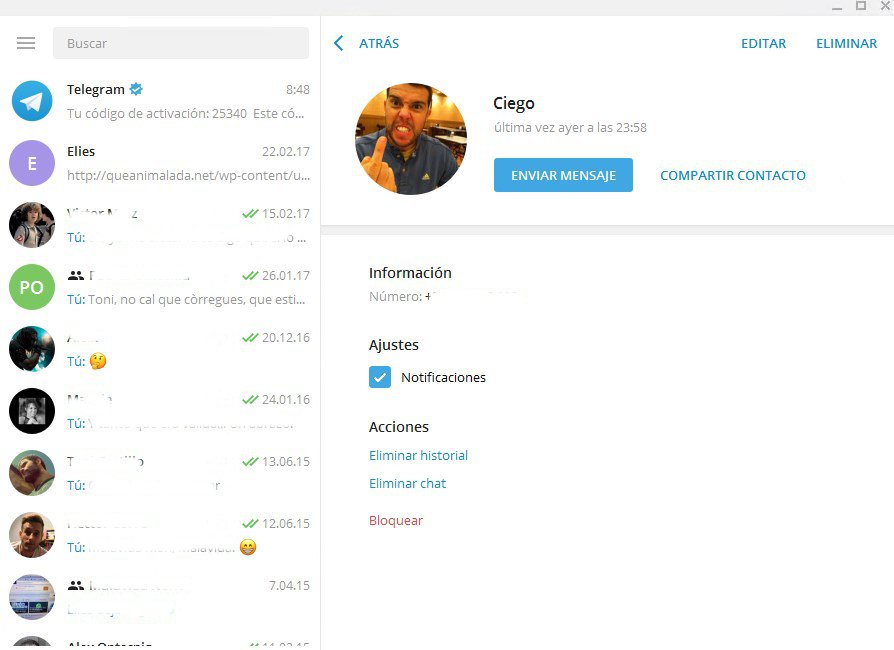


Рисунок 2 – Веб-версия Telegram

Telegram имеет практически схожий функционал и принцип работы, как и WhatsApp, однако обеспечивает не только хорошую защиту передаваемой информации благодаря шифрованию, но и позволяет участникам оставляться полностью анонимными для других пользователей и даже спец служб.

Добавлять друзей можно как по номеру телефона, так и по уникальному имени. Существующие контакты импортируются автоматически как в WhatsApp.

Одной из интересных особенностей является возможность создания новостного канала с большим числом участников, где оставлять сообщения может только владелец канала.

1. Discord – проприетарный бесплатный мессенджер с поддержкой VoIP, видеоконференций, предназначенный для использования различными сообществами по интересам, наиболее популярен у геймеров и учащихся.

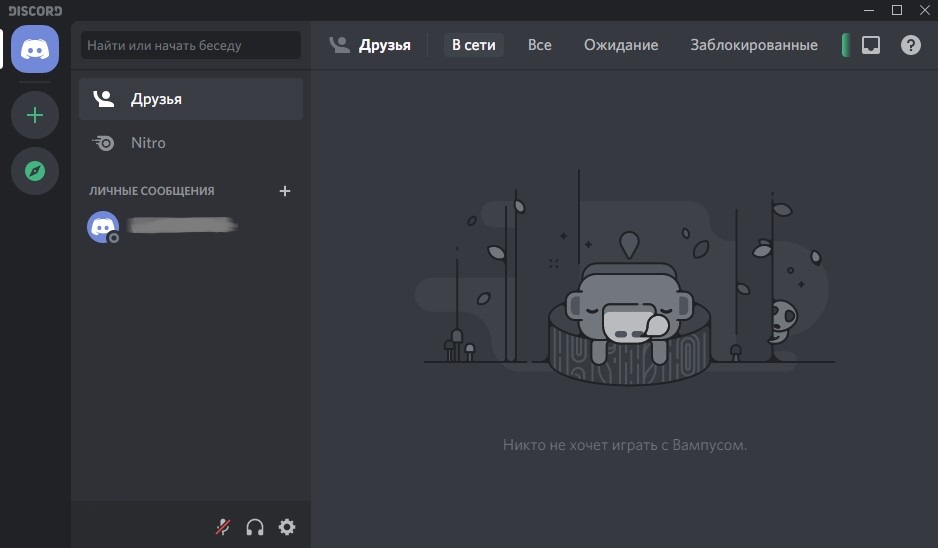


Рисунок 3 – Десктоп версия Discord

Данный мессенджер значительно отличается от предыдущих образцов как по внешнему оформлению, так и по целям использования.

Благодаря своей удобной голосовой и аудиосвязи больше остальных используется именно для такого типа общения. В отличие от WhatsApp и Telegram общаться по голосовой связи могут не 2 участника, а большое количество людей, из-за чего мессенджер пользуется большим спросом у геймеров.

Также присутствует возможность добавления друзей по уникальному имени и создание своих каналов или присоединение к существующим. На самих каналах публикуется какая-либо информация, показываются активные пользователи и текстовые чаты, а также присутствует возможность создания приватных или открытых комнат, в которых и происходит голосовое общение.

Выводы: представленные образцы являются не просто чатами, а мессенджерами. Имеют удобный и понятный интерфейс. Стиль интерфейса, непременно, можно использовать как пример при разработке своего чат-клиента.

## **2.4. Задачи, подлежащие решению в процессе разработки**

Первым делом необходимо написать клиент и сервер. Оба будут являться оконными приложениями. Разработку лучше начать с оформления дизайна, затем приступать к прописыванию действий кнопкам, а также использование полей для ввода и вывода кодом.

Взаимодействие клиента и сервера будет осуществляться посредством использования сокета. Поэтому надо применить сокет с необходимыми параметрами, а именно выбрать вид и тип сокета, адрес и порт работы.

После нужно уже реализовать остальной необходимый функционал на стороне клиента и сервера и написать определенные события (Events).

Также необходимо перехватывать и обрабатывать все возможные исключения.

Сервер должен автоматически ставить глобальный адрес текущего компьютера. Когда клиент подключается к серверу, у него должны быть заблокированы поля ввода адреса, порта и ника, а также кнопка “Подключиться”.

Следует написать алгоритмы работы приложения и действий пользователя.

## **2.5. Обоснование выбора инструментов и платформы для разработки**

В качестве языка разработки был выбран C# т. к. более удобен на основе имеющихся навыков и знаний.

Платформа разработки – WPF, поскольку является наиболее удобная для создания оконных приложений и имеет большой набор инструментов.

Основные преимущества WPF:

* **Веб-подобная модель компоновки:** WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высокодинамичного содержимого или к разным языкам.
* **Анимации:** в WPF нет необходимости использовать таймер для того, чтобы заставить форму перерисовать себя. Вместо этого доступна анимация — неотъемлемая часть платформы. Анимация определяется декларативными дескрипторами, и WPF запускает ее в действие автоматически.
* **Поддержка аудио и видео**: Windows Forms, были весьма ограничены в работе с мультимедиа. Однако WPF включает поддержку воспроизведения любого аудио или видеофайла, поддерживаемого проигрывателем Windows Media, позволяя воспроизводить более одного медиафайла одновременно. Что еще больше впечатляет — WPF предоставляет в распоряжение инструменты для интеграции видеосодержимого в остальную часть пользовательского интерфейса, позволяя выполнять такие экзотические трюки, как размещение видеоокна на поверхности вращающегося трехмерного куба.
* **Команды**: большинству пользователей известно, что не имеет значения, откуда они инициируют команду открытия (Open) — через меню или панель инструментов; конечный результат один и тот же. Теперь эта абстракция доступна коду — можно определять команды приложения в одном месте и привязывать их к множеству элементов управления.
* **Декларативный пользовательский интерфейс:** содержимое каждого окна сериализуется в виде XML-дескрипторов в документе XAML. Преимущество состоит в том, что пользовательский интерфейс полностью отделяется от кода, и дизайнеры графики могут использовать профессиональные инструменты для редактирования файлов XAML, улучшая внешний вид всего приложения.
* **Приложения на основе страниц:** используя WPF, можно строить браузер-подобные приложения, которые позволяют перемещаться по коллекции страниц, оснащенной кнопками навигации "вперед" и "назад". WPF автоматически обрабатывает все сложные детали, такие как хронология посещения страниц. Проект можно даже развернуть в виде браузерного приложения, которое выполняется внутри Internet Explorer.

# 3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ

## **3.1. Разработка структуры приложения**

В данной работе разрабатывается веб приложение на платформе ASP.NET – клиент-серверное приложение, где в качестве клиента выступает браузер, а сервером является веб-сервер.

Главная часть приложения содержится на стороне веб-сервера, который обрабатывает полученные запросы и формирует ответ, который будет отправлен пользователю (браузеру). При переходе пользователя по ссылке браузер отправляет запрос к серверу, затем сервер обрабатывает полученный запрос и вызывает скрипт, который формирует веб-страницу и отправляет ее клиенту по сети. На данном этапе, а если конкретнее – в клиентской части веб-приложения, используется браузер, т.к. именно он преобразовывает полученный ответ от сервера в графический интерфейс, который будет понятен и легко доступен пользователю. Именно через браузер осуществляется взаимодействие клиента с веб-приложением, в процессе нажатия на ссылки.

На сервере обязательно должны присутствовать БД и СУБД – специальное ПО, которое занимается хранением данных и их предоставлением в подходящий момент. В данной работе основное предназначение БД – хранение полученных от клиентов заявлений для их последующей обработки и автоматизации процесса работы. Серверная часть веб-приложения обращается к БД, извлекая при этом данные, необходимые для формирования запрашиваемой пользователем (клиентом) страницы.

Последовательность разработки приложения можно разбить на несколько пунктов:

1. *Создание таблиц БД;*
2. *Создание системы заполнения таблиц БД;*
3. *Разработка и написание алгоритмов обработки данных;*
4. *Создание интерфейса;*
5. *Написание алгоритмов отображения данных БД на клиентском уровне.*

## **3.2. Разработка алгоритмов обработки информации**

Для понимания работы механизма приложения ниже будут представлены фрагменты кода с их пояснением.



Рисунок 7 – Первый фрагмент кода

В коде, представленном на рисунке 7, присутствует метод Services(), в котором при помощи интерфейсов IEnumerable создаются 2 переменные. В них добавляются данные из БД. Здесь “EF.Services” и “EF.Clients” являются экземплярами БД, и от этих экземпляров уже данные отсылаются таблицам. Затем они с помощью интерфейсов перекидываются во ViewBag (инструмент, который вызывается на странице, и с помощью цикла выводятся сохраненные в него данные на HTML-страницу). В конце возвращается View, который является представлением, и уже с помощью ViewBag данные выводятся на странице.

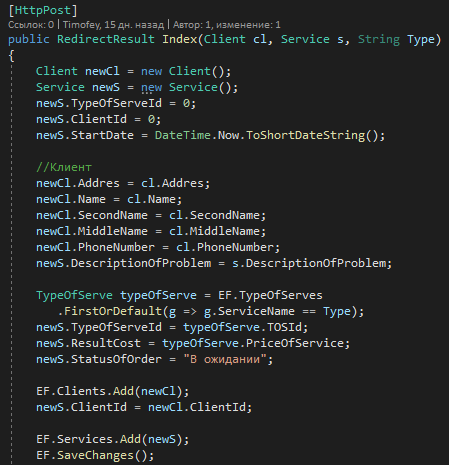


Рисунок 8 – Второй фрагмент кода

В коде на рисунке 8 написан алгоритм изменения данных БД. Здесь создаются экземпляры, в которые сохраняются временные данные (данные, которые принимаются по форме подачи заявления, присутствующей на сайте). Потом созданные экземпляры сохраняются, и в таблицу добавляются новые данные из этих экземпляров с помощью команды EF.Clients.Add(newCl) и EF.Services.Add(newS).

## **3.3. Логическая схема базы данных**

Разрабатываемая в данной работе логическая схема БД включает в себя 5 сущностей. Данная схема представлена на рисунке 9 и полностью отражает строение БД, используемой в приложении.

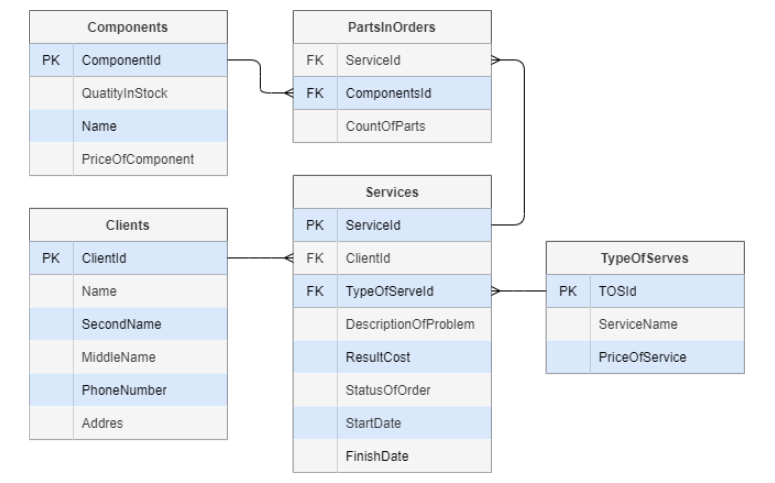


Рисунок 9 – Логическая схема БД

Сущность “Components” используется для хранения данных о комплектующих, которые может использовать сервис при осуществлении услуг типа “Замена комплектующих”. Содержит в себе 4 атрибута:

* ComponentId – первичный ключ;
* QuantityInStock – количество указанных комплектующих на складе;
* Name – название комплектующей;
* PriceOfComponent – цена, которая будет добавлена к стоимости услуги, если эта комплектующая используется в заказе.

Сущность “Clients” содержит в себе сведения о всех клиентах, которые подавали заявки и пользовались услугами данной организации. В ней присутствуют 6 атрибутов:

* ClientId – первичный ключ;
* Name – имя клиента;
* SecondName – фамилия клиента;
* MiddleName – отчество клиента;
* PhoneNumber – номер телефона клиента, используется для связи с клиентом согласования заказа;
* Addres – фактический адрес проживания клиента, используется если клиент вызывает мастера на дом.

Сущность “Services” используется для хранения данных о всех существующих заказах и содержит в себе самые важные детали. Здесь используются 8 атрибутов:

* ServiceId – первичный ключ;
* ClientId – вторичный ключ, содержит в себе уникальный номер клиента из сущности “Clients”;
* TypeOfServeId – вторичный ключ, содержит в себе уникальный номер услуги из сущности “TypeOfServes”;
* DescriptionOfProblem – краткое описание проблемы. Клиент может и не заполнять его, но это может облегчить понятие проблемы;
* ResultCost – полная стоимость, которую должен оплатить клиент при завершении заказа. В нее входит как стоимость работ, так и стоимость комплектующих;
* StatusOfOrder – текущий статус заказа;
* StartDate – дата подачи заявки, ставится автоматически;
* FinishDate – дата завершения заказа, ставится автоматически.

Сущность “PartsInOrders” используется для связи сущностей “Components” и “Services” и хранит в себе названия комплектующих из определенного заказа. В данной сущности всего 3 атрибута:

* ServiceId – вторичный ключ, содержащий в себе уникальный номер заказа из сущности “Services”;
* ComponentsId – вторичный ключ, уникальный номер комплектующей из сущности “Components”;
* CountOfParts – количество комплектующих с одинаковым уникальным номером, использующихся в данном заказе.

Последняя сущность в данной схеме – “TypeOfServes”. Содержит в себе сведения о всех оказываемых услугах и их ценах, то есть из нее формируется прайс лист на сайте. В ней хранится 3 атрибута:

* TOSId – первичный ключ, уникальный номер услуги и нумерация;
* ServiceName – название услуги;
* PriceOfService – номинальная стоимость услуги, без учета комплектующих.

## **3.4. Разработка интерфейса взаимодействия пользователя с системой**

Пользователей данного приложения можно поделить на 2 типа: клиент и сотрудник фирмы. Клиент использует сайт для подачи заявлений и просмотра прайс листа. Администратор (сотрудник) может просматривать заказы и изменять их статус.

**Часть I. Интерфейс взаимодействия клиента с приложением.**

****

Рисунок 10 – верхняя часть главной страницы

Здесь располагается название сервиса, а также контактный номер телефона. Поскольку главная страница получилась объёмной, для удобства пользователя были добавлены ссылки для мгновенного перехода к форме или к концу страницы, где располагается карта сервиса и ссылка на прайс лист.

Ниже располагаются баннеры и главная часть этой страницы – форма подачи заявки. В ней необходимо указать свои данные, выбрать тип услуги и кратко описать проблему, после чего нажать “Записаться”, и введенные данные занесутся в БД, после чего администратор увидит заявку на специальной странице. Форма проверяется на валидность, описание проблемы необязательно.

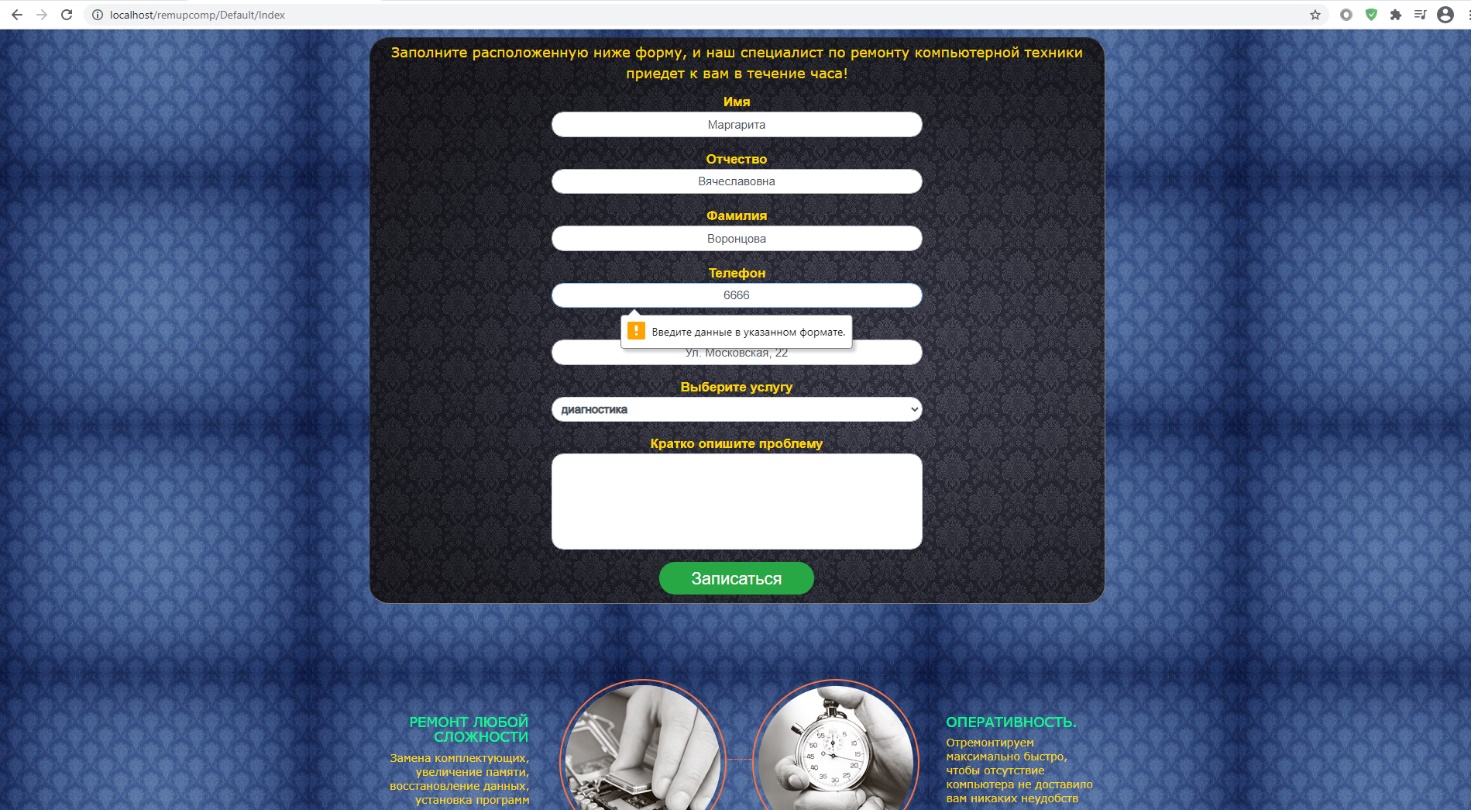


Рисунок 11 – Форма для записи

Если пользователь не хочет заполнять заявку и листает дальше вниз, он увидит отзывы о сервисе и схему работы, для поднятия доверия.

Далее, он может вернуться наверх и заполнить форму, либо переместиться в самый низ страницы (также сюда сразу можно попасть по ссылке в верхней части страницы). Здесь расположена Google-карта, на которой отмечено местоположение сервиса, и пользователь может сразу построить маршрут, в случае если захочет лично приехать в сервис или забрать заказ. Также здесь расположена ссылка на прайс-лист, где можно ознакомиться с ценами и списком оказываемых услуг.



Рисунок 12 – конец вёрстки страницы

При нажатии на кнопку “Прайс лист” пользователь попадет на отдельную страницу и там таблично отображены все оказываемые услуги и цены на них. Также присутствует кнопка возвращения на главную страницу.

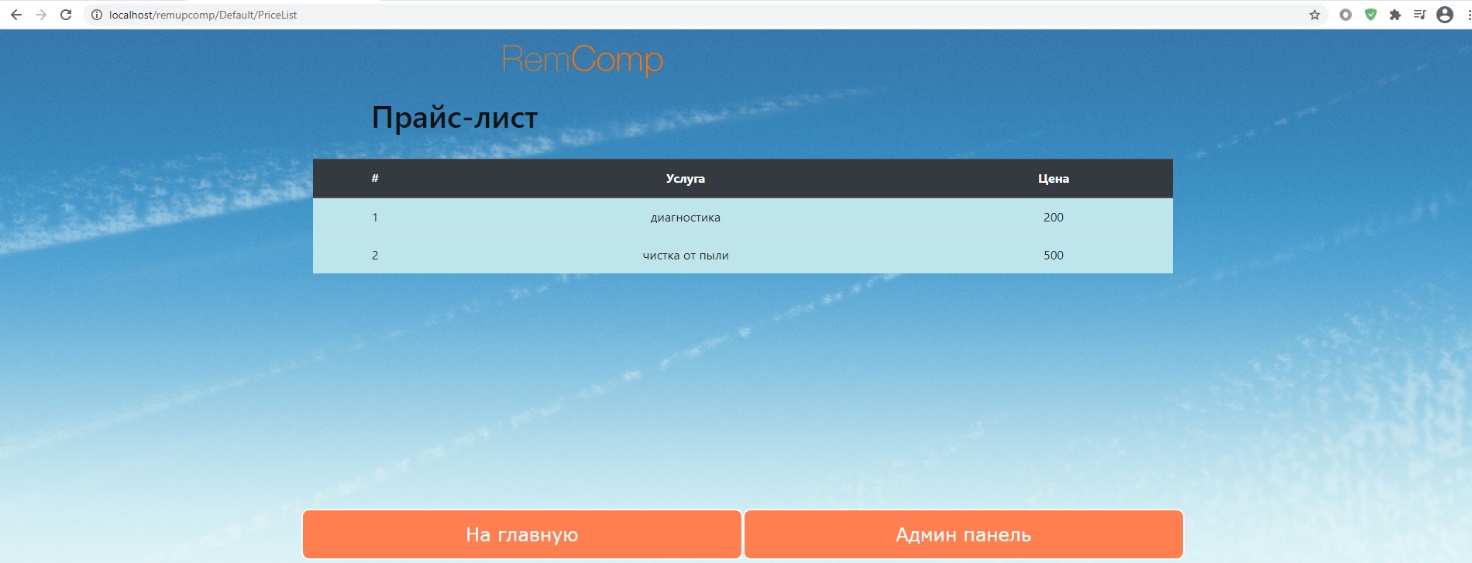


Рисунок 13 – Список услуг

**Часть II. Интерфейс взаимодействия админа с приложением.**

Главная цель использования администратором данного приложения – просмотр заказов и изменение их статуса на специальной странице, куда можно попасть авторизовавшись под именем администратора.

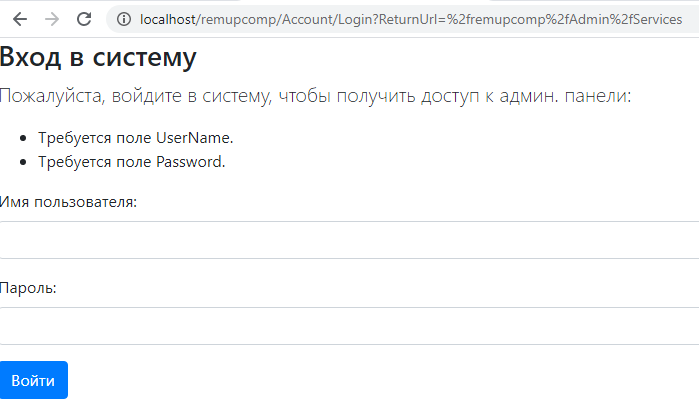


Рисунок 14 – Авторизация администратора

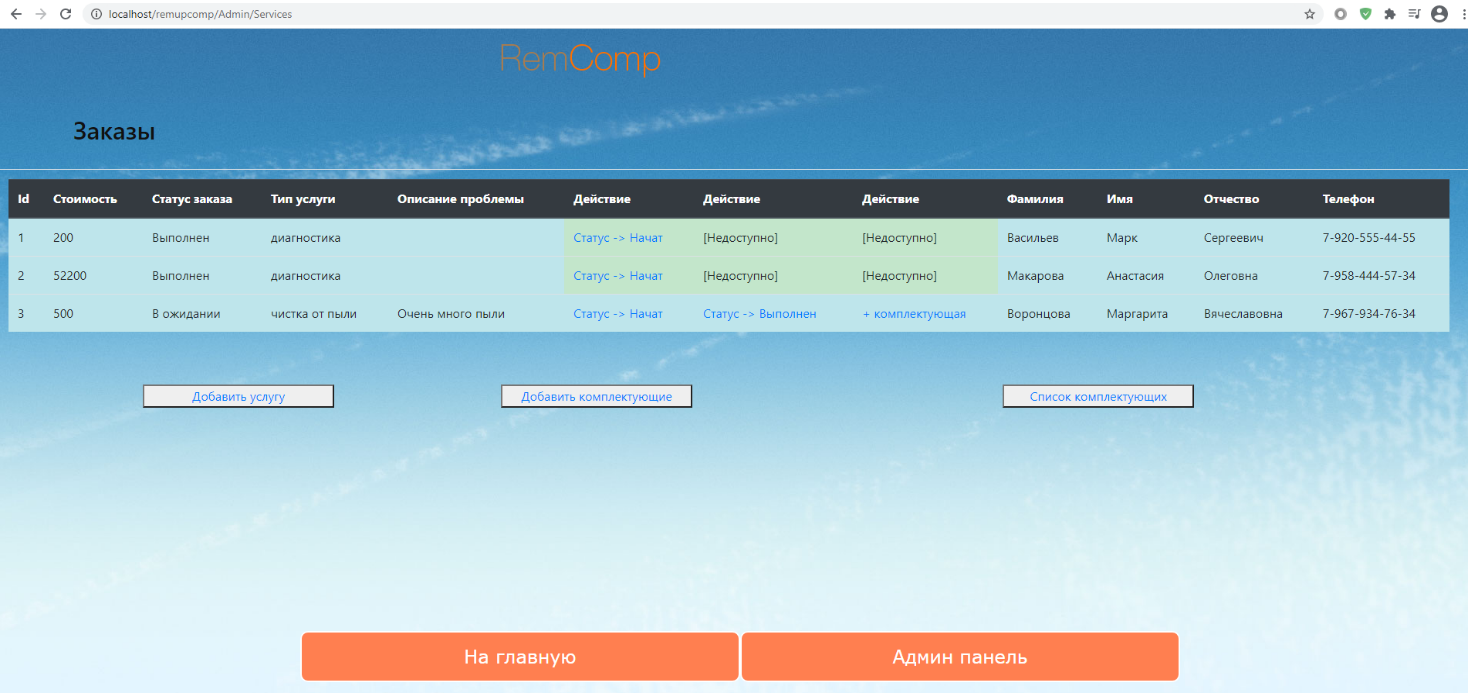


Рисунок 15 – Просмотр заказов

Здесь можно поменять статус заказа на “Начат” либо “Выполнен”, а также добавить комплектующие. Эти функции блокируются при завершении заказа.

Также с помощью сайта администратор может добавлять новые виды услуг и клиент сможет выбрать их при подаче заявки или увидеть в прайс-листе.

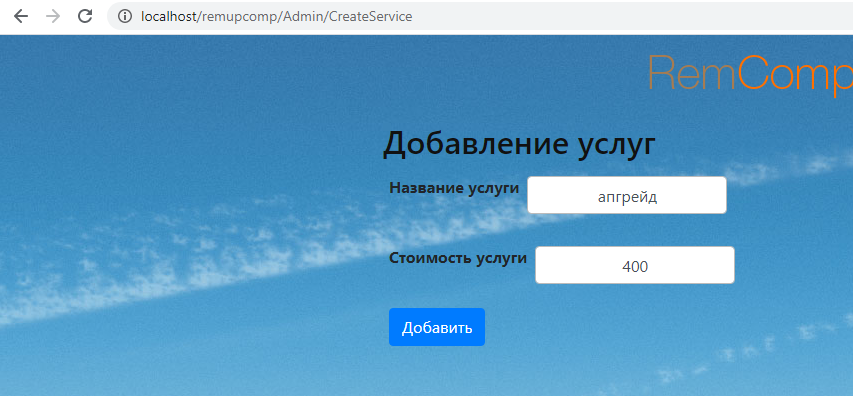


Рисунок 16 – Добавление услуг

Также администратор может на отдельной странице добавлять комплектующие, которые могут быть использованы при замене. Их список также ведётся сервисом на соответствующей странице.

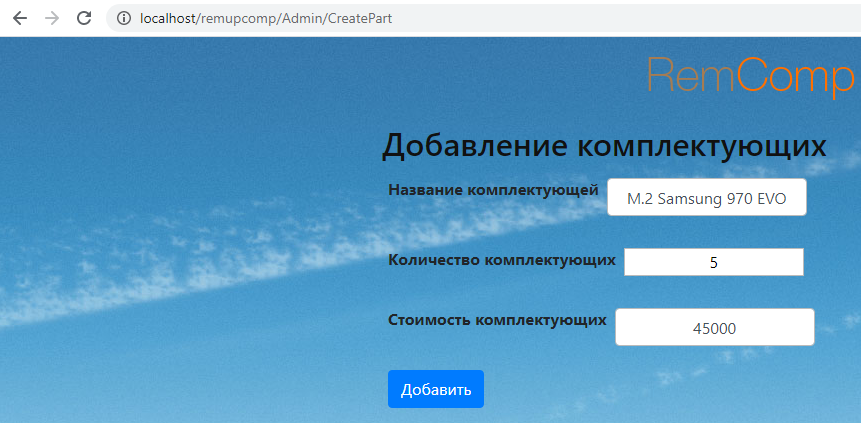


Рисунок 17 – Добавление комплектующих

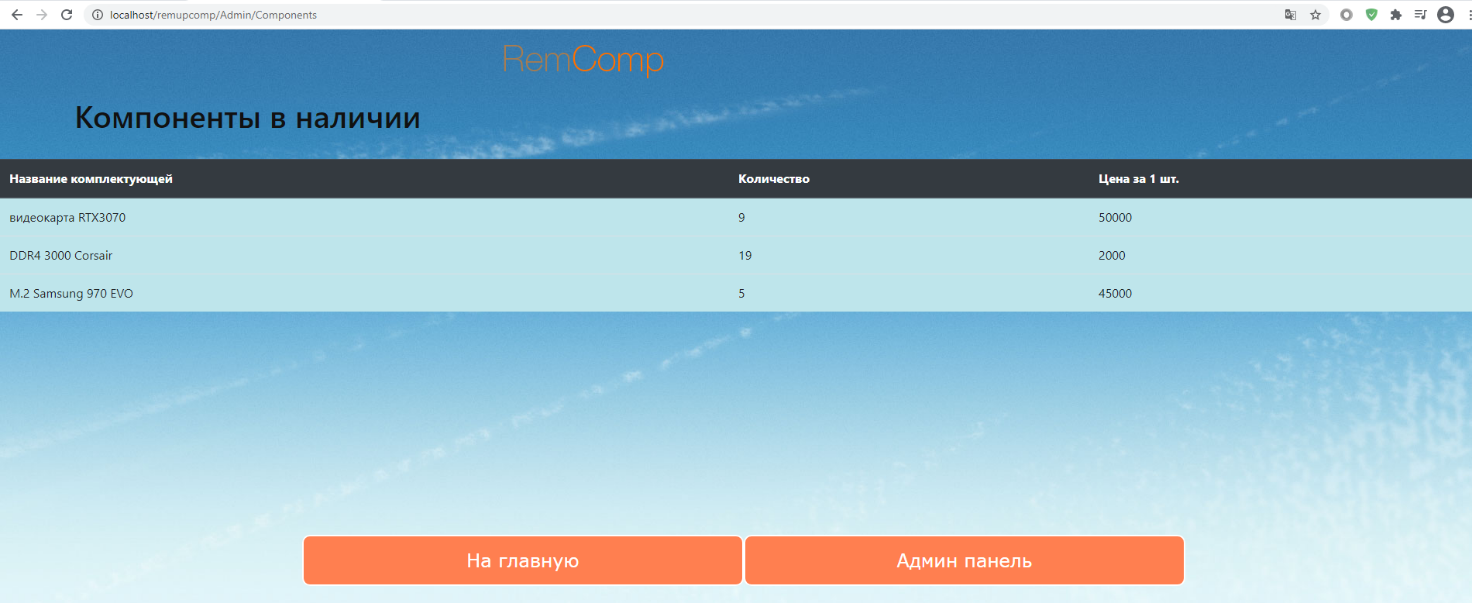


Рисунок 18 – Просмотр списка комплектующих

Как было сказано ранее, к заказам можно добавлять используемые комплектующие на странице просмотра заказов, что увеличит стоимость заказа. Особенно необходимая функция для услуг вида “апгрейд”.

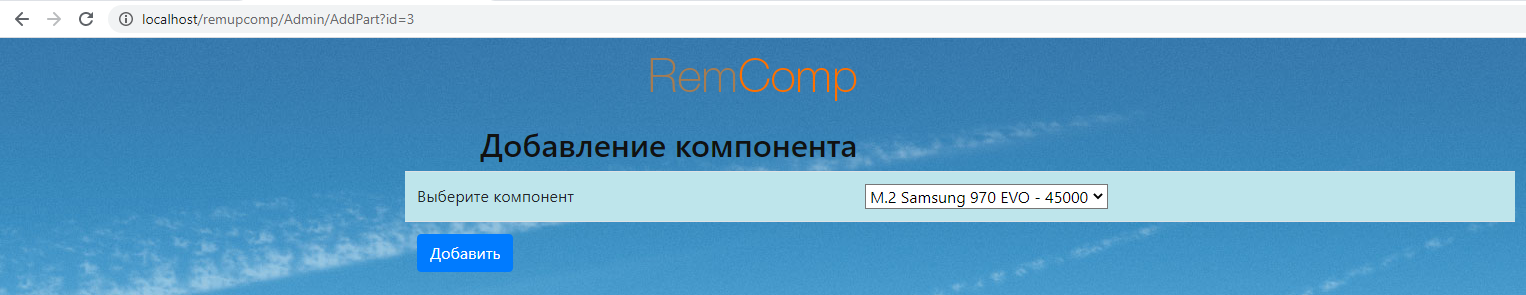


Рисунок 19 – добавление комплектующей к заказу

## **3.5. Разработка архитектуры приложения**

Разрабатываемое приложение имеет архитектуру клиент-сервер. Разработка ведется на платформе ASP.NET. Проект разделен на 2 решения для удобства: Domain - серверная часть и WebUi – клиентская часть.

Для взаимодействия приложения и сервера используется протокол HTTP по принципу REST, который был задуман как простой и однозначный интерфейс для управления данными.

REST предполагает несколько операций с сетевым хранилищем (сервером):

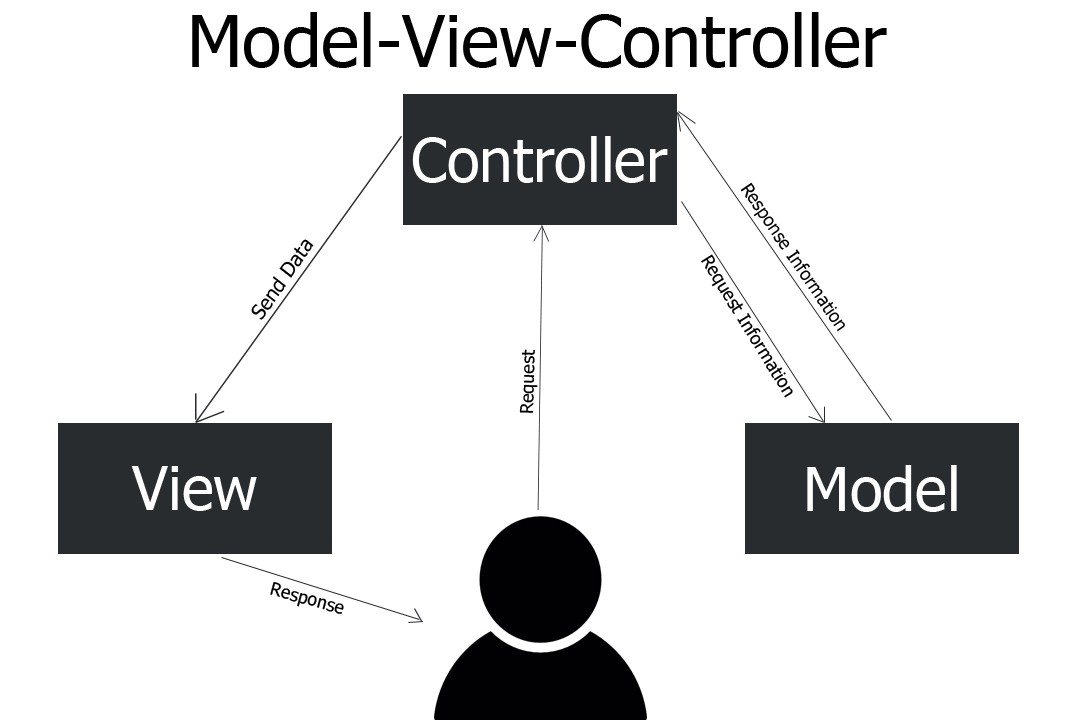
* GET — для получения (чтение)
* POST — для создания
* PUT — для изменения
* DELETE — для удаления

На сервере хранится База Данных и СУБД. В Asp.Net в качестве БД используется MS SQL, а в качестве СУБД - MS SQL Server. Также БД можно хранить и на отдельном сервере, если есть необходимость.

Благодаря используемому фреймворку Entity Framework в Asp.Net таблицы БД связаны с классами приложения, образуя тем самым ORM-модель. Это позволяет получать, вносить и изменять данные БД без применения языка SQL.

Asp.Net использует шаблон проектирования MVC (Model-View-Controller) – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на 3 компонента: модель, представление и контроллера, которые находятся отдельно друг от друга и могут изменяться независимо.

1. Модель — этот компонент отвечает за данные, а также определяет структуру приложения.
2. Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.
3. Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

  
Рисунок 20 – Схема MVC

Одной из важных деталей в веб-приложении является верстка страниц (раздел View), осуществляемая на языке Html с применением стилей Css и скриптов JavaScript. В приложении верстка имеет представление “cshtml”.

Для версток используется Bootstrap Framework, который предоставляет функции, улучшающие интерфейс и простые в использовании.

Важным звеном в архитектуре приложения является контроллер. Он обрабатывает данные запроса и передает результат обработки обратно. На языке C# он является обычным классом, который наследуется от абстрактного Microsoft.AspNetCore.Mvc.Controller.

Здесь разработано 3 контроллера.

* AdminController – предназначен для перехода на панель администратора после успешной авторизации. Обрабатывает и меняет данные.
* DefaultController – обрабатывает данные от обычных пользователей.
* AccountController – обрабатывает данные при авторизации.

Для передачи данных из контроллера в представление используется инструмент ViewBag.

Также главную роль играют классы C#, которые генерируют таблицы в БД. Подробная схема классов представлена в последующем пункте “Реализация функционирующего приложения”.

Следует отметить класс DbSet, описывающий набор сущностных классов, который затем можно использовать в коде для создания запросов CRUD (create, read, update, delete) к данным. С помощью экземпляров этого класса описываются различные объекты базы данных (таблицы, представления, хранимые процедуры и т.д.).

В Entity Framework создается модель EF, которая хранит сведения о сопоставлении классов и свойств приложений с таблицами и столбцами баз данных.

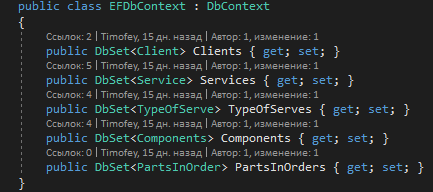


Рисунок 21 – создание коллекций сущностей через DbSet

## **3.6. Реализация функционирующего приложения**

Приложение было реализовано по всем требованиям ТЗ.

Ошибок при исполнении кода не проявляется, приложение работает стабильно.

Как было сказано раннее, для создания сущностей в БД используются классы C#. А также есть класс, в котором создаются коллекции сущностей БД. Сами сущности создаются при первом запуске приложения на основе классов.

Ниже представлена диаграмма основных классов.

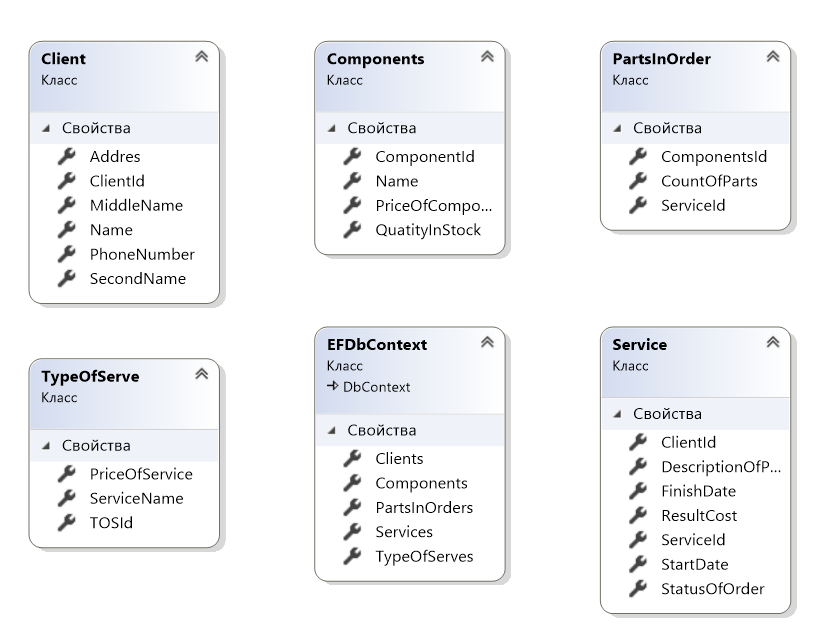


Рисунок 22 – диаграмма классов

На основе классов Client, Components, PartsInOrder, TypeOfServe, Service формируются сущности в БД, хранящие в себе соответственно данные о клиентах, комплектующих, комплектующих в заказах, а также виды услуг и сведения об оказанных услугах. Инкапсулированные свойства классов соответствуют атрибутам сущностей, представленных в логической схеме БД.

Класс EFDbContext унаследован от DbContext (базового класса Entity Framework, который предоставляет широкие возможности по работе с базой данных: создание запросов, отслеживание изменений и сохранение данных в базе.) и содержит в себе свойства DbSet, предоставляющие коллекции указанных сущностей в контексте, как показано на рисунке 20.

# 4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## **4.1. Порядок развертывания системы**

Исходную систему можно запустить как через Visual Studio 2019 (чтобы внести изменения), так и через IIS (рекомендуется версии 10 под Windows 10) при выполнении определенных операций. БД при этом может быть арендована и храниться на отдельном сервере либо храниться на локальном.

**Последовательность действий для запуска проекта через Visual Studio**

* Прежде всего, для стабильного и быстрого исполнения приложения рекомендуется использовать компьютер, имеющий следующие технические характеристики: ОЗУ – 4гб и выше, процессор – 4 потока с тактовой частотой 3 ГГц, свободное место – 500 мб., операционная система Win 7 или новее, Net Framework 4.8.
* Установить VS Studio 2019, поскольку разработка велась в нем, а также набор компонентов для разработки ASP.NET приложений, из отдельных компонентов – Net Framework 4.6.1 SDK (либо Targeting Pack если установлена более новая версия SDK), остальные необходимые компоненты выбираются по умолчанию.

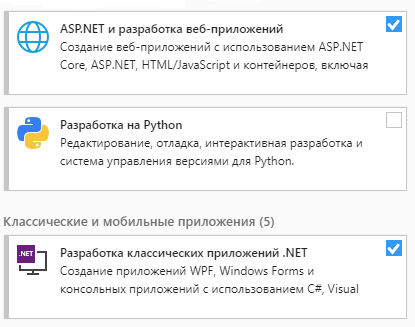


Рисунок 23 – компоненты установки

* Затем остается просто открыть данное решение в VS и запустить его. С этого момента сервер начинает работу и можно подключаться к сайту.

**Последовательность действий для развертывания в Internet Information Services на локальном сервере и с локальной СУБД.**

* Установить сами ISS, в Windows 10 они являются встроенными и отключены по-умолчанию. Включить можно через ‘Панель управления – Программы и Компоненты – Компоненты Windows’.



Рисунок 24 – необходимые компоненты Windows

* Затем нужно установить саму Субд. Для локального развертывания подойдет SQL Server Express LocalDB, либо можно скачать SQL Server Express 2014 (вместе с VS 2019 ставится автоматически 2013).
* После этого нужно зайти в диспетчер служб ISS. Файл проекта предварительно должен быть опубликован через VS Studio и находиться в папке ‘C:\inetpub\wwwroot’.

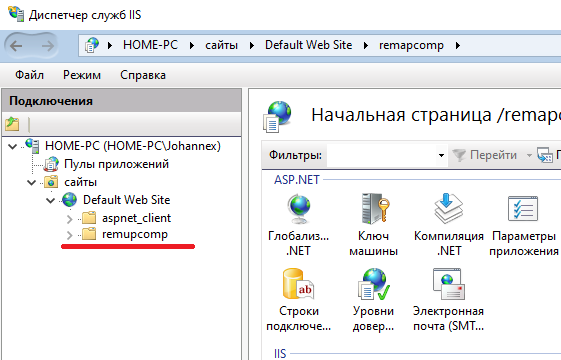


Рисунок 25 – папка приложения

* В диспетчере должна быть видна папка. Необходимо нажать ПКМ на нее и выбрать “Преобразовать в приложение”. Далее параметры можно оставить по умолчанию и подтвердить операцию, либо поменять по желанию псевдоним (имя сайта) и пул приложений.

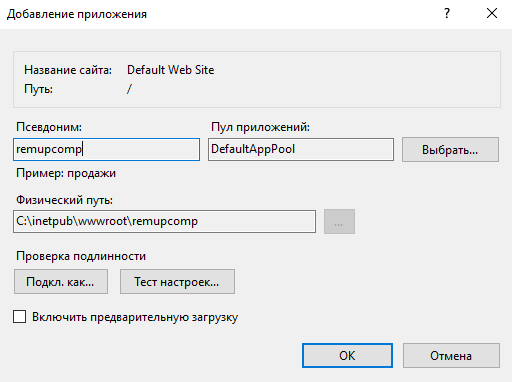


Рисунок 26 – Добавление приложения

* После этого обязательно нужно в настройке “Удостоверение пула приложений” выбрать “Встроенная учетная запись – LocalSystem”, чтобы приложение подключилось к СУБД.

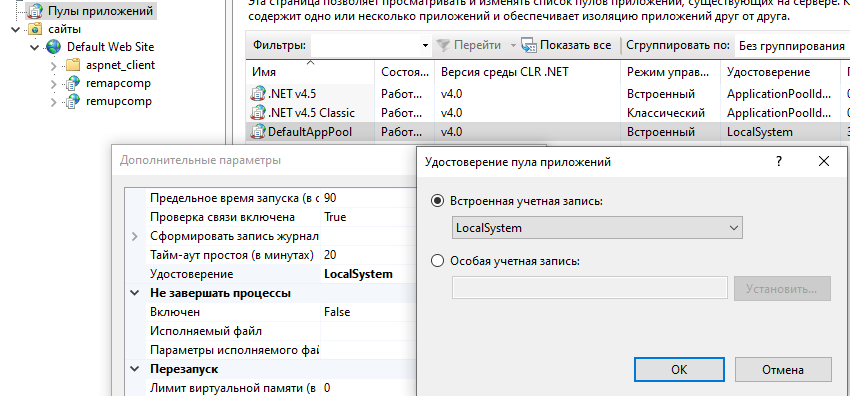


Рисунок 27 – настройка удостоверения

* Теперь можно подключаться к сайту по адресу ‘localhost**/**псевдоним**/**файл\_верстки’.

## **4.2. Уровень подготовки пользователя**

Для развертывания системы через ISS/VS2019 нужны определенные знания в области администрирования серверов ISS (если запуск без VS), установки компонентов Windows и программ, сборки и запуска проектов в VS.

Для пользования системой не нужно никаких определенных навыков или знаний, интерфейс был разработан максимально понятен для осуществления нужных операций любыми пользователями (как администратором, редактирующим заказы, так и клиентами, подающими заявления).

## **4.3. Руководство пользователя**

В данном пункте будут показаны подсказки по основной части главного сайта для клиентов сервиса.

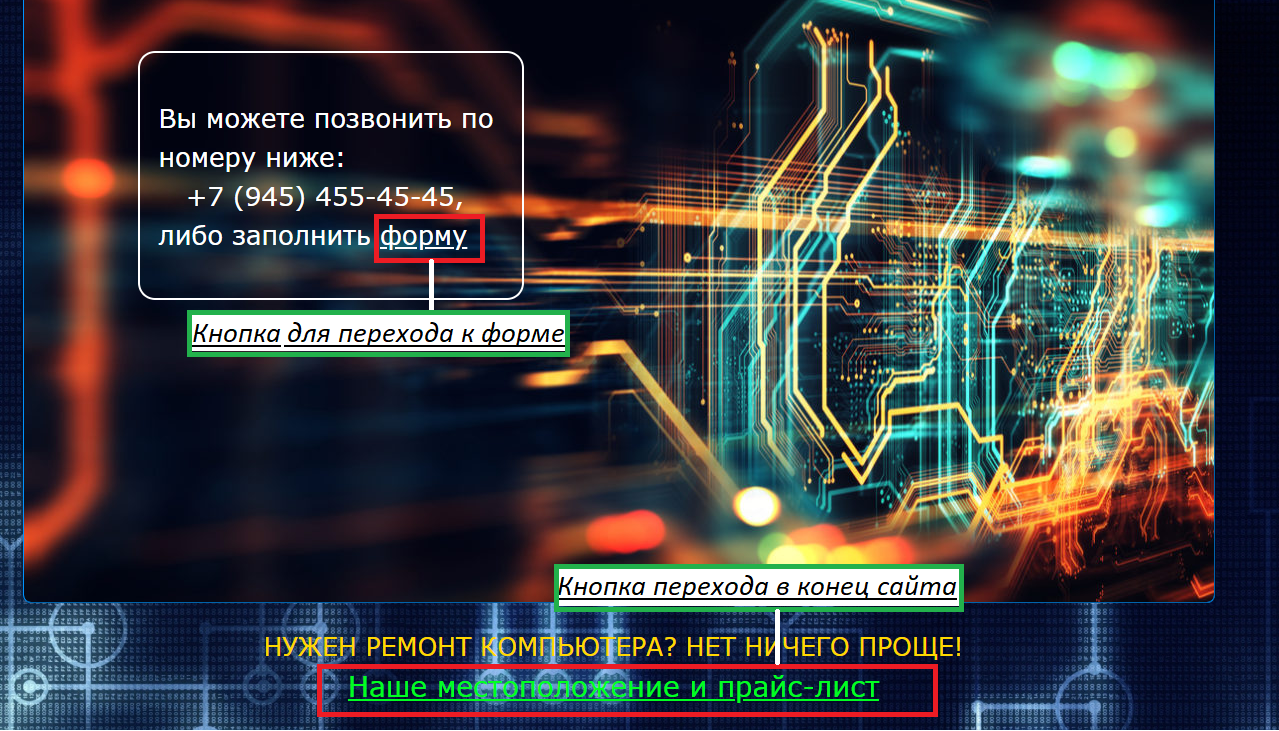


Рисунок 28 – руководство 1

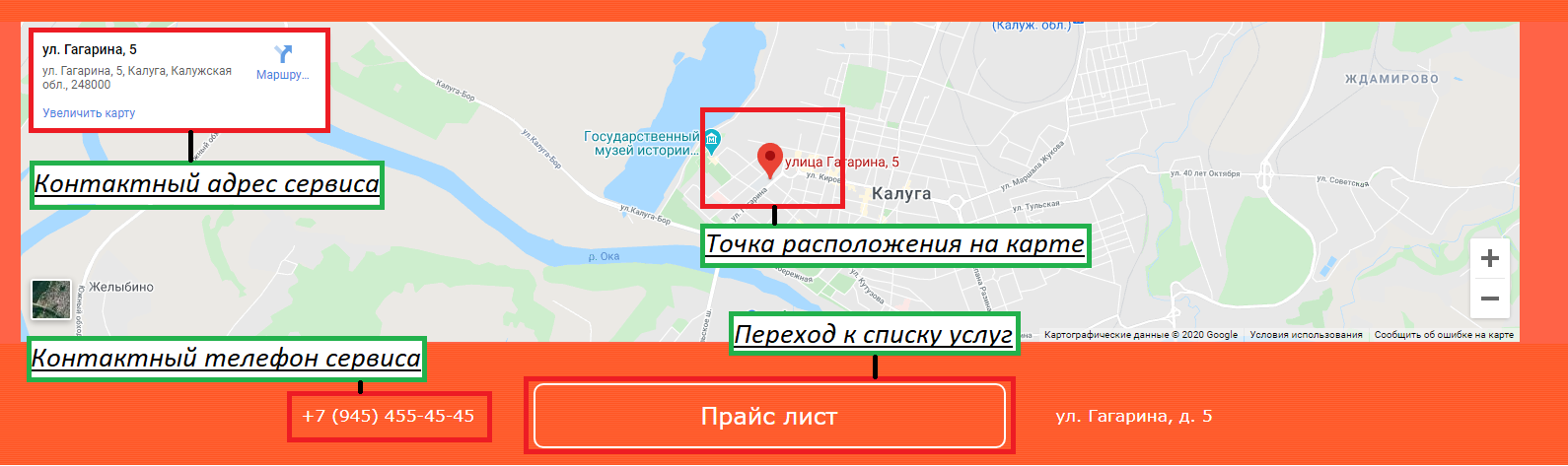


Рисунок 29 – руководство 2

Форма подачи заявления, прайс-лист пропущены в данном руководстве поскольку не нуждаются в комментировании ввиду своей понятности.

## **4.4. Возможности улучшения**

При необходимости в дальнейшем можно будет реализовать возможность пользователям самим просматривать статус своего заказа, а также получать чеки на завершенные заказы.

Также, при расширении возможностей сервиса до продаж БУ или новой техники, можно будет дополнить сайт страницами с товарами, а также создать механизм их покупки, личный кабинет пользователя для получения бонусов от покупки.

## **4.5. Тестирование приложения**

Для проверки стабильность приложения была проведено тестирование контроллеров, отвечающих за основные веб-страницы: главная страница, прайс лист, просмотр заказов и просмотр списка комплектующих.

Тесты были успешно пройдены, ошибок не возникло.

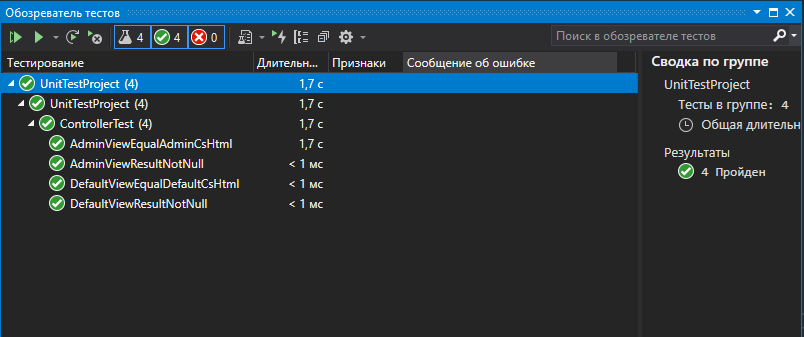


Рисунок 30 – результаты тестирования



Рисунок 31 – класс тестирования

# ЗАКЛЮЧЕНИЕ

При выполнении курсовой работы “Сервис по ремонту и обслуживанию АО” по дисциплине Базы Данных была выбрана и описана предметная область, выполнен анализ аналогов разрабатываемого сайта, а также составлена Научно-Исследовательская часть, для понимания сути проекта, Проектно-Конструкторская для понимания структуры и архитектуры, а также Проектно-Технологическая часть для обоснования способов взаимодействия с приложением.

Получены знания и навыки в написании Net приложений, применении ORM модели при работе с БД, а также отработаны навыки по верстке веб-страниц.

Также было разработано и протестировано приложение, соответствующее всем описанным в ТЗ требованиям.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Адам Фримен. Entity Framework Core 2 для ASP.NET Core MVC для профессионалов
2. Джесс Чедвик. ASP.NET MVC 4. Разработка реальных веб-приложений с помощью ASP.NET MVC
3. Джеффри Рихтер. Applied Microsoft.NET Framework Programming
4. Джеффри Рихтер. CLR VIA C#
5. Джон Скит. C#. Программирование для профессионалов
6. Эндрю Троелсен. Язык программирования C# 7 и платформы .NET и .NET Core
7. Создание сайта с нуля на ASP.NET Core MVC [Электронный ресурс] – URL://<https://alekseev74.ru/lessons/show/aspnet-core-mvc/step-by-step>
8. Начало работы с ASP.NET MVC 5 [Электронный ресурс] – URL://<https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/introduction/getting-started>
9. ASP.NET MVC. Урок 0. Вступление [Электронный ресурс] – URL://<https://habr.com/ru/post/175999/>
10. Изучаем ASP.NET MVC 5 [Электронный ресурс] – URL://<https://professorweb.ru/my/ASP_NET/mvc/level1/>