

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств ВТ и
автоматизированных систем»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Клиент–серверное приложение “Планировщик задач”

Студент

Ир. 03.06.20 Мультяева Ирина Игоревна
(подпись, дата)

Руководитель

Н.С. 03.06.20 Карамышева Н.С.
(подпись, дата)

Нормоконтролёр

А.В. 03.06.20 А.В. Кучин
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой

М.А. 03.06.20 М.А. Митрохин
(подпись)

Работа защищена с отметкой

хорошо (протокол заседания ГЭК от 22.06.2020 № 12)

Секретарь ГЭК

Е.И. 22.06.20 Е.И. Гурин
(подпись)

Пенза 2020

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«30» апреля 2020 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Мультяева Ирина Игоревна гр. 16вв2 факультета вычислительной техники.
2. Тема работы «Клиент–серверное приложение “Планировщик задач”».

Тема утверждена приказом ПГУ № от "30" апреля 2020 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.
4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение разработки – клиент–серверное приложение, предназначенное для планирования выполнения задач на удаленной ЭВМ.
5. Перечень подлежащих разработке вопросов.
Требуется произвести анализ предметной области, определить архитектуру приложения, выбрать языки программирования для клиентского и серверного приложения, выбрать способ взаимодействия между клиентом и сервером, разработать протокол взаимодействия.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.20	
2.	Выбор архитектуры приложения, языков программирования, способа взаимодействия	10	04.05.20	
3.	Разработка протокола взаимодействия	10	10.05.20	
4.	Разработка серверной части приложения	20	15.05.20	
5.	Разработка клиентской части приложения	20	20.05.20	
6.	Тестирование и отладка	10	23.05.20	
7.	Оформления выпускной работы	20	31.05.20	

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы  30.04.20 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент  - 30.04.20 Мультяева И.И.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 135 листов, 49 рисунков, 6 приложений.

КЛИЕНТ–СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, ПРОТОКОЛ
ВЗАИМОДЕЙСТВИЯ ТСР/IP, ПЛАНИРОВЩИК ЗАДАЧ, КЛИЕНТ,
СЕРВЕР, ЗАДАЧА.

В рамках данной выпускной квалификационной работы было создано клиент–серверное приложение шифрования данных.

Цель работы – разработать клиент–серверное приложение, планирующее и запускающее задачи на удаленной ЭВМ.

Объектом разработки является клиент–серверное приложение по планированию и запуску задач, состоящее из клиентской программы с графическим интерфейсом и сервера, который представляет собой службу, работающую в фоновом режиме.

В результате проведенной работы реализован проект, удовлетворяющий поставленной задаче, а именно, создано клиент–серверное приложение, осуществляющее планирование и запуск задач на удаленной ЭВМ.

Разработка проводилась в среде программирования Visual Studio 2010.
Использованные языки программирования: C#.

Созданное приложение может быть внедрено на предприятиях, где необходимо администрирование большого числа рабочих станций. Оно может быть использовано в тех случаях, где требуется запуск программ и утилит в конкретное время, автоматизированный запуск повторяющихся задач.

					ПГУ 2.090301.26.001 ПЗ								
Изм.	Лист	№ докум.	Подпись	Дата	Клиент-серверное приложение “Планировщик задач” Пояснительная записка				Лит.	Лист	Листов		
Разраб.		Мультияева И.И.										2	135
Пров.		Карамышева Н.С.											
Н. Контр.		Кучин А.В.											
Утв.		Митрохин М.А.											
					ФВТ, каф. ВТ гр. 16ВВ2								

Содержание

РЕФЕРАТ	2
Введение.....	6
1 Обзор предметной области.....	7
1.1 Постановка задачи.....	7
1.2 Инструменты для реализации клиента	8
1.2.1 Анализ требований.....	8
1.2.2 Язык программирования для реализации клиента	9
1.2.3 Windows Forms API для создания графического интерфейса	10
1.3 Инструменты для реализации сервера.....	11
1.3.1 Анализ требований.....	11
1.3.2 Язык программирования для реализации сервера.....	12
1.3.3 Windows служба как способ реализации сервера	13
1.4 Механизм взаимодействия клиента и сервера	14
2 Реализация клиент–серверного приложения.....	16
2.1 Описание работы клиент–серверного приложения.....	16
2.1.1 Описание работы клиента	16
2.1.2 Описание работы сервера.....	17
2.2 Разработка клиентской части приложения.....	19
2.3 Разработка серверной части приложения.....	22
3 Описание приложения	27
3.1 Руководство пользователя.....	27
3.2 Спецификация клиента.....	29
3.3 Спецификация сервера	44
3.4 Библиотека Data	45
Заключение	48
Список используемых источников.....	49
Приложение А Листинги программы клиента.....	50
Приложение А.1 Файл «Program.cs»	50

Приложение А.2 Файл «MainForm.cs».....	50
Приложение А.3 Файл «TaskForm.cs»	54
Приложение А.4 Файл «TriggerForm.cs»	58
Приложение А.5 Файл «ActionForm.cs»	75
Приложение А.6 Файл «ServerForm.cs».....	77
Приложение А.7 Файл «PasswordForm.cs».....	78
Приложение А.8 Файл «General.cs»	80
Приложение А.9 Файл «Triggers.cs»	82
Приложение А.10 Файл «Actions.cs»	86
Приложение А.11 Файл «Safety.cs».....	90
Приложение А.12 Файл «ClientSocket.cs»	93
Приложение В Листинги программы сервера.....	96
Приложение В.1 Файл «Program.cs»	96
Приложение В.2 Файл «TaskManager.cs»	96
Приложение В.3 Файл «Client.cs»	98
Приложение В.4 Файл «TaskHandler.cs»	99
Приложение С Листинг библиотеки Data.....	112
Приложение С.1 Файл «Data.cs».....	112
Приложение D Результаты работы программы	114
Приложение D.1 – Вид начального экрана приложения.....	114
Приложение D.2 – Окно создания задачи. Вкладка «Общие».	114
Приложение D.3 – Окно создания задачи. Вкладка «Триггеры»	115
Приложение D.4 – Окно создания триггера.	115
Приложение D.5 – Окно создания задачи. Вкладка «Действия»	116
Приложение D.6 – Окно создания действия.....	116
Приложение D.7 – Окно создания задачи. Вкладка «Безопасность»	117
Приложение D.8 – Окно изменения задачи.....	117
Приложение E UML–диаграммы приложения.....	118
Приложение E.1 – UML–диаграмма классов клиента.....	118
Приложение E.2 – UML–диаграмма классов сервера.	119
Приложение E.3 – UML–диаграмма вариантов использования клиента.	119

Приложение Е.4 – UML–диаграмма вариантов использования сервера.....	120
Приложение Е.5 – UML–диаграмма последовательности.....	120
Приложение Е.6 – UML–диаграмма деятельности клиента.	121
Приложение Е.7 – UML–диаграмма деятельности сервера.....	122
Приложение Е.8 – UML–диаграмма развертывания	123
Приложение F Презентация	124

Введение

В настоящее время локальные вычислительные сети являются необходимым условием для эффективной работы большинства предприятий. Они объединяют в единую систему десятки и сотни автономных персональных компьютеров, расположенных на удаленных рабочих местах. К несомненным достоинствам локальных вычислительных сетей относятся: совместное использование файлов, программ и оборудования; упрощение организации и координирования совместной работы; настройка общего доступа к подключению Интернета и ряд других возможностей.

Даже имея множество преимуществ, использование локальной вычислительной сети требует постоянного квалифицированного обслуживания специалистом – системным администратором. В его обязанности входит поддержание бесперебойной работы компьютерной техники, установка и деинсталляция программного обеспечения, резервное копирование данных и их восстановление в случае необходимости. Выполнение этих задач вручную – трудоемкий и затратный по времени процесс, поэтому администратору желательно обеспечить автоматизацию работы для постоянно повторяющихся действий. Для таких целей существуют специальные программы, занимающиеся планированием и автоматическим запуском процессов – планировщики задач.

В ходе данной работы необходимо разработать клиент–серверное приложение для планирования выполнения задач на локальных и удаленных ЭВМ. Серверная часть должна обеспечивать хранение и выполнение задач, их отправку клиенту для последующего редактирования. Клиентская часть должна предоставить функционал для работы с задачами, а именно для их создания, изменения или удаления. Также клиентская программа должна иметь графический интерфейс для визуализации списка работающих задач.

					ПГУ 2.090301.26.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Обзор предметной области

1.1 Постановка задачи

Необходимо разработать клиент–серверное приложение, позволяющее создавать, изменять, удалять и выполнять по расписанию задачи на локальных и удаленных ЭВМ.

Приложение должно удовлетворять следующим требованиям:

1. Планировщик задач должен иметь функции добавления, изменения, удаление и получения списка задач.
2. У каждой задачи должен быть свой список действий, которые она выполняет.

Действие – это запуск программы или открытие файла. Задача должна выполнять как минимум одно действие.

3. У каждой задачи должен быть свой список триггеров, которые запускают её выполнение при наступлении заданного времени.

Триггер – это событие, при наступлении которого запускается выполнение задачи.

Необходимо реализовать следующие типы триггеров:

1. Однократный триггер – срабатывает один раз при наступлении указанной пользователем даты и времени.
2. Ежедневный триггер – срабатывает один раз в день при наступлении определенного времени, через указанное количество дней.
3. Еженедельный триггер – срабатывает в указанные пользователем дни недели при наступлении установленного времени, через определенное количество недель.
4. Ежемесячный триггер – срабатывает либо в определенные дни выбранных пользователем месяцев, либо в указанные дни недель выбранных месяцев.

4. Каждая задача должна иметь информацию о пользователе, в контексте которого будет осуществляться выполнение задачи.
5. Клиентская часть должна показывать список запущенных задач.
6. Серверная часть должна давать возможность подключения клиенту для редактирования её задач.
7. Приложение должно работать в операционной системе Microsoft Windows, минимальная версия ОС – Microsoft Windows 7.

1.2 Инструменты для реализации клиента

1.2.1 Анализ требований

Взаимодействие пользователя с приложением происходит непосредственно через клиентскую часть, поэтому интерфейс клиентской части должен обеспечить пользователю положительный опыт взаимодействия с ним. Интерфейс пользователя – это интерфейс, с помощью которого пользователь управляет программным обеспечением. Правильно спроектированный интерфейс должен быть минималистичным, удобным в использовании, а взаимодействие с ним должно происходить на интуитивно понятном уровне.

Важным аспектом при проектировании приложения является выбор языка программирования. Он должен быть объектно–ориентированным, то есть, построенным на принципах объектно–ориентированного программирования и использующим его основные понятия: объекты, классы и методы. Также необходимо, чтобы этот язык имел возможность для реализации событийно–ориентированного программирования – такой парадигмы программирования, в которой работа программы основывается на обработке поступающих событий: действий пользователя, сообщений программ и потоков, событий операционной системы.

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

Для того чтобы наибольшим образом упростить создание приложения, интегрированная среда разработки должна включать в себя конструктор графического пользовательского интерфейса. Он позволит с легкостью создать интерфейс программы в готовом виде, при этом автоматически генерируя его исходный код.

1.2.2 Язык программирования для реализации клиента

На основании требований, описанных выше, в качестве языка программирования для реализации клиентского приложения был выбран C#. Язык C# – это высокоуровневый, мультипарадигмальный язык программирования, поддерживающий такие парадигмы программирования, как объектно–ориентированная и событийно–ориентированная парадигмы. C# имеет широкий набор инструментов и функций для разработки, а также некоторые особенности, например:

- строгая типизация, которая запрещает обращение к неинициализированным переменным, индексацию массивов вне их границ, бесконтрольное приведение типов;
- единая система типов, к которой можно применять единый набор операций для обработки и хранения значений типизации;
- использование ссылочных типов данных, что позволяет динамически выделять память под объект;
- обработка исключений, позволяющая с легкостью выявить и отладить ошибки и исключения;
- сборка мусора, в автоматическом режиме очищающая память от неиспользованных или уничтоженных объектов.

C# может работать с основными понятиями объектно–ориентированного программирования:

- Объектами – сущностями, которые состоят из данных (полей) и

поведения (методов);

- Классами – пользовательскими типами данных, с возможностью определить начальные значения состояний: инициализировать поля класса и реализовать поведение функций или методов;
- Методами – функциями, принадлежащими классу или объекту.

Может реализовать четыре основных принципа объектно-ориентированного программирования:

- Абстрагирование – выделение и представление главных, наиболее значимых характеристик;
- Инкапсуляция – ограничение доступа к данным и возможности их изменения;
- Наследование – создание нового класса на основе существующего и получение доступа к его данным и методам;
- Полиморфизм – использование одного и того же интерфейса для различных действий.

Также C# может работать с основным понятием событийно-ориентированного программирования – событием, или же с сообщением, возникающем в исполняемом коде при наступлении определенных условий.

1.2.3 Windows Forms API для создания графического интерфейса

Для реализации графического интерфейса клиентской программы были выбраны Windows Forms. Windows Forms – это набор управляемых библиотек графического пользовательского интерфейса, включенных в платформу .NET Framework. По сути, Windows Forms является объектно-ориентированной оболочкой вокруг интерфейса прикладного программирования Win32 API, что упрощает доступ к элементам интерфейса Microsoft Windows. Приложения Windows Forms выполняются в управляемой среде .NET Common Language Runtime (CLR).

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

10

Создание Windows Forms приложений имеет немало преимуществ. Windows Forms создают более однородную структуру программной модели, устраняют ошибки и противоречия, вызванные работой с Windows API. Библиотека классов .NET Framework содержит больше возможностей по реализации графического интерфейса, чем Windows API, а приложения, использующие Windows Forms, содержат меньше кода в сравнении с приложениями, использующими Windows API или библиотеку MFC. Кроме того, Windows Forms использует один и тот же API из библиотеки классов .NET Framework, вне зависимости от выбранного языка программирования, что позволит разработчику создавать приложения фактически на любом языке, поддерживающем .NET Framework.

1.3 Инструменты для реализации сервера

1.3.1 Анализ требований

Сервер – это программное или аппаратное устройство, которое принимает и обрабатывает запросы клиента, в ответ на них выполняет требуемые функции или предоставляет доступ к определенным данным или ресурсам. В сетевой архитектуре «клиент – сервер» и клиент, и сервер являются программным обеспечением. Как правило, эти программы располагаются на разных устройствах и взаимодействуют друг с другом с помощью сетевых протоколов, но они так же могут располагаться и на одном компьютере.

К необходимым функциям разрабатываемого серверного приложения относится: подключение клиентов, прием и обработка входящих запросов, загрузка массива задач из файла, их хранение, запись массива задач в файл, расчет времени активации триггеров задачи, запуск задач, отправка массива задач клиенту.

Как и для клиентской программы, довольно значимым моментом

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

является выбор языка программирования. Он должен иметь возможность реализовать весь требуемый функционал сервера, то есть, должен поддерживать:

- многопоточность, для параллельной работы функции подключения клиентов, обработки клиентских запросов и выполнения задач;
- межпроцессное взаимодействие удаленных устройств, для организации связи клиента и сервера;
- передачу объектов сложного типа по сети, для отправки массивов и одиночных задач.

1.3.2 Язык программирования для реализации сервера

Так же, как и для клиентской программы, в качестве языка программирования был выбран язык C#. Он содержит все инструменты для реализации указанных выше требований к серверному приложению.

Многопоточность – это свойство программы или платформы, заключающееся в том, что созданный в операционной системе процесс может состоять из нескольких потоков, выполняющихся параллельно. Стандартно каждое приложение имеет один основной поток в методе Main. Использование многопоточности ускоряет работу программы, так как сервер не должен дожидаться завершения одной функции, чтобы начать выполнение следующей. Язык C# имеет встроенную поддержку многопоточности, инструменты для работы с потоками находятся в пространстве имен System.Threading.

C# предоставляет возможность обмениваться данными между клиентом и сервером через функционал TCP и UDP сокетов. Подробное взаимодействие клиентского и серверного приложений описано в разделе 1.4.

Язык C# также предусматривает возможность передачи по сети

					ПГУ 2.090301.26.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

объектов сложного типа, например, структур, классов, перечислений. Для этой цели применяется механизм сериализации.

Сериализация – это процесс преобразования объекта в поток байтов для последующей передачи по сети или сохранения в файл. При необходимости объект можно вернуть к исходному формату, этот процесс называется десериализацией. Для возможности применения сериализации и десериализации к объекту добавляется атрибут `Serializable`, если для какого-либо поля или метода необходимо отключить сериализацию, используется атрибут `NonSerialized`. В разрабатываемом приложении используется бинарная сериализация, функции для работы с ней расположены в пространстве имен `System.Runtime.Serialization.Formatters.Binary`.

1.3.3 Windows служба как способ реализации сервера

Службы (сервисы) Windows – это долговременные специальные программы, автоматически запускаемые вместе со стартом операционной системы. Служба не имеет графического интерфейса, а ее работа протекает в фоновом режиме и скрыта от пользователя. Хотя служба начинает работу при загрузке операционной системы, она также может быть остановлена или перезапущена в любой другой момент. Эти особенности делают ее идеальным способом реализации серверного приложения.

Для запуска и управления службой необходимы три компонента:

- Диспетчер управления службами (SCM – service control manager). Именно он является основным средством для администрирования служб в операционной системе Windows. С его помощью можно создавать, запускать, останавливать и удалять службы.
- Программа управления службой (SCP – service control program), работает вместе с диспетчером управления службами.
- Приложение службы. Код приложения должен быть написан таким

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

13

образом, чтобы оно имело возможность принимать сообщения от диспетчера управления службами.

Служба может быть запущена из диспетчера управления службами, или из кода программы. Во втором случае, диспетчер управления службами вызовет ее, запустив метод OnStart. Если больше не посылать команд службе, то она будет находиться в запущенном состоянии до завершения работы операционной системы. У службы есть три основных состояния, в которых она может существовать: “Работает”, “Приостановлена” или “Остановлена”. Две последние команды могут быть выполнены аналогично команде запуска службы. В данном приложении реализованы две команды: OnStart – команда запуска и OnStop – команда остановки.

1.4 Механизм взаимодействия клиента и сервера

Реализация механизма взаимодействия между клиентским и серверным приложением будет выполнена при помощи протокола TCP/IP.

TCP/IP – это сетевая модель передачи данных в цифровом формате. Она описывает способ передачи данных от источника к приемнику. Модель TCP/IP предполагает прохождение информации через четыре уровня, каждый из которых описывается определенным правилом, или же протоколом передачи. Свое название модель получила от двух основных протоколов передачи данных в сети – Transmission Control Protocol (TCP) и Internet Protocol (IP).

В сравнении с аналогичным протоколом UDP, протокол TCP имеет множество достоинств, что делает его наилучшим решением для обеспечения взаимодействия между клиентом и сервером. TCP является надежным протоколом, так как после отправки данных отправитель спрашивает получателя о целостности полученных данных, и, при необходимости, пересылает их повторно. Также протокол TCP осуществляет управление

потоком информации, контролирует перегрузку сети и упорядоченно отправляет пакеты данных.

Протоколы TCP и UDP используют сокет – программные интерфейсы, обеспечивающие обмен данными между процессами как на одном, так и на разных устройствах. В языке C# работа с протоколом TCP происходит при помощи трех классов из пространства имен System.Net.Sockets:

- Класс Socket – обеспечивает базовые функции TCP и UDP сокетов.
- Класс TcpListener – класс серверного сокета, занимается прослушиванием входящих подключений от TCP-клиентов.
- Класс TcpClient – класс клиентского сокета, предоставляет методы для подключения, отправки и получения потоковых данных на стороне клиента.

При работе с протоколом TCP, клиентское приложение использует только клиентский сокет, а серверное – и серверные, и клиентские сокеты.

2 Реализация клиент–серверного приложения

2.1 Описание работы клиент–серверного приложения

Клиент–серверное приложение включает в себя клиентскую и серверную часть. Клиентская часть – программа, имеющая графический интерфейс и поддерживающая связь с сервером – осуществляет взаимодействие с пользователем. Серверная часть представляет собой программу, работающую в фоновом режиме, планируя и выполняя задачи. Клиент и сервер обмениваются данными между собой, используя протокол передачи данных TCP/IP.

2.1.1 Описание работы клиента

Работа клиента начинается с загрузки окна клиентского приложения. Пользователю необходимо настроить подключение к серверу, введя IP–адрес сервера и порт подключения. Если при попытке подключения к серверу возникнет ошибка, то пользователь получит сообщение о неудачном подключении. После подключения пользователь может обновить список работающих задач, создать новую задачу, изменить или удалить имеющуюся.

При добавлении новой задачи пользователю необходимо ввести имя задачи и создать для задачи как минимум одно действие. Для изменения или удаления задачи пользователю необходимо выбрать задачу из списка, затем нажать на кнопку с соответствующим действием. Если выбранная задача защищена паролем, то для выполнения этих действий пользователю необходимо ввести пароль задачи. Изменение задачи происходит аналогично созданию, но при этом вся имеющаяся информация о задаче заносится в поля ввода, отображаются списки созданных триггеров и действий. Для обновления списка задач необходимо выбрать соответствующее действие из окна клиентского приложения. Если при обновлении списка, добавлении, изменении или удалении задачи возникла ошибка, пользователь получит

					ПГУ 2.090301.26.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

сообщение об этом.

2.1.2 Описание работы сервера

Работа сервера начинается с загрузки задач из файла и их добавления в список запущенных задач. Далее создаются две нити, одна из которых принимает входящие запросы и передает запрошенную информацию клиенту, другая – управляет выполнением задач, поочередно проверяя указанные в триггерах события и запуская действия задачи при их наступлении. Если от клиента поступил запрос на добавление, изменение или удаление задачи, то, во избежание коллизий при попытке одновременного изменения общего ресурса, сервер производит блокировку нити запуска задач. После выполнения действий, запрошенных клиентом, сервер снимает блокировку с нити.

Сервер выполняет следующие действия:

1. Загрузка задач. При помощи десериализации происходит последовательное считывание сохраненных задач из файла и их добавление в список действующих задач.
2. Добавление задачи. Сервер получает задачу от клиента, добавляет ее в список, после чего перезаписывает в файл обновленный список задач. В случае возникновения ошибки сервер отправляет сообщение об этом клиенту.
3. Удаление задачи. Сервер получает от клиента номер задачи, которую нужно удалить, затем удаляет её из списка и перезаписывает в файл обновленный список задач. В случае возникновения ошибки сервер отправляет сообщение об этом клиенту.
4. Изменение задачи. Сервер получает от клиента номер задачи, которую нужно изменить, затем производит удаление старой задачи под указанным номером и добавление новой.

5. Вычисление времени. Сервер поочередно просматривает триггеры каждой задачи и, в зависимости от типа триггера, ищет следующее время его срабатывания.
6. Отправка списка задач клиенту. Сервер выполняет загрузку списка задач и его последующую передачу клиенту.

По описанию работы клиента и сервера можно составить UML–диаграмму последовательности (рисунок 1), иллюстрирующую описанные выше действия.

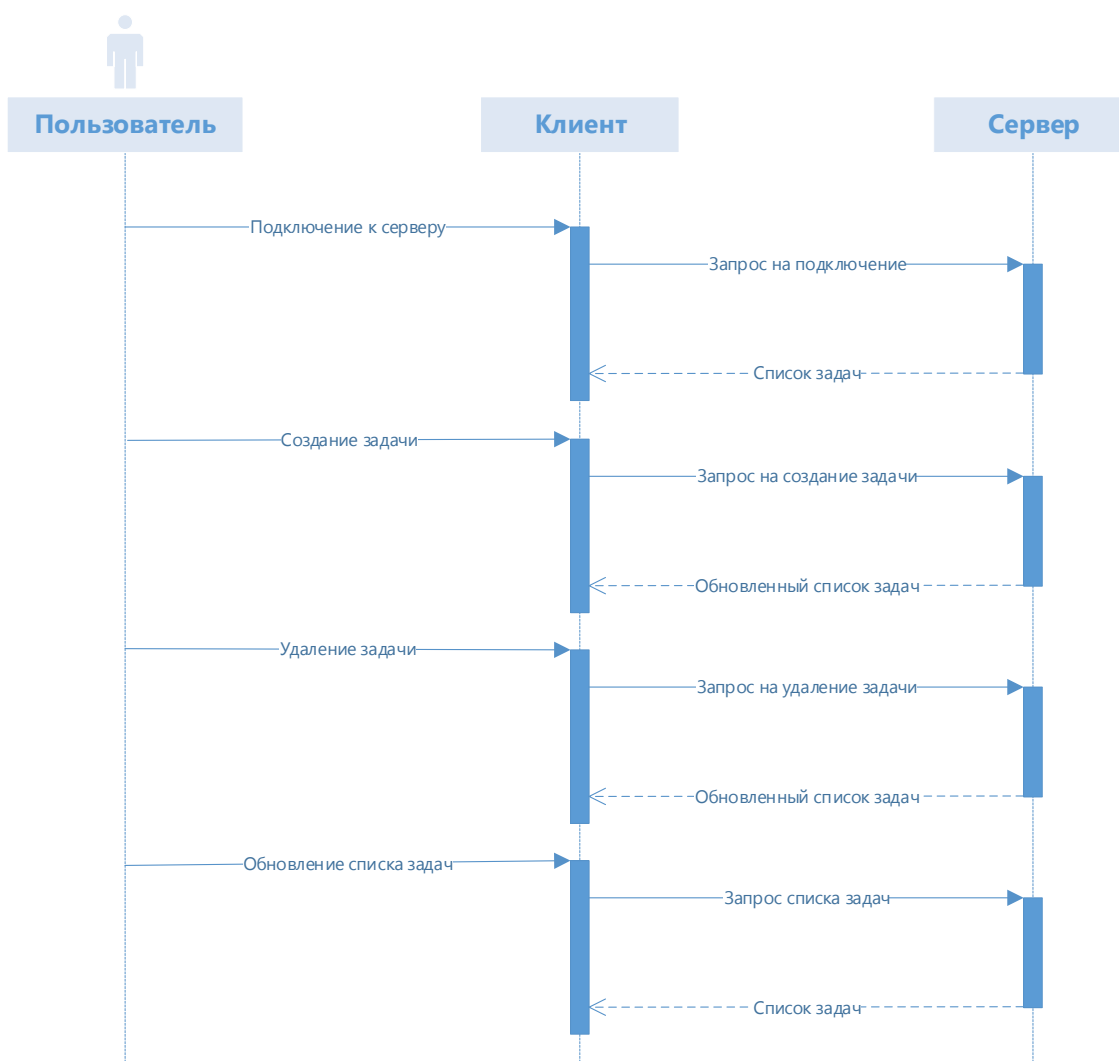


Рисунок 1 – UML–диаграмма последовательности

2.2 Разработка клиентской части приложения

Одной из основных задач клиента является взаимодействие с пользователем. Именно поэтому клиентское приложение должно иметь простой и понятный графический интерфейс, который позволит клиенту добавлять, изменять, удалять или получать задачи. На основе этого описания можно построить UML–диаграмму вариантов использования клиента, представленную на рисунке 2.



Рисунок 2 – UML–диаграмма вариантов использования клиента

Клиент начинает работу с загрузки графического окна приложения. Для дальнейшей работы пользователю нужно открыть окно настройки сервера и ввести необходимые параметры для подключения (IP–адрес сервера и порт). Если подключение к серверу завершится неудачно, приложение уведомит пользователя об этом. Пользователь так же получит уведомление, если попытается создать задачу или обновить их список без подключения к серверу. Успешно подключившись к серверу, пользователь может выполнить следующие действия: создание задачи, изменение задачи, удаление задачи, обновление списка работающих задач. При создании задачи пользователь может ввести ее имя и описание, добавить действия и триггеры, защитить задачу паролем. Обязательным для заполнения является поле имени задачи.

В тех случаях, если пользователь не ввел имя задачи или задача с таким именем уже существует, приложение уведомляет об этом и блокирует

переход на следующую вкладку окна создания задачи, до тех пор, пока вводимая информация не будет правильной. Аналогично с добавлением действий к задаче: она должна иметь хотя бы одно действие в списке. Удаление и изменение задачи становятся доступными после выбора одной задачи из списка. Изменение задачи практически не отличается от ее создания, за исключением того, что пользователю открывается окно с заполненными полями ввода. При обновлении списка задач клиент запрашивает новый список и заносит его в компонент listView главной формы. В конце каждой операции клиент отправляет серверу сообщение с текстовой командой или информацией о задаче. Если операция успешно выполнялась – клиент обновляет список действующих задач, если нет – уведомляет об ошибке.

Используя изложенную информацию, можно построить UML– диаграмму деятельности клиента, представленную на рисунке 3.

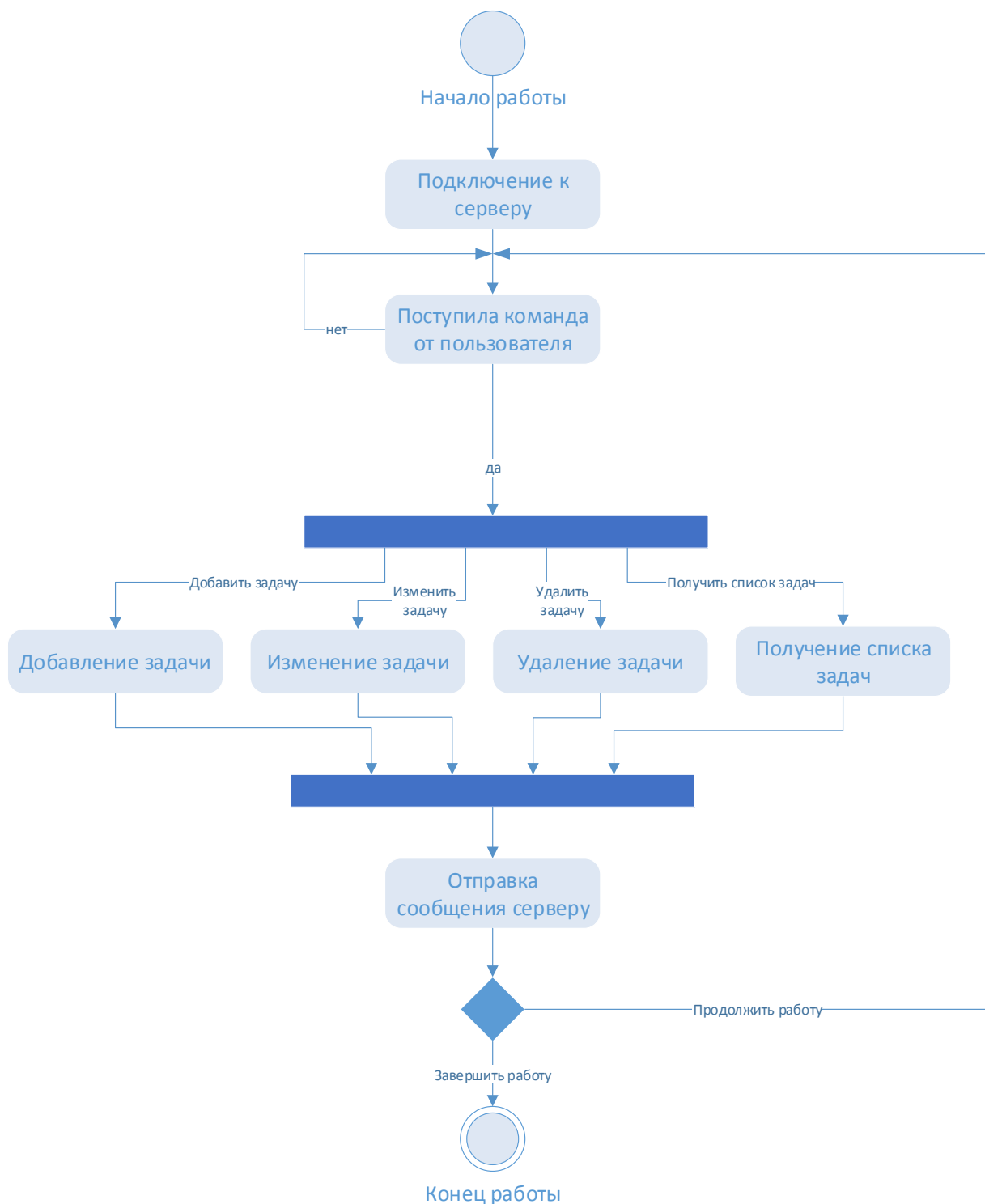


Рисунок 3 – UML–диаграмма вариантов использования клиента

По диаграмме деятельности клиента можно описать следующие его состояния:

1. Состояние подключения к серверу. Клиент производит попытку подключения к серверу.
2. Состояние ожидания действия от пользователя. Клиент ожидает от пользователя одно из четырех действий.

3. Состояние добавления задачи. Клиент формирует запрос о добавлении задачи серверу.
4. Состояние изменения задачи. Клиент формирует запрос об изменении задачи серверу.
5. Состояние удаления задачи. Клиент формирует запрос об удалении задачи серверу.
6. Состояние получения списка задач. Клиент формирует запрос о получении списка задач серверу.
7. Состояние отправки сообщения серверу. Клиент отправляет запрос серверу.

2.3 Разработка серверной части приложения

Серверное приложение было реализовано в виде службы – специальной программы, работающей в фоновом режиме и скрытой от пользователя. Они идеально подходят для реализации серверных процессов в приложениях типа клиент–сервер. Поскольку всё взаимодействие с сервером проходит через программу клиента, графический и консольный интерфейс приложения не был реализован. UML–диаграмма вариантов использования сервера представлена на рисунке 4.



Рисунок 4 – UML–диаграмма вариантов использования сервера.

Серверная часть содержит две основные функции, каждая из которых работает в отдельной нити. Первая нить принимает и обрабатывает входящие запросы клиента. Порядок её действий можно описать следующим образом.

Нить обработки запросов (или же обработчик запросов) создается и запускается сразу после установления подключения по клиентскому сокету. Она находится в бесконечном цикле ожидания, выполняет работу при получении сообщения от клиента, и затем снова уходит в ожидание. Сообщение представляет собой текстовую строку, разбираемую обработчиком запросов для определения нужной клиенту функции. После этого сервер выполняет запрос клиента, а именно: отправляет список задач, добавляет задачу, изменяет задачу или удаляет задачу.

Задачи хранятся в динамическом массиве типа List. Список действующих задач является критическим ресурсом – разделяемым между двумя нитями ресурсом, доступ к которому в текущий момент времени может получить только одна нить. Если действия обработчика изменяют содержимое списка задач (добавляют, изменяют или удаляют задачу), то, во избежание коллизий, сервер блокирует нить выполнения задач на время проделывания запрашиваемой операции. После проделанной работы сервер снимает блокировку с нити.

Передача и получение данных происходят через сетевой поток NetworkStream.

Для отправки списка задач по сетевому потоку сервер выполняет преобразование списка задач в поток байтов (сериализацию объектов), после чего происходит непосредственно передача информации.

Добавление задачи происходит следующим образом: сначала осуществляется ее получение из потока в сериализованном виде, затем выполняется ее десериализация и добавление в конец списка работающих задач.

При удалении задачи клиент посылает серверу номер той задачи, которую требуется удалить. Сервер производит поиск задачи с таким номером и ее изъятие из списка.

В том случае, если в процессе выполнения этих операций возникнет ошибка, клиент получит сообщение об этом. Если операция выполняется

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

успешно, происходит запись сериализованного списка задач в файл «tasks.dat».

Вторая нить отвечает за выполнение действий задач, работая следующим образом. Она поочередно просматривает каждую задачу из списка, проверяя, является ли задача активной («Включена»), и, если да, то просматривает каждый триггер этой задачи, с целью проверки условий активации триггера (в данном приложении этими условиями являются дата и время). Если время активации триггера совпадает с текущим временем, нить запускает все действия, принадлежащие этой задаче, и пересчитывает следующее время активации триггера. Если срок действия триггера истек, нить удаляет его из списка. Если у задачи завершилось время действия всех триггеров, нить удаляет эту задачу.

На основе описанной выше информации можно построить UML–диаграмму деятельности сервера, представленную на рисунке 5.

					ПГУ 2.090301.26.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

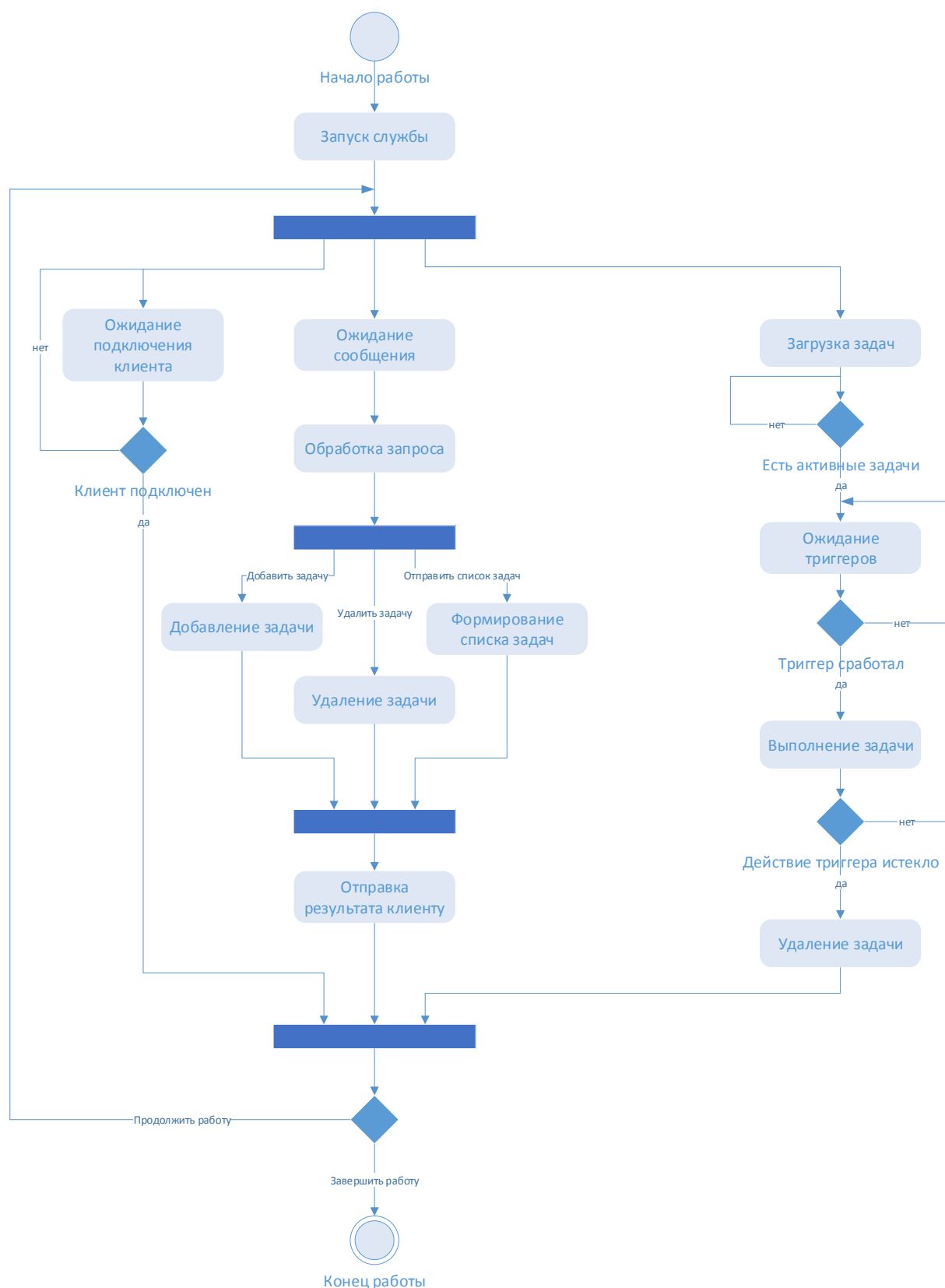


Рисунок 5 – UML-диаграмма вариантов использования клиента

По диаграмме деятельности сервера можно описать следующие его состояния:

1. Состояние запуска службы.

2. Состояние ожидания подключения клиента. Сервер ожидает подключения клиента.
3. Состояние ожидания сообщения. Сервер ожидает сообщение от клиента.
4. Состояние обработки запроса. Сервер определяет, какое действие необходимо выполнить клиенту.
5. Состояние добавления задачи. Сервер добавляет полученный от клиента объект задачи в список.
6. Состояние удаления задачи. Сервер удаляет задачу под указанным номером из списка.
7. Состояние формирования списка задач. Сервер подготавливает к отправке список задач.
8. Состояние отправки результата. Сервер отправляет результат работы клиенту.
9. Состояние загрузки задач. Сервер загружает из файла сохраненные задачи.
10. Состояние ожидания триггеров. Сервер ожидает срабатывания триггера какой-либо задачи.
11. Состояние выполнения задачи. Сервер выполняет задачу.
12. Состояние завершения работы.

3 Описание приложения

3.1 Руководство пользователя

После запуска клиентского приложения откроется главное окно программы (рисунок 6). Оно содержит таблицу задач, кнопки для добавления, изменения, удаления задачи. Над таблицей расположена кнопка обновления списка задач, под тремя кнопками с действиями расположены настройки сервера.

Таблица задач содержит следующие столбцы: название задачи, ее описание, дата последнего запуска, дата следующего запуска, триггеры задачи.

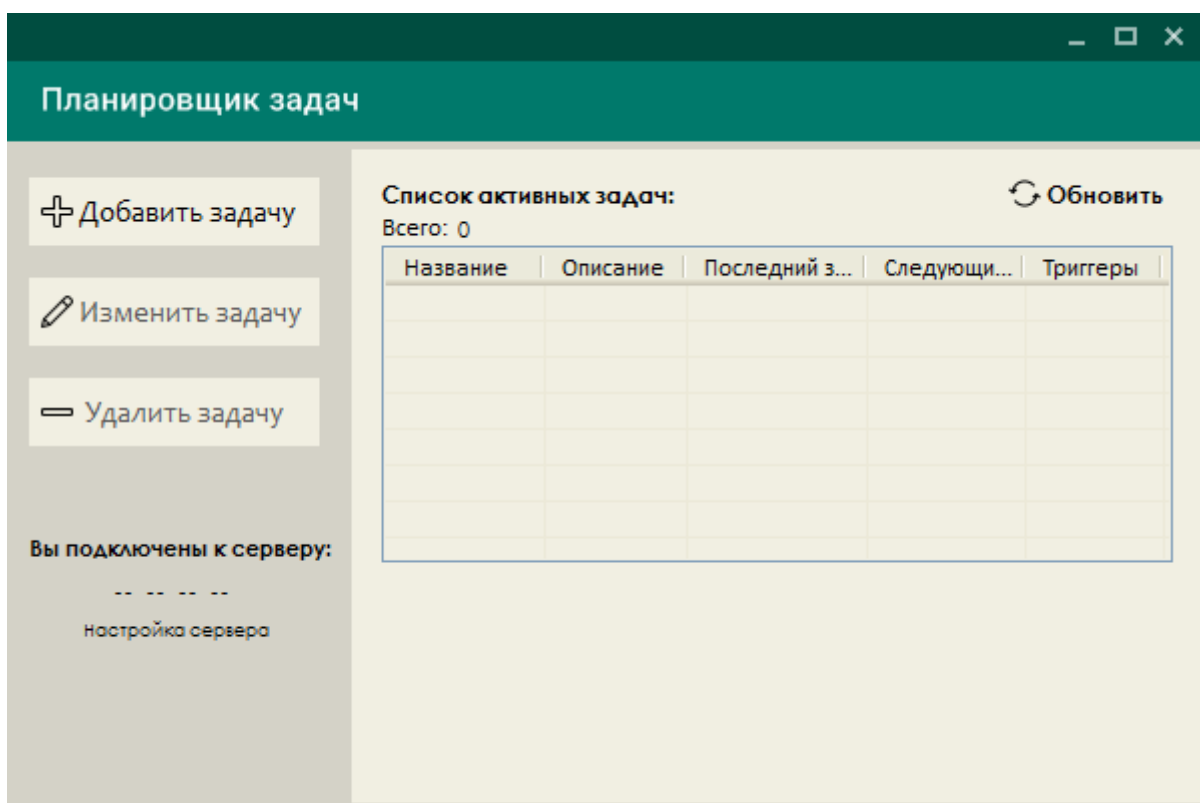


Рисунок 6 – Главное окно клиентского приложения

Прежде чем пользователь сможет получить список имеющихся задач или добавить новую, он должен подключиться к серверу, нажав на «Настройка сервера», после чего откроется следующее окно (рисунок 7):

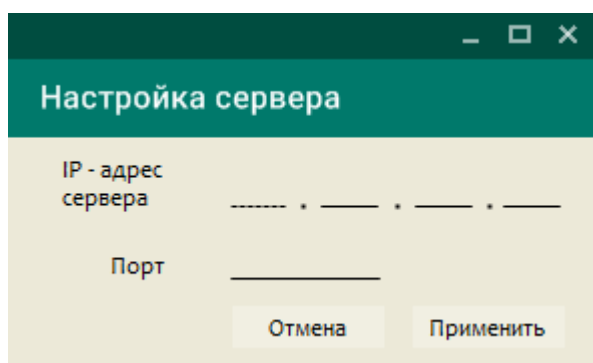


Рисунок 7 – Окно настройки сервера

Пользователь должен ввести IP–адрес и порт подключения сервера, затем нажать «Применить». Если клиент успешно подключится к серверу, форма будет выглядеть следующим образом:

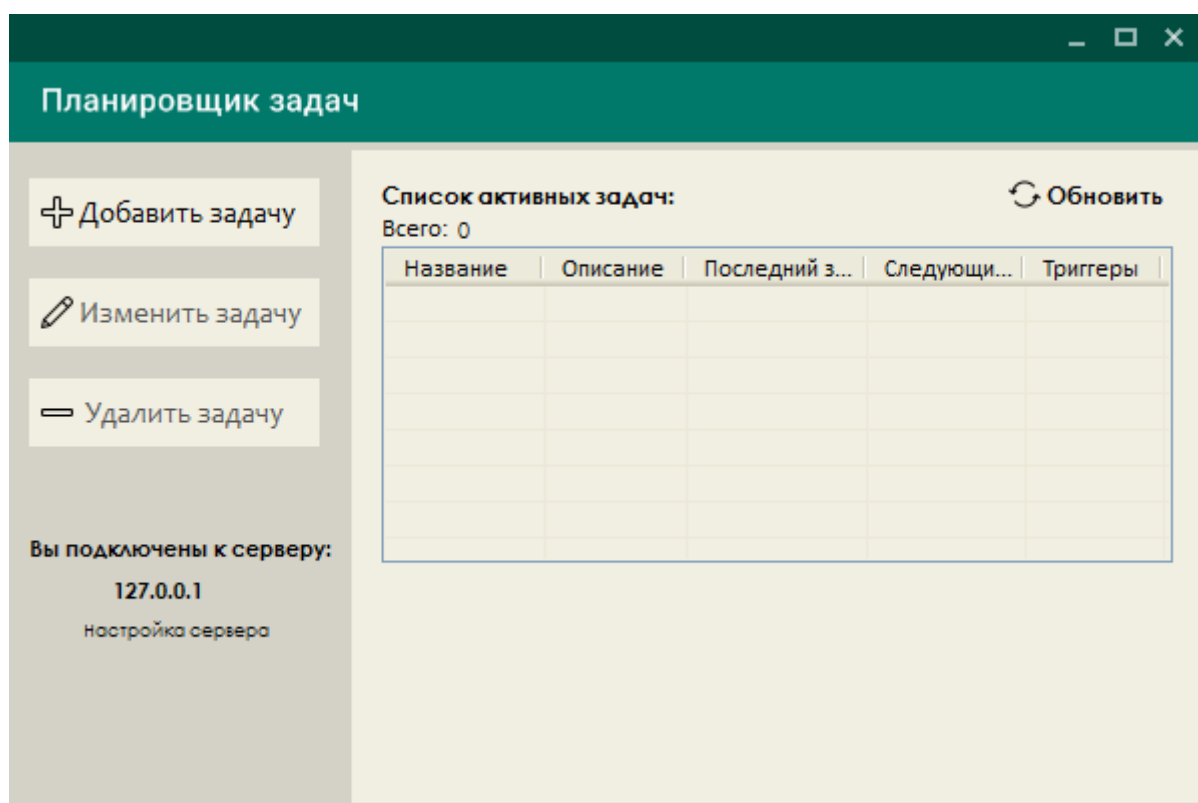


Рисунок 8 – Главное окно клиентского приложения после подключения к серверу
Иначе пользователь получит сообщение об ошибке (рисунок 9):

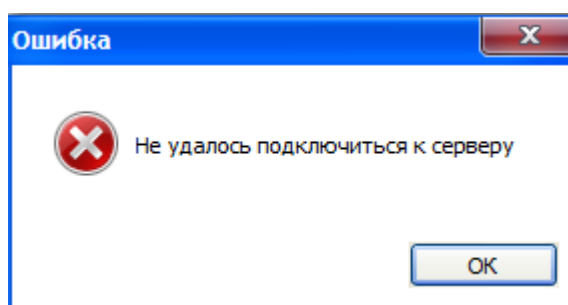


Рисунок 9 – Ошибка при подключении к серверу

Изм.	Лист	№ докум.	Подпись	Дата

Если пользователь попытается обновить список или создать задачу без подключения к серверу, то получит следующее предупреждение (рисунок 10), предупреждение для обновления аналогично:

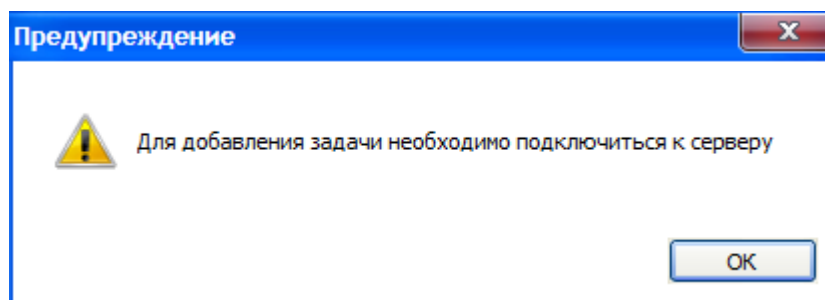


Рисунок 10 – Предупреждение о необходимости подключения к серверу

Для того, чтобы обновить список запущенных задач, пользователю нужно нажать на «Обновить» (рисунок 11).

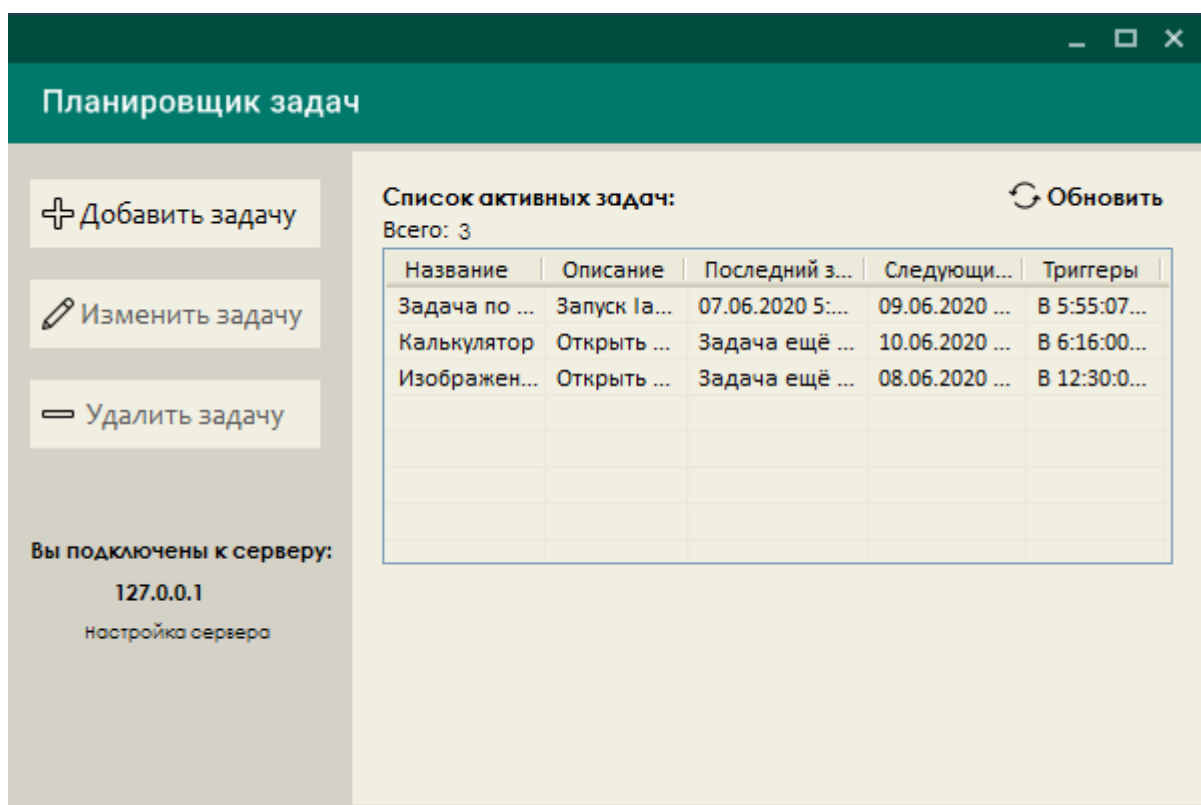


Рисунок 11 – Обновление списка задач

Для создания новой задачи необходимо выбрать «Добавить задачу», после чего откроется окно создания задачи (рисунок 12). В первой вкладке «Общие» вводится имя и описание задачи.

Рисунок 12 – Создание задачи. Вкладка «Общие»

Поле «Имя» является обязательным для заполнения. Если поле останется пустым, то на экране появится следующее предупреждение (рисунок 13):

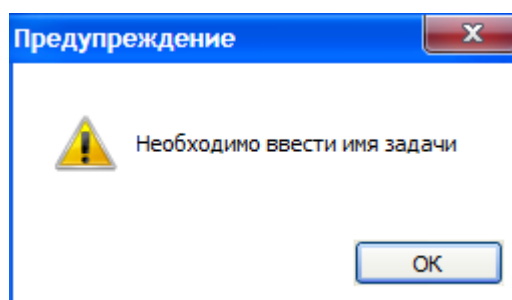


Рисунок 13 – Предупреждение о необходимости ввода имени

Если задача с таким именем уже существует, на экране также появится предупреждение (рисунок 14):

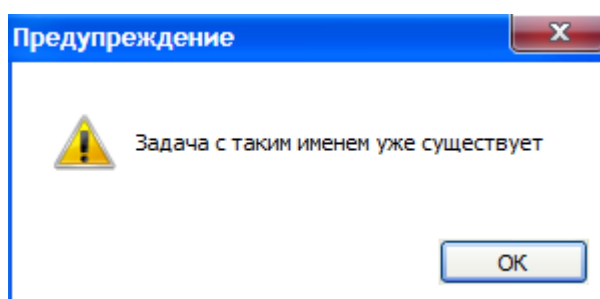


Рисунок 14 – Предупреждение о необходимости переименования задачи

Если введенная информация корректна, откроется следующая вкладка «Триггеры» (рисунок 15).

Создание новой задачи

Триггеры

Условия запуска задачи (не обязательны)

Триггер	Описание	Состояние

Создать Изменить Удалить

Назад Далее

Рисунок 15 – Создание задачи. Вкладка «Триггеры»

Для создания триггера необходимо нажать на кнопку «Создать», после чего откроется окно создания триггера (рисунок 16). Добавление триггеров не является обязательным. Если задача не содержит триггеров, она выполнится сразу же после ее получения сервером.

Создание триггера

Начать задачу: По расписанию

Параметры

☒ Однократно
☐ Ежедневно
☐ Еженедельно
☐ Ежемесячно

Начать: 5:58:07 01.06.2020

Дополнительные параметры

☒ Отложить на (произвольная задержка): 10 мин.
☐ Срок действия: 5:58:07 01.06.2020
☒ Включено

Отмена Сохранить

Рисунок 16 – Создание задачи. Окно создания триггера

Создаваемый триггер может быть определен множеством различных параметров. Общие для всех типов триггеров находятся в группе «Дополнительные параметры».

В нее входят:

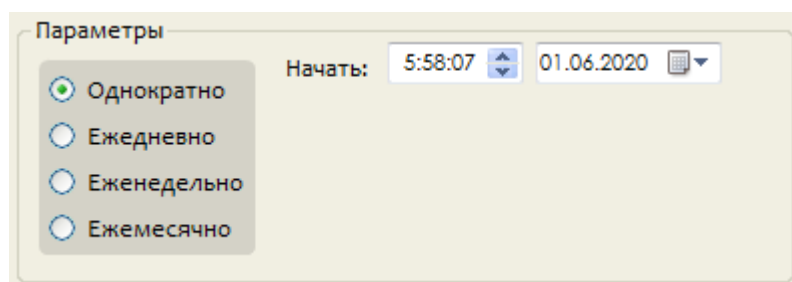
«Задержка» – время, на которое откладывается срабатывание триггера. Значение может быть выбрано из списка или введено вручную, в этом случае необходимо ввести целое число и единицу измерения времени («сек.» – секунд, «мин.» – минут, «ч.» – часов, «дн.» – дней).

«Срок действия» – дата и время, после наступления которых триггер перестанет быть действующим и будет удален из списка.

«Включено» – определяет, является ли триггер действующим.

Различные для каждого типа триггера настройки находятся в группе «Параметры», общим параметром является только время начала действия триггера. Выбрать тип триггера можно, нажав на один из следующих видов триггеров:

«Однократно». Для такого типа триггера необходимо указать дату и время срабатывания (рисунок 17).



Параметры

Начать: 5:58:07 01.06.2020

☒ Однократно

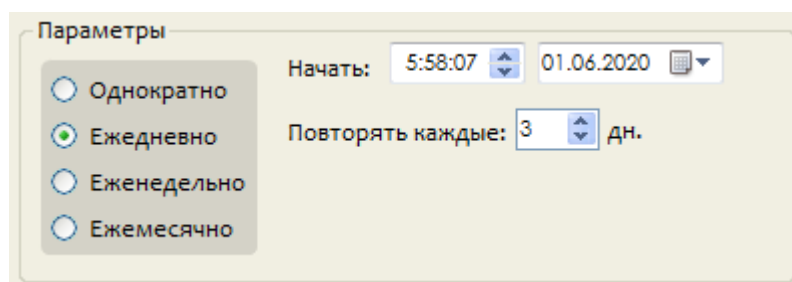
☐ Ежедневно

☐ Еженедельно

☐ Ежемесячно

Рисунок 17 – Тип триггера «Однократно»

«Ежедневно». Для такого типа триггера необходимо указать дату и время срабатывания, частоту повторения (количество дней) (рисунок 18).



Параметры

Начать: 5:58:07 01.06.2020

Повторять каждые: 3 дн.

☐ Однократно

☒ Ежедневно

☐ Еженедельно

☐ Ежемесячно

Рисунок 18 – Тип триггера «Ежедневно»

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

32

«Еженедельно». Для такого типа триггера необходимо указать дату и время срабатывания, частоту повторения (количество недель), дни недели, в которые триггер будет активирован (рисунок 19).

Параметры

Начать: 5:58:07 01.06.2020

Повторять каждую 2 неделю в

☐ Однократно

☐ Ежедневно

☒ Еженедельно

☐ Ежемесячно

☐ Пн ☒ Вт ☐ Ср ☒ Чт

☐ Пт ☒ Сб ☐ Вс

Рисунок 19 – Тип триггера «Еженедельно»

«Ежемесячно». Для такого типа триггера необходимо указать дату и время срабатывания, выбрать подтип триггера. Если нажать на «Дни», выберется подтип «дни в месяце», если нажать на «В», выберется подтип «дни в неделях месяца». Для каждого подтипа указываются месяцы срабатывания триггера, для первого подтипа – определенные дни в месяце, для второго подтипа – определенные дни недели в месяце.

Параметры

Начать: 5:58:07 01.06.2020

Месяцы: ☐ Январь ☒ Февраль

Дни: ☐ 1 ☒ 2

☐ В: ☒ 1 ☐ 2

☐ Однократно

☐ Ежедневно

☐ Еженедельно

☒ Ежемесячно

Рисунок 20 – Тип триггера «Ежемесячно», подтип «дни в месяце»

Параметры

Начать: 5:58:07 01.06.2020

Месяцы: ☐ Январь ☒ Февраль

Дни: ☒ 1 ☐ 2

В: ☐ Пн ☒ Вт

☐ Однократно

☐ Ежедневно

☐ Еженедельно

☒ Ежемесячно

Рисунок 21 – Тип триггера «Ежемесячно», подтип «дни в неделях месяца»

После нажатия кнопки «Сохранить» триггер добавится в таблицу. Для изменения или удаления триггера необходимо выбрать его из таблицы и нажать на кнопку «Изменить» или «Удалить».

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

33

Далее идет вкладка «Действия» (рисунок 22).

Рисунок 22 – Создание задачи. Вкладка «Действия»

Для добавления действия нужно нажать на кнопку «Создать», после чего откроется окно создания действия (рисунок 23).

Рисунок 23 – Окно создания действия.

После нажатия на кнопку «Обзор» появится диалоговое окно открытия файла, в котором пользователю необходимо выбрать программу или файл для выполнения. Параметр «Включено» определяет, является ли задача действующей.

После выбора «Сохранить», задача добавится в список. Изменение и удаление действия происходят аналогично действиям во вкладке «Триггеры».

Если задача не содержит ни одного действия, то на экране появится следующее предупреждение (рисунок 24):

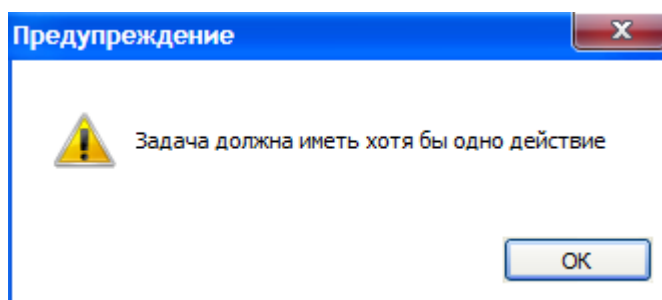


Рисунок 24 – Предупреждение о необходимости добавления действия

Если список действий не пуст, откроется последняя вкладка «Безопасность» (рисунок 25).

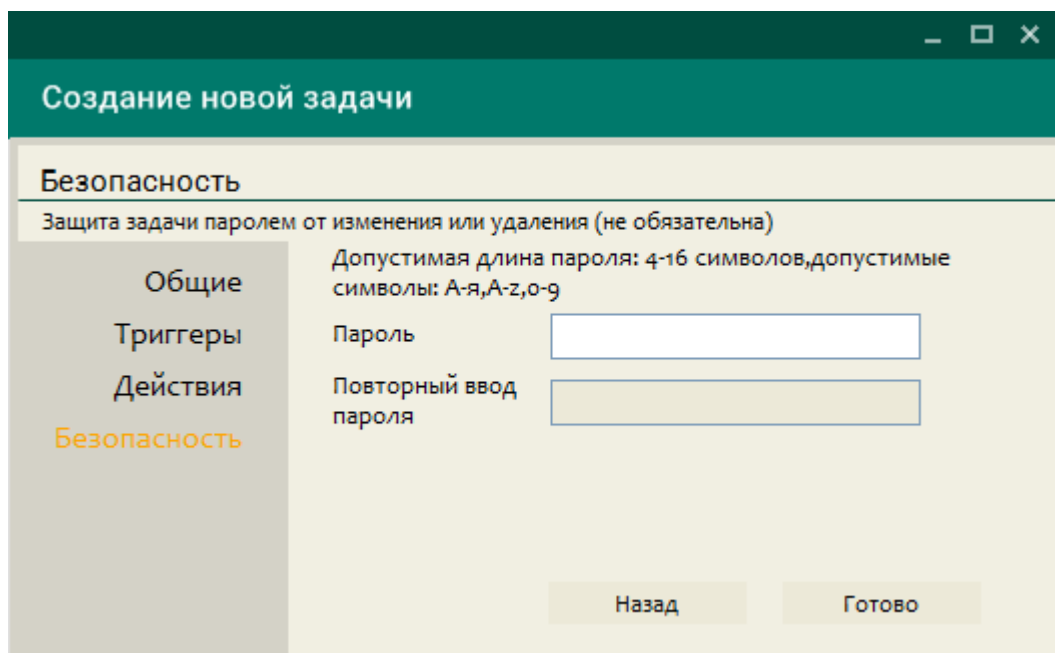


Рисунок 25 – Создание задачи. Вкладка «Безопасность»

По желанию пользователя, задачу можно защитить паролем, который должен удовлетворять следующим требованиям:

Длина пароля: от 4 до 16 символов.

Допустимые символы: кириллический и латинский алфавит, цифры.

Если пароль введен некорректно, на форме появится уведомление. На рисунках 26 – 28 представлены ошибки, встречающиеся при вводе пароля.

Создание новой задачи

Безопасность

Защита задачи паролем от изменения или удаления (не обязательна)

Допустимая длина пароля: 4-16 символов, допустимые символы: A-я, A-z, 0-9

Общие

Триггеры

Действия

Безопасность

Пароль

Повторный ввод пароля

Пароль слишком короткий

Назад Готово

Рисунок 26 – Пароль недостаточной длины

Создание новой задачи

Безопасность

Защита задачи паролем от изменения или удаления (не обязательна)

Допустимая длина пароля: 4-16 символов, допустимые символы: A-я, A-z, 0-9

Общие

Триггеры

Действия

Безопасность

Пароль

Повторный ввод пароля

Пароль содержит недопустимые символы

Назад Готово

Рисунок 27 – Пароль содержит недопустимые символы

Создание новой задачи

Безопасность

Защита задачи паролем от изменения или удаления (не обязательна)

Допустимая длина пароля: 4-16 символов, допустимые символы: А-я, А-z, 0-9

Общие

Триггеры

Действия

Безопасность

Пароль: [masked]

Повторный ввод пароля: [masked]

Пароли не совпадают

Назад Готово

Рисунок 28 – Пароли не совпадают

После нажатия «Готово», задача добавится в таблицу в главном окне приложения. Для изменения или удаления задачи необходимо выбрать ее из списка, затем нажать на кнопку с соответствующим действием (рисунки 29 – 30).

Планировщик задач

+ Добавить задачу

✎ Изменить задачу

= Удалить задачу

Вы подключены к серверу:
127.0.0.1
настройка сервера

Список активных задач: Обновить

Всего: 3

Название	Описание	Последний з...	Следующи...	Триггеры
Задача по ...	Запуск la...	07.06.2020 5:...	09.06.2020 ...	В 5:55:07...
Калькулятор	Открыть ...	Задача ещё ...	10.06.2020 ...	В 6:16:00...
Изображен...	Открыть ...	Задача ещё ...	08.06.2020 ...	В 12:30:0...

Рисунок 29 – Изменение задачи

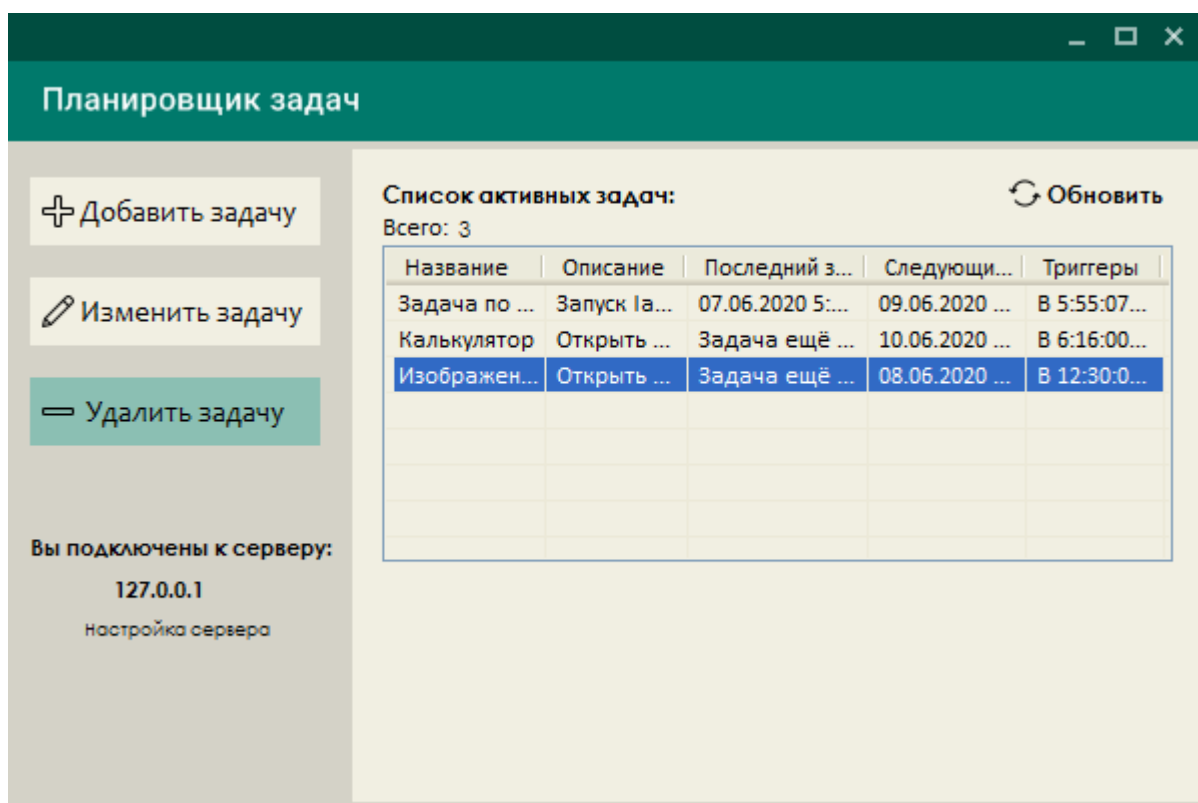


Рисунок 30 – Удаление задачи

Окно изменения задачи аналогично окну создания, но при этом вся информация о задаче заносится в поля ввода при открытии окна (рисунок 31).

Изменение задачи

Общие

Общая информация о задаче

Общие

Триггеры

Действия

Безопасность

Имя: Калькулятор

Автор: User

Описание: Открыть калькулятор

Назад Далее

Рисунок 31 – Окно изменения задачи

Если задача была защищена паролем, то перед ее изменением или удалением появится форма для его ввода (рисунок 32).

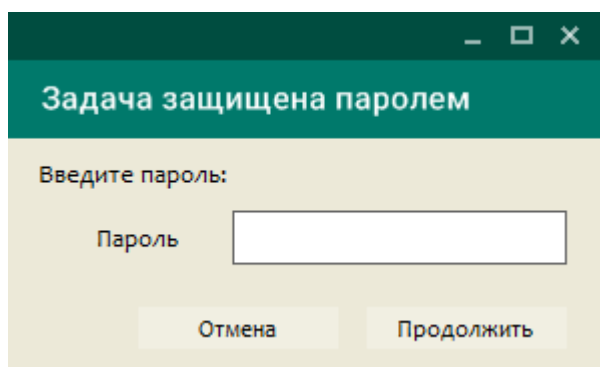


Рисунок 32 – Защита задачи паролем

Если пользователем введен неверный пароль, сообщение об этом появится на экране (рисунок 33):

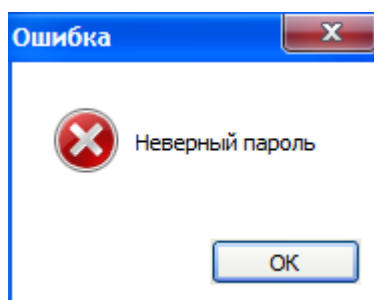


Рисунок 33 – Ошибка при вводе пароля

При корректном вводе пароля приложение выполнит запрашиваемое изменение.

3.2 Спецификация клиента

Класс MainForm

Класс главной формы программы, запускающейся при открытии приложения. Содержит таблицу работающих задач, а также кнопки для работы с ними.

Основные поля и методы класса MainForm:

clientSocket – поле класса MainForm, клиентский сокет типа TcpClient, через который происходит подключение к серверу.

ReceivedTasks – поле класса MainForm, динамический массив, хранящий

полученный от сервера список задач.

SetTasksToListView – метод класса MainForm, загружает полученные задачи в таблицу.

Класс TaskForm

Класс формы создания задачи, запускающейся при добавлении или изменении задачи.

Основные поля и методы класса TaskForm:

data – класс, в который записывается информация о создаваемой задаче.

GetGenData – метод класса TaskForm, заносит информацию в класс data с вкладки «Общее» окна создания задачи.

GetSafeData – метод класса TaskForm, заносит информацию в класс data с вкладки «Безопасность» окна создания задачи, а также проверяет, одинаковы ли пароли из двух текстовых полей ввода.

Класс TriggerForm

Класс формы создания триггера, запускающейся при добавлении или изменении триггера.

Основные поля и методы класса TriggerForm:

DailyCreate – метод класса TriggerForm, добавляет на форму поле ввода информации для типа триггера «Ежедневный».

DailyRemove – метод класса TriggerForm, удаляет с формы поле ввода, добавленное методом DailyCreate.

WeeklyCreate – метод класса TriggerForm, добавляет на форму поля ввода информации для типа триггера «Еженедельный».

WeeklyRemove – метод класса TriggerForm, удаляет с формы поля ввода, добавленные методом WeeklyCreate.

MonthlyCreate – метод класса TriggerForm, добавляет на форму поля ввода информации для типа триггера «Ежемесячный».

MonthlyRemove – метод класса TriggerForm, удаляет с формы поля ввода, добавленные методом MonthlyCreate.

DaysCreate – метод класса TriggerForm, добавляет на форму поля ввода информации для подтипа ежемесячного триггера «дни в месяце».

DaysRemove – метод класса TriggerForm, удаляет с формы поля ввода, добавленные методом DaysCreate.

InCreate – метод класса TriggerForm, добавляет на форму поля ввода информации для подтипа ежемесячного триггера «дни в неделях месяца».

InRemove – метод класса TriggerForm, удаляет с формы поля ввода, добавленные методом InCreate.

InitData – метод класса TriggerForm, заносит в поля ввода информацию о триггере, если для него выбрано действие «Изменить».

Класс ActionForm

Класс формы создания действия, запускающейся при добавлении или изменении действия.

Основные поля и методы класса ActionForm:

InitData – метод класса ActionForm, заносит в поля ввода информацию о действии, если для него выбрано действие «Изменить».

Класс ServerForm

Класс формы настроек сервера, содержит поле ввода IP–адреса сервера и порта подключения.

Класс PasswordForm

Класс формы ввода пароля, открывающейся при защите задачи паролем от изменения или удаления.

Класс General

Класс, добавляющий на форму создания задачи поля для ввода информации из вкладки «Общие».

Основные поля и методы класса General:

GenInit – метод класса General, добавляет на форму поля ввода информации из вкладки «Общие».

GenRemove – метод класса General, удаляет с формы поля ввода, добавленные методом GenInit.

Класс Triggers

Класс, добавляющий на форму создания задачи поля для ввода информации из вкладки «Триггеры».

Основные поля и методы класса Triggers:

TrigInit – метод класса Triggers, добавляет на форму поля ввода информации из вкладки «Триггеры».

TrigRemove – метод класса Triggers, удаляет с формы поля ввода, добавленные методом TrigInit.

Класс Actions

Класс, добавляющий на форму создания задачи поля для ввода информации из вкладки «Действия».

Основные поля и методы класса Actions:

ActInit – метод класса Actions, добавляет на форму поля ввода информации из вкладки «Действия».

ActRemove – метод класса Actions, удаляет с формы поля ввода, добавленные методом ActInit.

Класс Safety

Класс, добавляющий на форму создания задачи поля для ввода информации из вкладки «Безопасность» и проверяющий правильность ввода паролей.

Основные поля и методы класса Safety:

passval – поле класса Actions, переменная логического типа, принимающая значение «true», если пароль введен корректно, «false» – если введен

некорректно.

ActInit – метод класса Actions, добавляет на форму поля ввода информации из вкладки «Действия».

ActRemove – метод класса Actions, удаляет с формы поля ввода, добавленные методом ActInit.

CheckPassword – метод класса Actions, проверяет введенный пароль на соответствие минимальной длине и отсутствие недопустимых символов.

Класс ClientSocket

Класс, реализующий взаимодействие клиентского приложения с серверным. Позволяет клиенту подключаться к серверу, отправлять и принимать данные.

Основные поля и методы класса ClientSocket:

ip – поле класса ClientSocket, IP–адрес сервера.

port – поле класса ClientSocket, порт подключения сервера.

writerStream – поле класса ClientSocket, поток для передачи данных серверу.

readerStream – поле класса ClientSocket, поток для получения данных от сервера.

CreateClientSocket – метод класса ClientSocket, подключает клиента к серверу, используя поля ip и port, получает поток для отправки и приема данных.

CreateMessage – метод класса ClientSocket, при помощи сериализации преобразует задачу в поток байтов для дальнейшей передачи серверу.

CreateTasks – метод класса ClientSocket, при помощи десериализации преобразует полученные от сервера данные в массив задач.

DeleteTask – метод класса ClientSocket, отправляет серверу сообщение об удалении указанной задачи.

SendMessage – метод класса ClientSocket, отправляет серверу сообщения о создании или изменении задачи, получении списка задач.

ReceiveTasks – метод класса ClientSocket, получает в сериализованном виде список задач от сервера.

3.3 Спецификация сервера

Класс TaskManager

Главный класс серверной службы. Создает и запускает нити, выполняющие основные функции приложения.

Основные поля и методы класса TaskManager:

port – поле класса TaskManager, номер порта, по которому выполняется подключение к серверу.

listener – поле класса TcpListener, прослушивающий входящие подключения по порту port.

OnStart – метод класса TcpListener, обработчик события запуска службы. Запускает listener, выполняет соединение клиента с сервером и создает нить для приема и обработки клиентских сообщений.

OnStop – метод класса TcpListener, обработчик события остановки службы. Закрывает объект прослушивания listener.

Класс Client

Класс отдельного клиентского подключения, содержит действия по работе с подключенным клиентом.

Основные поля и методы класса Client:

index – поле класса Client, номер задачи, которую необходимо изменить или удалить. Если же поле index равно –1, происходит создание новой задачи.

Process – метод класса Client, обработчик клиентских сообщений. Производит разбор текстового сообщения, и, в зависимости от содержимого, вызывает нужные функции из класса TaskHandler.

Класс TaskHandler

Класс, содержащий все функции по работе с задачами: функцию загрузки, добавления, удаления и т.д.

Основные поля и методы класса TaskHandler:

LoadTasks – метод класса TaskHandler, десериализует сохраненные в файл «tasks.dat» задачи и заносит их в массив Tasks.

ReceiveTask – метод класса TaskHandler, получает от клиента созданную или измененную задачу, затем сохраняет её в файл «1task.dat».

AddTask – метод класса TaskHandler, добавляет полученную в методе ReceiveTask задачу в массив Tasks. Если задача была создана, она добавляется в конец списка, если изменена – на место старой задачи. Обновленный список задач записывается в файл «tasks.dat».

DeleteTask – метод класса TaskHandler, удаляет полученную в методе ReceiveTask задачу из массива Tasks. Обновленный список задач записывается в файл «tasks.dat».

SendTasks – метод класса TaskHandler, считывает задачи из файла «tasks.dat» и отправляет их клиенту через поток типа NetworkStream.

CountTime – метод класса TaskHandler, вычисляет следующее время активации триггеров для каждой задачи.

DoingTasks – метод класса TaskHandler, проверяет условия активации триггеров каждой задачи, и, если триггер сработал, выполняет действия, принадлежащие этой задаче.

3.4 Библиотека Data

Класс Data

Класс, содержащий всю информацию об отдельной задаче. Вместе со своими вложенными структурами класс помечен атрибутом Serializable для возможности сериализации.

					ПГУ 2.090301.26.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

Основные поля и методы класса Data:

genData – структура типа GeneralData, содержащая общую информацию о задаче.

Включает в себя:

name – поле структуры genData, имя задачи.

author – поле структуры genData, имя учетной записи пользователя, создавшего задачу.

description – поле структуры genData, описание задачи.

T – динамический массив структур типа TriggerData. Каждый элемент массива содержит информацию об одном из триггеров задачи.

Структура типа TriggerData включает в себя:

repeat – поле структуры типа TriggerData, тип триггера.

begin – структура типа Begin, содержащая дату и время запуска задачи.

Включает в себя:

time – поле структуры begin, время запуска.

date – поле структуры begin, дата запуска.

day – поле структуры типа TriggerData, периодичность повторения для типа триггера «Ежедневный».

week – структура типа Week, содержащая параметры для типа триггера «Еженедельный».

Включает в себя:

week – поле структуры week, периодичность повторения.

days – поле структуры week, дни недели, в которые будет активирован триггер.

month – структура типа Month, содержащая параметры для типа триггера «Ежемесячный».

Включает в себя:

month – поле структуры month, месяцы, в которые будет активирован триггер.

days – поле структуры month, дни месяцев, в которые будет активирован триггер (для подтипа «дни в месяце»).

weeks – поле структуры month, номер недель, в которые будет активирован триггер (для подтипа «дни в неделях месяца»).

wdays – поле структуры month, дни недели, в которые будет активирован триггер (для подтипа «дни в неделях месяца»).

delay – поле структуры типа TriggerData, время задержки запуска триггера.

doing – структура типа Doing, содержащая дату и время окончания действия триггера.

Включает в себя:

time – поле структуры doing, время окончания.

date – поле структуры doing, дата окончания.

enable – поле структуры типа TriggerData, показывает, включен ли триггер.

tableDescription – поле структуры типа TriggerData, содержит данные для отображения в таблице триггеров в клиентском приложении.

A – динамический массив структур типа ActionData. Каждый элемент массива содержит информацию об одном из действий задачи.

Структура типа ActionData включает в себя:

path – поле структуры типа ActionData, путь к файлу или программе, который будет открыт или запущен.

enable – поле структуры типа ActionData, показывает, включено ли действие.

password – поле класса Data, пароль задачи.

lasttime – поле класса Data, время последнего запуска задачи.

nexttime – поле класса Data, время следующего запуска задачи.

					ПГУ 2.090301.26.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе данной работы было создано клиент–серверное приложение, планирующее выполнение задач на локальных и удаленных ЭВМ. Приложение полностью соответствует всем заявленным требованиям. Планировщик задач может получать, создавать, изменять и удалять задачи как локального, так и удаленного устройства. Каждая задача может иметь свой список триггеров и действий. Приложение позволяет гибко настраивать условия выполнения задач, предоставляя разнообразные настройки триггеров, к примеру, четыре типа триггеров и прилагающиеся к ним параметры, а также дополнительные настройки вроде отложенного выполнения и времени действия.

Функционал клиентской и серверной частей был разработан с использованием объектно–ориентированного языка программирования С#. Графический интерфейс клиента был создан при помощи интерфейса программирования приложений Windows Forms. Сетевое взаимодействие между клиентом и сервером было реализовано с использованием протокола ТСР/ІР.

В дальнейшем приложение можно улучшить, добавив новые типы триггеров (выполнение задачи при входе в систему, при запуске системы, простое компьютера, при наступлении события, при блокировании или разблокировании рабочей станции), добавив дополнительные параметры выполнения задачи (повтор, сбой выполнения, остановка, принудительная остановка). Также можно добавить другие типы действия задачи, например, вывод уведомления или отправка электронной почты.

Во время выполнения работы были получены знания основных принципов объектно–ориентированного программирования. Усовершенствованы навыки создания клиент–серверного приложения и получен опыт работы с языком программирования С#.

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

Список используемых источников

1. Библиотека Microsoft Developer Network. [Электронный ресурс]. URL: <https://msdn.microsoft.com/>
2. Пахомов Б. «С# для начинающих». 2014 г., 432 стр.
3. Страуструп Б. «Программирование. Принципы и практика с использованием С#». 2016 г., 1328 стр.
4. Евдокимов П. «С# на примерах». 2019 г., 320 стр.
5. Билл Вагнер «Наиболее эффективное программирование на С#. 50 способов улучшения кода». 2017 г., 240 стр.
6. Создание сервиса в Visual Studio 2008. [Электронный ресурс]. URL: <http://csharpcoding.org/sozдание-servisa-v-visual-studio-2008/>

Приложение А

Листинги программы клиента

Приложение А.1

Файл «Program.cs»

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

namespace Client
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

Приложение А.2

Файл «MainForm.cs»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

using System.Net;
using System.Net.Sockets;

namespace Client
{
```

```

public partial class MainForm : MaterialForm
{
    public ClientSocket Socket = new ClientSocket();
    public static TcpClient clientSocket = new
    TcpClient(AddressFamily.InterNetwork);
    public List<Data.Data> ReceivedTasks = new
    List<Data.Data>();
    ToolTip tt = new ToolTip();
    int taskindex = -1;
    public MainForm()
    {
        InitializeComponent();
        var materialSkinManager =
        MaterialSkinManager.Instance;
        materialSkinManager.AddFormToManage(this);
        materialSkinManager.Theme =
        MaterialSkinManager.Themes.LIGHT;
        materialSkinManager.ColorScheme = new
        ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Blue50,
        Accent.Teal700, TextShade.WHITE);
        label6.Text = listView1.Items.Count.ToString();
        button2.Enabled = false;
        button3.Enabled = false;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (!clientSocket.Connected)
        {
            MessageBox.Show("Для добавления задачи
            необходимо подключиться к серверу", "Предупреждение",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        else
        {
            TaskForm form2 = new TaskForm(this, Socket,
            taskindex);
            form2.Show();
        }
    }

    private void label3_Click(object sender, EventArgs e)
    {
        ServerForm form3 = new ServerForm(this,
        clientSocket, Socket);
        form3.Show();
        label3.Enabled = false;
    }

    public void SetTasksToListView()
    {
        listView1.Items.Clear();
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

51

```

for (int i = 0; i < ReceivedTasks.Count; i++)
{
    String name =
ReceivedTasks.ElementAt(i).genData.name;
    String description =
ReceivedTasks.ElementAt(i).genData.description;
    String lasttime;
    if (ReceivedTasks.ElementAt(i).lasttime == null)
    {
        lasttime = "Задача ещё не была запущена";
    }
else
    {
        lasttime =
ReceivedTasks.ElementAt(i).lasttime;
    }
    String nexttime =
ReceivedTasks.ElementAt(i).nexttime;
    String trigs = null;
    for (int j = 0; j <
ReceivedTasks.ElementAt(i).T.Count; j++)
    {
        trigs = trigs +
ReceivedTasks.ElementAt(i).T.ElementAt(j).tableDescription + ";
";
    }
    ListViewItem item = new ListViewItem(new
string[] { name, description, lasttime, nexttime, trigs });
    listView1.Items.Add(item);
}
label6.Text = Convert.ToString(ReceivedTasks.Count);
}

private void listView1_MouseMove(object sender,
MouseEventArgs e)
{
    ListView listView = ((ListView)sender);
    ListViewItem item = listView.GetItemAt(e.X, e.Y);
    ListViewHitTestInfo info = listView.HitTest(e.X,
e.Y);
    if (item != null && info.SubItem != null &&
info.SubItem.Tag != null)
    {
        tt.ShowAlways = true;
        tt.Active = true;

        tt.SetToolTip(listView, (String)info.SubItem.Tag);
    }
}

```

```

private void label7_Click(object sender, EventArgs e)
{
    if (!clientSocket.Connected)
    {
        MessageBox.Show("Для обновления списка задач
необходимо подключиться к серверу", "Предупреждение",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    else
    {
        Socket.ReceiveTasks(ReceivedTasks, taskindex);
        SetTasksToListView();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    String action = "edit";
    if (taskindex != -1 &&
ReceivedTasks.ElementAt(taskindex).password != null)
    {
        PasswordForm pass = new PasswordForm(this,
Socket, taskindex, action);
        pass.Show();
    }
    else
    {
        TaskForm form2 = new TaskForm(this, Socket,
taskindex);
        form2.Show();
    }
}

private void listView1_ItemSelectionChanged(object
sender, ListViewItemSelectionChangedEventArgs e)
{
    if (listView1.SelectedItems.Count != 0)
    {
        taskindex = listView1.SelectedIndices[0];
        button2.Enabled = true;
        button3.Enabled = true;
    }
    else
    {
        taskindex = -1;
        button2.Enabled = false;
        button3.Enabled = false;
    }
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    String action = "delete";
    if (taskindex != -1 &&
ReceivedTasks.ElementAt(taskindex).password != null)
    {
        PasswordForm pass = new PasswordForm(this,
Socket, taskindex, action);
        pass.Show();
    }
    else
    {
        Socket.DeleteTask(taskindex);
        Socket.ReceiveTasks(ReceivedTasks, taskindex);
        SetTasksToListView();
    }
}
}
}

```

Приложение А.3

Файл «TaskForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

using System.Net;
using System.Net.Sockets;

namespace Client
{
    public partial class TaskForm : MaterialForm
    {
        int taskindex = -1;
        General genpage = new General();
        Triggers trigpage = new Triggers();
        Actions actionpage = new Actions();
        Safety safepage = new Safety();
        public Data.Data data;
        MainForm form;
        ClientSocket Socket;
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

54

```

        public int pagecount = 0;

        public TaskForm(MainForm form, ClientSocket Socket, int
taskindex)
        {
            InitializeComponent();
            this.form = form;
            this.Socket = Socket;
            this.taskindex = taskindex;
            if (form.ReceivedTasks.Count == 0) taskindex = -1;
            if (taskindex == -1)
            {
                data = new Data.Data();
                this.Text = "Создание новой задачи";
            }
        else
        {
            data = form.ReceivedTasks.ElementAt(taskindex);
            this.Text = "Изменение задачи";
        }
        var materialSkinManager =
MaterialSkinManager.Instance;
        materialSkinManager.AddFormToManage(this);
        materialSkinManager.Theme =
MaterialSkinManager.Themes.LIGHT;
        materialSkinManager.ColorScheme = new
ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Amber300,
Accent.Teal700, TextShade.WHITE);
    }
    private void Form2_Load(object sender, EventArgs e)
    {
        genpage.GenInit(this, taskindex);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        pagecount--;
        switch (pagecount)
        {
            case 0:
                this.button1.Enabled = false;
                genpage.GenInit(this, taskindex);
                trigpage.TrigRemove(this);
                break;
            case 1:
                trigpage.TrigInit(this, data, taskindex);
                actionpage.ActRemove(this);
                break;
            case 2:
                actionpage.ActInit(this, data, taskindex);
                safepage.SafeRemove(this);
                break;
        }
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

55

```

    }

    }

    private void button2_Click(object sender, EventArgs e)
    {
        pagecount++;
        int c = 0;
        this.button1.Enabled = true;
        switch (pagecount)
        {
            case 1:
                GetGenData();
                Socket.ReceiveTasks(form.ReceivedTasks,
taskindex);
                for (int i = 0; i <
form.ReceivedTasks.Count; i++)
                {
                    if
(form.ReceivedTasks.ElementAt(i).genData.name ==
data.genData.name)
                    {
                        c++;
                    }
                }
                if (c != 0)
                {
                    MessageBox.Show("Задача с таким именем
уже существует", "Предупреждение", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    pagecount--;
                    c = 0;
                }
                else if (data.genData.name == "")
                {
                    MessageBox.Show("Необходимо ввести имя
задачи", "Предупреждение", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    pagecount--;
                }
                else
                {
                    genpage.GenRemove(this);
                    trigpage.TrigInit(this, data, taskindex);
                }
                break;
            case 2:
                trigpage.TrigRemove(this);
                actionpage.ActInit(this, data, taskindex);
                break;
            case 3:
                if (data.A.Count == 0)
                {

```

```

        MessageBox.Show("Задача должна иметь хотя бы одно действие",
        "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        pagecount--;
    }
    else
    {
        actionpage.ActRemove(this);
        safepage.SafeInit(this, taskindex);
    }
    break;
case 4:
    pagecount--;
    GetSafeData();
    try
    {
        Socket.CreateMessage(data);
        Socket.SendMessage(taskindex);
        Socket.ReceiveTasks(form.ReceivedTasks,
taskindex);

        form.SetTasksToListView();
    }
    catch (SocketException)
    {
        MessageBox.Show("Не удалось создать
задачу.Соединение с сервером разорвано.", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    this.Close();
    break;
    }
}

private void GetGenData()
{
    data.genData.name = genpage.textBox1.Text;
    data.genData.author = genpage.label7.Text;
    data.genData.description =
genpage.richTextBox1.Text;
}
private void GetSafeData()
{
    if (safepage.textBox1.Text != "" &&
safepage.textBox2.Text != ""
        && safepage.textBox1.Text !=
safepage.textBox2.Text)
    {
        safepage.label4.Text = "Пароли не совпадают";
    }
    else if (safepage.textBox1.Text != "" && safepage.textBox2.Text
!= "" && safepage.textBox1.Text != safepage.textBox2.Text)

```

```

{
    safepage.label4.Text = "Пароли не совпадают";
}
else if (safepage.textBox1.Text != "" &&
safepage.textBox2.Text != ""
&& safepage.textBox1.Text ==
safepage.textBox2.Text && safepage.passval == true)
{
    safepage.label4.Text = "";
    data.password = safepage.textBox2.Text;
}
}
}
}

```

Приложение А.4

Файл «TriggerForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

namespace Client
{
    public partial class TriggerForm : MaterialForm
    {
        Triggers trig;
        Data.Data data;
        Data.Data.TriggerData trigData;
        bool m, d, i1, i2 = false;
        int index;
        Label dlabel1 = new System.Windows.Forms.Label();
        Label dlabel2 = new System.Windows.Forms.Label();
        NumericUpDown dnumericUpDown1 = new
System.Windows.Forms.NumericUpDown();

        Label wlabel1 = new System.Windows.Forms.Label();
        Label wlabel2 = new System.Windows.Forms.Label();
        NumericUpDown wnumericUpDown1 = new
System.Windows.Forms.NumericUpDown();
        CheckBox wcheckBox1 = new System.Windows.Forms.CheckBox();
        CheckBox wcheckBox2 = new
System.Windows.Forms.CheckBox();

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

58

```

CheckBox wcheckBox3 = new System.Windows.Forms.CheckBox();
    CheckBox wcheckBox4 = new
System.Windows.Forms.CheckBox();
    CheckBox wcheckBox5 = new
System.Windows.Forms.CheckBox();
    CheckBox wcheckBox6 = new
System.Windows.Forms.CheckBox();
    CheckBox wcheckBox7 = new
System.Windows.Forms.CheckBox();

    CheckedListBox mcheckedListBox1 = new
System.Windows.Forms.CheckedListBox();
    Label mlabel1 = new System.Windows.Forms.Label();
    RadioButton mradioButton1 = new
System.Windows.Forms.RadioButton();
    RadioButton mradioButton2 = new
System.Windows.Forms.RadioButton();

    CheckedListBox dcheckedListBox1 = new
System.Windows.Forms.CheckedListBox();

    CheckedListBox incheckedListBox1 = new
System.Windows.Forms.CheckedListBox();
    CheckedListBox incheckedListBox2 = new
System.Windows.Forms.CheckedListBox();
public TriggerForm(Triggers trig, Data.Data data, int index)
{
    this.trig = trig;
    this.data = data;
    this.index = index;
    InitializeComponent();
    var materialSkinManager =
MaterialSkinManager.Instance;
    materialSkinManager.AddFormToManage(this);
    materialSkinManager.Theme =
MaterialSkinManager.Themes.LIGHT;
    materialSkinManager.ColorScheme = new
ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Blue50,
Accent.Teal700, TextShade.WHITE);
}
private void Form4_Load(object sender, EventArgs e)
{
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker2.Value = DateTime.Now;
    dateTimePicker3.Value = DateTime.Now;
    dateTimePicker4.Value = DateTime.Now;
if (index != -1)
{
    InitData(index);
}
    trig.createbtn.Enabled = false;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

59

```

trig.changebtn.Enabled = false;
    trig.deletebtn.Enabled = false;
}

private void Form4_FormClosing(object sender,
FormClosingEventArgs e)
{
    trig.n = true;
    trig.m = true;
    if (index == -1)
    {
        trig.createbtn.Enabled = true;
        trig.changebtn.Enabled = false;
        trig.deletebtn.Enabled = false;
    }
    else
    {
        trig.createbtn.Enabled = false;
        trig.changebtn.Enabled = true;
        trig.deletebtn.Enabled = true;
    }
}

private void radioButton2_CheckedChanged(object sender,
EventArgs e)
{
    if (radioButton2.Checked == true) { DailyCreate(); }
    if (radioButton2.Checked == false) { DailyRemove(); }
}

private void radioButton3_CheckedChanged(object sender,
EventArgs e)
{
    if (radioButton3.Checked == true) { WeeklyCreate(); }
    if (radioButton3.Checked == false) { WeeklyRemove(); }
}

private void radioButton4_CheckedChanged(object sender,
EventArgs e)
{
    if (radioButton4.Checked == true) { MonthlyCreate(); }
    if (radioButton4.Checked == false) { MonthlyRemove(); }
}

private void mradioButton1_CheckedChanged(object sender,
EventArgs e)
{
    if (mradioButton1.Checked == true) { DaysCreate(); }
}

```

```

if (mradioButton1.Checked == false) { DaysRemove(); }
    }

    private void mradioButton2_CheckedChanged(object sender,
EventArgs e)
    {
        if (mradioButton2.Checked == true) { InCreate(); }
        if (mradioButton2.Checked == false) { InRemove(); }
    }

    private void checkBox1_CheckedChanged(object sender,
EventArgs e)
    {
        if (checkBox1.Checked == true) { comboBox1.Enabled =
true; }
        if (checkBox1.Checked == false) { comboBox1.Enabled
= false; }
    }

    private void checkBox2_CheckedChanged(object sender,
EventArgs e)
    {
        if (checkBox2.Checked == true) {
dateTimePicker3.Enabled = true; dateTimePicker4.Enabled = true;
}
        if (checkBox2.Checked == false) {
dateTimePicker3.Enabled = false; dateTimePicker4.Enabled =
false; }
    }

    private void DailyCreate()
    {
        dlabel1.AutoSize = true;
        dlabel1.Font = new System.Drawing.Font("Candara",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        dlabel1.Location = new System.Drawing.Point(131,
51);
        dlabel1.Size = new System.Drawing.Size(116, 15);
        dlabel1.Text = "Повторять каждые:";
        dnumericUpDown1.Font = new System.Drawing.Font("Century Gothic",
8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        dnumericUpDown1.Location = new
System.Drawing.Point(248, 47);
        dnumericUpDown1.Size = new System.Drawing.Size(42,
21);
        dnumericUpDown1.Minimum = 1;
        dlabel2.AutoSize = true;
        dlabel2.Font = new System.Drawing.Font("Candara", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));

```

```

dlabel2.Location = new System.Drawing.Point(291, 50);
    dlabel2.Size = new System.Drawing.Size(24, 15);
    dlabel2.Text = "дн.";

    this.groupBox2.Controls.Add(dlabel1);
    this.groupBox2.Controls.Add(dlabel2);
    this.groupBox2.Controls.Add(dnumericUpDown1);

    Refresh();
}
private void DailyRemove()
{
    this.groupBox2.Controls.Remove(dlabel1);
    this.groupBox2.Controls.Remove(dlabel2);
    this.groupBox2.Controls.Remove(dnumericUpDown1);

    Refresh();
}
private void WeeklyCreate()
{
    wlabel1.AutoSize = true;
    wlabel1.Font = new System.Drawing.Font("Candara",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wlabel1.Location = new System.Drawing.Point(131,
51);

    wlabel1.Size = new System.Drawing.Size(116, 15);
    wlabel1.Text = "Повторять каждую";

    wnumericUpDown1.Font = new
System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wnumericUpDown1.Location = new
System.Drawing.Point(248, 47);
    wnumericUpDown1.Size = new System.Drawing.Size(42,
21);

    wnumericUpDown1.Minimum = 1;
wlabel2.AutoSize = true;
    wlabel2.Font = new System.Drawing.Font("Candara",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wlabel2.Location = new System.Drawing.Point(291,
50);

    wlabel2.Size = new System.Drawing.Size(24, 15);
    wlabel2.Text = "неделю в";
wcheckBox1.AutoSize = true;
    wcheckBox1.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox1.Location = new System.Drawing.Point(134,
74);

```

```

wcheckBox1.Size = new System.Drawing.Size(40, 18);
    wcheckBox1.Text = "ПН";
    wcheckBox1.UseVisualStyleBackColor = true;

    wcheckBox2.AutoSize = true;
    wcheckBox2.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox2.Location = new System.Drawing.Point(180,
74);

    wcheckBox2.Size = new System.Drawing.Size(38, 18);
    wcheckBox2.Text = "БТ";
    wcheckBox2.UseVisualStyleBackColor = true;

    wcheckBox3.AutoSize = true;
    wcheckBox3.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox3.Location = new System.Drawing.Point(226,
74);

    wcheckBox3.Size = new System.Drawing.Size(40, 18);
    wcheckBox3.Text = "Ср";
    wcheckBox3.UseVisualStyleBackColor = true;

    wcheckBox4.AutoSize = true;
    wcheckBox4.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox4.Location = new System.Drawing.Point(272,
74);

    wcheckBox4.Size = new System.Drawing.Size(38, 18);
    wcheckBox4.Text = "ЧТ";
    wcheckBox4.UseVisualStyleBackColor = true;
wcheckBox5.AutoSize = true;
    wcheckBox5.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox5.Location = new System.Drawing.Point(134,
97);

    wcheckBox5.Size = new System.Drawing.Size(39, 18);
    wcheckBox5.Text = "ПТ";
    wcheckBox5.UseVisualStyleBackColor = true;
wcheckBox6.AutoSize = true;
    wcheckBox6.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    wcheckBox6.Location = new System.Drawing.Point(180,
97);

    wcheckBox6.Size = new System.Drawing.Size(40, 18);
    wcheckBox6.Text = "Сб";
    wcheckBox6.UseVisualStyleBackColor = true;
wcheckBox7.AutoSize = true;

```

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

```

        wcheckBox7.Font = new System.Drawing.Font("Candara",
9F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        wcheckBox7.Location = new System.Drawing.Point(226,
97);

        wcheckBox7.Size = new System.Drawing.Size(38, 18);
        wcheckBox7.Text = "Bc";
        wcheckBox7.UseVisualStyleBackColor = true;

        this.groupBox2.Controls.Add(wlabel1);
        this.groupBox2.Controls.Add(wlabel2);
        this.groupBox2.Controls.Add(wnumericUpDown1);
        this.groupBox2.Controls.Add(wcheckBox1);
        this.groupBox2.Controls.Add(wcheckBox2);
        this.groupBox2.Controls.Add(wcheckBox3);
        this.groupBox2.Controls.Add(wcheckBox4);
        this.groupBox2.Controls.Add(wcheckBox5);
        this.groupBox2.Controls.Add(wcheckBox6);
        this.groupBox2.Controls.Add(wcheckBox7);

        Refresh();
    }

    private void WeeklyRemove()
    {
        this.groupBox2.Controls.Remove(wlabel1);
        this.groupBox2.Controls.Remove(wlabel2);
        this.groupBox2.Controls.Remove(wnumericUpDown1);
        this.groupBox2.Controls.Remove(wcheckBox1);
        this.groupBox2.Controls.Remove(wcheckBox2);
        this.groupBox2.Controls.Remove(wcheckBox3);
        this.groupBox2.Controls.Remove(wcheckBox4);
        this.groupBox2.Controls.Remove(wcheckBox5);
        this.groupBox2.Controls.Remove(wcheckBox6);
        this.groupBox2.Controls.Remove(wcheckBox7);

        Refresh();
    }

    private void MonthlyCreate()
    {
        mcheckedListBox1.CheckOnClick = true;
        mcheckedListBox1.Font = new
System.Drawing.Font("Candara", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        mcheckedListBox1.FormattingEnabled = true;
        if (m == false)
        {
            mcheckedListBox1.Items.AddRange(new object[] {
"Январь",
"Февраль",

```

```

        "Март",
        "Апрель",
        "Май",
        "Июнь",
        "Июль",
        "Август",
        "Сентябрь",
        "Октябрь",
        "Ноябрь",
        "Декабрь"});
        m = true;
    }
    mcheckedListBox1.Location = new
System.Drawing.Point(193, 41);
    mcheckedListBox1.ScrollAlwaysVisible = true;
    mcheckedListBox1.Size = new System.Drawing.Size(159,
36);

    mlabel1.AutoSize = true;
    mlabel1.Font = new System.Drawing.Font("Candara",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
    mlabel1.Location = new System.Drawing.Point(131,
51);

    mlabel1.Size = new System.Drawing.Size(56, 15);
    mlabel1.Text = "Месяцы:";
    mradioButton1.AutoSize = true;
    mradioButton1.Location = new System.Drawing.Point(3,
3);

    mradioButton1.Size = new System.Drawing.Size(50,
19);

    mradioButton1.TabStop = true;
    mradioButton1.Text = "Дни:";
    mradioButton1.UseVisualStyleBackColor = true;
    mradioButton1.CheckedChanged += new
System.EventHandler(this.mradioButton1_CheckedChanged);
    mradioButton1.Checked = true;
    mradioButton2.AutoSize = true;
    mradioButton2.Location = new System.Drawing.Point(3,
20);

    mradioButton2.Size = new System.Drawing.Size(36,
19);

    mradioButton2.TabStop = true;
    mradioButton2.Text = "В:";
    mradioButton2.UseVisualStyleBackColor = true;
    mradioButton2.CheckedChanged += new
System.EventHandler(this.mradioButton2_CheckedChanged);
    this.panel2.Visible = true;
    this.groupBox2.Controls.Add(mlabel1);
    this.groupBox2.Controls.Add(mcheckedListBox1);
    this.panel2.Controls.Add(mradioButton1);

```

```

this.panel2.Controls.Add(mradioButton2);

        DaysCreate();

        Refresh();
    }

    private void MonthlyRemove()
    {
        this.panel2.Visible = false;
        this.groupBox2.Controls.Remove(mlabel1);
        this.groupBox2.Controls.Remove(mcheckedListBox1);
        this.groupBox2.Controls.Remove(mradioButton1);
        this.groupBox2.Controls.Remove(mradioButton2);
        this.groupBox2.Controls.Remove(dcheckedListBox1);
        this.groupBox2.Controls.Remove(incheckedListBox1);
        this.groupBox2.Controls.Remove(incheckedListBox2);

        Refresh();
    }
    private void DaysCreate()
    {
        dcheckedListBox1.CheckOnClick = true;
        dcheckedListBox1.Font = new
System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        dcheckedListBox1.FormattingEnabled = true;
        if (d == false)
        {
            dcheckedListBox1.Items.AddRange(new object[] {
                "1",
                "2",
                "3",
                "4",
                "5",
                "6",
                "7",
                "8",
                "9",
                "10",
                "11",
                "12",
                "13",
                "14",
                "15",
                "16",
                "17",
                "18",
                "19",
                "20",
                "21",
            });
        }
    }

```

```

        "22",
        "23",
        "24",
        "25",
        "26",
        "27",
        "28",
        "29",
        "30",
        "31"});
        d = true;
    }
    "22",
    "23",
    "24",
    "25",
    "26",
    "27",
    "28",
    "29",
    "30",
    "31"});
    d = true;
}
dcheckedListBox1.Location = new
System.Drawing.Point(193, 83);
dcheckedListBox1.ScrollAlwaysVisible = true;
dcheckedListBox1.Size = new System.Drawing.Size(60,
36);

this.groupBox2.Controls.Add(dcheckedListBox1);

Refresh();
}
private void DaysRemove()
{
    this.groupBox2.Controls.Remove(dcheckedListBox1);

    Refresh();
}

private void InCreate()
{
    incheckedListBox1.CheckOnClick = true;
    incheckedListBox1.Font = new
System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
    incheckedListBox1.FormattingEnabled = true;
    if (i1 == false)
    {
        incheckedListBox1.Items.AddRange(new object[] {

```

```

        "1",
        "2",
        "3",
        "4",});
        i1 = true;
    }
    incheckedListBox1.Location = new
System.Drawing.Point(193, 83);
    incheckedListBox1.Size = new System.Drawing.Size(60,
36);

    incheckedListBox2.CheckOnClick = true;
    incheckedListBox2.Font = new
System.Drawing.Font("Candara", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
    incheckedListBox2.FormattingEnabled = true;
    if (i2 == false)
    {
        incheckedListBox2.Items.AddRange(new object[] {
        "ПН",
        "БТ",
        "Ср",
        "ЧТ",
        "Пт",
        "Сб",
        "Вс"});
        i2 = true;
    }
    incheckedListBox2.Location = new
System.Drawing.Point(259, 83);
    incheckedListBox2.ScrollAlwaysVisible = true;
    incheckedListBox2.Size = new System.Drawing.Size(93,
36);

    this.groupBox2.Controls.Add(incheckedListBox1);
    this.groupBox2.Controls.Add(incheckedListBox2);

    Refresh();
}
private void InRemove()
{
    this.groupBox2.Controls.Remove(incheckedListBox1);
    this.groupBox2.Controls.Remove(incheckedListBox2);

    Refresh();
}
private void button2_Click(object sender, EventArgs e)
{
    String[] btime = dateTimePicker1.Value.ToString().Split(new
char[] { ' ' });

```

```

String[] bdate = dateTimePicker2.Value.ToString().Split(new
char[] { ' ' });

    trigData.begin.time = btime[1];
    trigData.begin.date = bdate[0];

    if (radioButton1.Checked == true)
    {
        trigData.repeat = "Однократный";
        trigData.tableDescription = "В " +
trigData.begin.time + " " + trigData.begin.date;
    }

    if (radioButton2.Checked == true)
    {
        trigData.repeat = "Ежедневный";
        trigData.day = dnumericUpDown1.Value.ToString();
        trigData.tableDescription = "В " +
trigData.begin.time + " каждые " + trigData.day + " дней";
    }
    if (radioButton3.Checked == true)
    {
        trigData.repeat = "Еженедельный";
        trigData.week.week =
wnumericUpDown1.Value.ToString();
        if (wcheckBox1.Checked == true) {
trigData.week.days = "Пн "; }
        if (wcheckBox2.Checked == true) {
trigData.week.days += "Вт "; }
        if (wcheckBox3.Checked == true) {
trigData.week.days += "Ср "; }
        if (wcheckBox4.Checked == true) {
trigData.week.days += "Чт "; }
        if (wcheckBox5.Checked == true) {
trigData.week.days += "Пт "; }
        if (wcheckBox6.Checked == true) {
trigData.week.days += "Сб "; }
        if (wcheckBox7.Checked == true) {
trigData.week.days += "Вс "; }
        trigData.tableDescription = "В " +
trigData.begin.time + " по " + trigData.week.days + "каждую " +
trigData.week.week +
"-ю неделю, начиная с " + trigData.begin.date;
    }
    if (radioButton4.Checked == true)
    {
        trigData.repeat = "Ежемесячный";
        if
(mcheckedListBox1.CheckedItems.Contains("Январь")) {
trigData.month.month = "Янв "; }

```

```

if (mcheckedListBox1.CheckedItems.Contains("Февраль")) {
    trigData.month.month += "Фев "; }
    if
    (mcheckedListBox1.CheckedItems.Contains("Март")) {
    trigData.month.month += "Мар "; }
        if
        (mcheckedListBox1.CheckedItems.Contains("Апрель")) {
        trigData.month.month += "Апр "; }
            if
            (mcheckedListBox1.CheckedItems.Contains("Май")) {
            trigData.month.month += "Май "; }
                if
                (mcheckedListBox1.CheckedItems.Contains("Июнь")) {
                trigData.month.month += "Июн "; }
                    if
                    (mcheckedListBox1.CheckedItems.Contains("Июль")) {
                    trigData.month.month += "Июл "; }
                        if
                        (mcheckedListBox1.CheckedItems.Contains("Август")) {
                        trigData.month.month += "Авг "; }
                            if
                            (mcheckedListBox1.CheckedItems.Contains("Сентябрь")) {
                            trigData.month.month += "Сен "; }
                                if
                                (mcheckedListBox1.CheckedItems.Contains("Октябрь")) {
                                trigData.month.month += "Окт "; }
                                    if
                                    (mcheckedListBox1.CheckedItems.Contains("Ноябрь")) {
                                    trigData.month.month += "Ноя "; }
                                        if
                                        (mcheckedListBox1.CheckedItems.Contains("Декабрь")) {
                                        trigData.month.month += "Дек"; }

                                if (mradioButton1.Checked == true)
                                {
                                    foreach (object itemChecked in
dcheckedListBox1.CheckedItems)
                                    {
                                        trigData.month.days +=
itemChecked.ToString() + " ";
                                    }
                                    trigData.tableDescription = "В " +
trigData.begin.time + " в " + trigData.month.days + "день месяца
" +
trigData.month.month
+ " начиная с " + trigData.begin.date;
                                }
                                if (mradioButton2.Checked == true)
                                {
                                    foreach (object itemChecked in incheckedListBox1.CheckedItems)
                                    {

```

```

trigData.month.weeks += itemChecked.ToString() + " ";
    }

    foreach (object itemChecked in
incheckedListBox2.CheckedItems)
    {
        trigData.month.wdays +=
itemChecked.ToString() + " ";
    }
    trigData.tableDescription = "В " +
trigData.begin.time + " в " + trigData.month.weeks +
        trigData.month.wdays
+ " месяца " + trigData.month.month + " начиная с " +
trigData.begin.date;
    }
}

if (checkBox1.Checked == true) { trigData.delay =
comboBox1.Text; }
if (checkBox2.Checked == true)
{
    String[] dtime =
dateTimePicker4.Value.ToString().Split(new char[] { ' ' });
    String[] ddate =
dateTimePicker3.Value.ToString().Split(new char[] { ' ' });

    trigData.doing.time = dtime[1];
    trigData.doing.date = ddate[0];

    trigData.tableDescription += ",Срок истечения
действия триггера: " + trigData.doing.date + " " +
trigData.doing.time;
}
if (checkBox3.Checked == true) { trigData.enable =
"Включен"; }
if (checkBox3.Checked == false) { trigData.enable =
"Отключен"; }
System.Windows.Forms.ListViewItem item = new ListViewItem(new
string[] { trigData.repeat, trigData.tableDescription,
trigData.enable }, 0);
if (index != -1)
{
    data.T.RemoveAt(index);
    trig.listView.Items.RemoveAt(index);
    data.T.Insert(index, trigData);
    trig.listView.Items.Insert(index, item);
}
else
{
    data.T.Add(trigData);
    trig.listView.Items.Add(item);
}

```

```

this.Close();
}

private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}

private void InitData(int index)
{
    DateTime dt1 =
Convert.ToDateTime(data.T[index].begin.time + " " +
data.T[index].begin.date);
    dateTimePicker1.Value = dt1;
    dateTimePicker2.Value = dt1;

    if (data.T[index].delay != null)
    {
        checkBox1.Checked = true;
        comboBox1.Enabled = true;
        comboBox1.Text = data.T[index].delay;
    }

    if (data.T[index].doing.date != null &&
data.T[index].doing.time != null)
    {
        checkBox2.Checked = true;
        DateTime dt2 =
Convert.ToDateTime(data.T[index].doing.time + " " +
data.T[index].doing.date);
        dateTimePicker3.Enabled = true;
        dateTimePicker4.Enabled = true;
        dateTimePicker3.Value = dt2;
        dateTimePicker4.Value = dt2;
    }
    if (data.T[index].enable == "Включен") checkBox3.Checked = true;
    if (data.T[index].enable == "Отключен")
checkBox3.Checked = false;

    if (data.T[index].repeat == "Однократный")
    {
        radioButton1.Checked = true;
    }
    if (data.T[index].repeat == "Ежедневный")
    {
        radioButton2.Checked = true;
        dnumericUpDown1.Value =
Convert.ToDecimal(data.T[index].day);
    }
    if (data.T[index].repeat == "Еженедельный")
    {
        radioButton3.Checked = true;
    }
}

```

```

wnumericUpDown1.Value =
Convert.ToDecimal(data.T[index].week.week);

        if (data.T[index].week.days.Contains("Пн"))
wcheckBox1.Checked = true;
        if (data.T[index].week.days.Contains("Вт"))
wcheckBox2.Checked = true;
        if (data.T[index].week.days.Contains("Ср"))
wcheckBox3.Checked = true;
        if (data.T[index].week.days.Contains("Чт"))
wcheckBox4.Checked = true;
        if (data.T[index].week.days.Contains("Пт"))
wcheckBox5.Checked = true;
        if (data.T[index].week.days.Contains("Сб"))
wcheckBox6.Checked = true;
        if (data.T[index].week.days.Contains("Вс"))
wcheckBox7.Checked = true;
    }
    if (data.T[index].repeat == "Ежемесячный")
    {
        radioButton4.Checked = true;

        if (data.T[index].month.month.Contains("Янв"))
mcheckedListBox1.SetItemChecked(0, true);
        if (data.T[index].month.month.Contains("Фев"))
mcheckedListBox1.SetItemChecked(1, true);
        if (data.T[index].month.month.Contains("Мар"))
mcheckedListBox1.SetItemChecked(2, true);
        if (data.T[index].month.month.Contains("Апр"))
mcheckedListBox1.SetItemChecked(3, true);
        if (data.T[index].month.month.Contains("Май"))
mcheckedListBox1.SetItemChecked(4, true);
        if (data.T[index].month.month.Contains("Июн"))
mcheckedListBox1.SetItemChecked(5, true);
        if (data.T[index].month.month.Contains("Июл"))
mcheckedListBox1.SetItemChecked(6, true);
        if (data.T[index].month.month.Contains("Авг"))
mcheckedListBox1.SetItemChecked(7, true);
        if (data.T[index].month.month.Contains("Сен"))
mcheckedListBox1.SetItemChecked(8, true);
        if (data.T[index].month.month.Contains("Окт"))
mcheckedListBox1.SetItemChecked(9, true);
        if (data.T[index].month.month.Contains("Ноя"))
mcheckedListBox1.SetItemChecked(10, true);
        if (data.T[index].month.month.Contains("Дек"))
mcheckedListBox1.SetItemChecked(11, true);
        if (data.T[index].month.days != null)
        {
            mradioButton1.Checked = true;
            string[] temp = data.T[index].month.days.Split(new char[] { ' ' }
);

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

73

```

for (int i = 0; i < temp.Count(); i++)
{
    foreach (object item in
dcheckedListBox1.Items)
    {
        if (item.ToString() == temp[i])
        {
dcheckedListBox1.SetItemChecked(dcheckedListBox1.Items.IndexOf(i
tem), true);
        }
    }
}

if (data.T[index].month.weeks != null ||
data.T[index].month.wdays != null)
{
    mradioButton2.Checked = true;

    if (data.T[index].month.weeks.Contains("1"))
incheckedListBox1.SetItemChecked(0, true);
    if (data.T[index].month.weeks.Contains("2"))
incheckedListBox1.SetItemChecked(1, true);
    if (data.T[index].month.weeks.Contains("3"))
incheckedListBox1.SetItemChecked(2, true);
    if (data.T[index].month.weeks.Contains("4"))
incheckedListBox1.SetItemChecked(3, true);

if (data.T[index].month.wdays.Contains("Пн"))
incheckedListBox2.SetItemChecked(0, true);
    if
(data.T[index].month.wdays.Contains("Вт"))
incheckedListBox2.SetItemChecked(1, true);
    if
(data.T[index].month.wdays.Contains("Ср"))
incheckedListBox2.SetItemChecked(2, true);
    if
(data.T[index].month.wdays.Contains("Чт"))
incheckedListBox2.SetItemChecked(3, true);
    if
(data.T[index].month.wdays.Contains("Пт"))
incheckedListBox2.SetItemChecked(4, true);
    if
(data.T[index].month.wdays.Contains("Сб"))
incheckedListBox2.SetItemChecked(5, true);
    if
(data.T[index].month.wdays.Contains("Вс"))
incheckedListBox2.SetItemChecked(6, true);
}
}

```

```

Refresh();
    }
}
}

```

Приложение A.5

Файл «ActionForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

namespace Client
{
    public partial class ActionForm : MaterialForm
    {
        Actions act;
        Data.Data data;
        Data.Data.ActionData actData;
        int index;
        public ActionForm(Actions act, Data.Data data, int
index)
        {
            this.act = act;
            this.data = data;
            this.index = index;
            InitializeComponent();
            var materialSkinManager =
MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme =
MaterialSkinManager.Themes.LIGHT;
            materialSkinManager.ColorScheme = new
ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Blue50,
Accent.Teal700, TextShade.WHITE);
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            if (index != -1)
            {
                InitData(index);
            }
        }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

75

```

act.createbtn.Enabled = false;
    act.changebtn.Enabled = false;
    act.deletebtn.Enabled = false;
    button2.Enabled = false;
}

private void Form5_FormClosing(object sender,
FormClosingEventArgs e)
{
    act.n = true;
    act.m = true;
    if (index == -1)
    {
        act.createbtn.Enabled = true;
        act.changebtn.Enabled = false;
        act.deletebtn.Enabled = false;
    }
    else
    {
        act.createbtn.Enabled = false;
        act.changebtn.Enabled = true;
        act.deletebtn.Enabled = true;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
    {
        textBox1.Text = openFileDialog1.FileName;
    }
    if (textBox1.Text != "")
        button2.Enabled = true;
}

private void button2_Click(object sender, EventArgs e)
{
    actData.path = textBox1.Text;
    if (checkBox1.Checked == true) { actData.enable =
"Включена"; }
    if (checkBox1.Checked == false) { actData.enable =
"Отключена"; }

    System.Windows.Forms.ListViewItem item = new
ListViewItem(new string[] { "Запуск программы или открытие
файла" , actData.path }, 0);
    if (index != -1)
    {
        data.A.RemoveAt(index);
        act.listView.Items.RemoveAt(index);
        data.A.Insert(index, actData);
    }
}

```

```

act.listView.Items.Insert(index, item);
    }
    else
    {
        data.A.Add(actData);
        act.listView.Items.Add(item);
    }
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Close();
}

private void InitData(int index)
{
    textBox1.Text = data.A[index].path;
    if (data.A[index].enable == "Включена")
    {
        checkBox1.Checked = true;
    }
    if (data.A[index].enable == "Отключена")
    {
        checkBox1.Checked = false;
    }
}
}
}

```

Приложение А.6

Файл «ServerForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

using System.Net;
using System.Net.Sockets;

namespace Client
{
    public partial class ServerForm : MaterialForm

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

77

```

{
    MainForm form;
    TcpClient clSock;
    ClientSocket Socket;
    public ServerForm(MainForm form, TcpClient clSock,
ClientSocket Socket)
    {
        this.form = form;
        this.clSock = clSock;
        this.Socket = Socket;
        InitializeComponent();
        var materialSkinManager =
MaterialSkinManager.Instance;
        materialSkinManager.AddFormToManage(this);
        materialSkinManager.Theme =
MaterialSkinManager.Themes.LIGHT;
        materialSkinManager.ColorScheme = new
ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Blue50,
Accent.Teal700, TextShade.WHITE);
    }

    private void Form3_FormClosing(object sender,
FormClosingEventArgs e)
    {
        form.label3.Enabled = true;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        String ip = textBox1.Text + "." + textBox2.Text +
"." + textBox3.Text + "." + textBox4.Text;
        String port = textBox5.Text;
        Socket.CreateClientSocket(ip, port, form, clSock);
        this.Close();
    }
}
}

```

Приложение А.7

Файл «PasswordForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;

```

					ПГУ 2.090301.26.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		78

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MaterialSkin;
using MaterialSkin.Controls;

namespace Client
{
    public partial class PasswordForm : MaterialForm
    {
        MainForm form;
        ClientSocket Socket;
        int taskindex = 0;
        String action = "";
        public PasswordForm(MainForm form, ClientSocket Socket,
int taskindex, String action)
        {
            this.form = form;
            this.Socket = Socket;
            this.taskindex = taskindex;
            this.action = action;
            InitializeComponent();
            var materialSkinManager =
MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme =
MaterialSkinManager.Themes.LIGHT;
            materialSkinManager.ColorScheme = new
ColorScheme(Primary.Teal700, Primary.Teal900, Primary.Blue50,
Accent.Teal700, TextShade.WHITE);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (form.ReceivedTasks.ElementAt(taskindex).password
== textBox1.Text)
            {
                this.Close();
                if (action == "edit")
                {
                    TaskForm form2 = new TaskForm(form, Socket,
taskindex);

                    form2.Show();
                }
            }
        }
    }
}

```

```

if (action == "delete")
{
    Socket.DeleteTask(taskindex);
    Socket.ReceiveTasks(form.ReceivedTasks,
taskindex);
    form.SetTasksToListView();
}
else
{
    MessageBox.Show("Неверный пароль", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}
}

```

Приложение А.8

Файл «General.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Client
{
    public class General
    {
        Font font = new Font("Candara", 9.75f);
        Label label5 = new Label();
        Label label6 = new Label();
        public Label label7 = new Label();
        Label label8 = new Label();
        public TextBox textBox1 = new TextBox();
        public RichTextBox richTextBox1 = new RichTextBox();

        public void GenInit(TaskForm form, int taskindex)
        {
            label5.Text = "Имя задачи";
            label5.Font = font;
            label5.BackColor = Color.Transparent;
            label5.Size = new Size(70, 20);
            label5.Location = new Point(158, 66);

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

80

```

label6.Text = "Автор";
label6.Font = font;
label6.BackColor = Color.Transparent;
label6.Size = new Size(70, 20);
label6.Location = new Point(158, 92);

label7.Text = Environment.UserName;
label7.Enabled = false;
label7.Font = font;
label7.BackColor = Color.Transparent;
label7.Size = new Size(300, 20);
label7.Location = new Point(247, 92);

label8.Text = "Описание";
label8.Font = font;
label8.BackColor = Color.Transparent;
label8.Size = new Size(70, 20);
label8.Location = new Point(158, 118);

textBox1.Size = new Size(235, 20);
textBox1.Location = new Point(250, 64);

richTextBox1.Size = new Size(235, 84);
richTextBox1.Location = new Point(250, 118);

form.panel1.Controls.Add(label5);
form.panel1.Controls.Add(label6);
form.panel1.Controls.Add(label7);
form.panel1.Controls.Add(label8);
form.panel1.Controls.Add(textBox1);
form.panel1.Controls.Add(richTextBox1);
form.materialLabel1.Text = "Общие";
form.label5.Text = "Общая информация о задаче";
form.label1.ForeColor = Color.Orange;
form.label2.ForeColor = Color.Black;

if (taskindex != -1)
{
    textBox1.Text = form.data.genData.name;
    richTextBox1.Text =
form.data.genData.description;
}

public void GenRemove(TaskForm form)
{
    form.panel1.Controls.Remove(label5);
    form.panel1.Controls.Remove(label6);
    form.panel1.Controls.Remove(label7);
    form.panel1.Controls.Remove(label8);
}

```

```

form.panell1.Controls.Remove(textBox1);
        form.panell1.Controls.Remove(richTextBox1);

        form.Refresh();
    }
}

```

Приложение А.9

Файл «Triggers.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Client
{
    public class Triggers
    {
        Font font = new Font("Candara", 9.75f);
        public ListView listView = new ListView();
        ColumnHeader columnHeader1 = new ColumnHeader();
        ColumnHeader columnHeader2 = new ColumnHeader();
        ColumnHeader columnHeader3 = new ColumnHeader();
        public Button createbtn = new Button();
        public Button changebtn = new Button();
        public Button deletebtn = new Button();
        Data.Data data;
        public bool n = true, m = true;
        int index = -1;

        public void TrigInit(TaskForm form, Data.Data data, int
taskindex)
        {
            this.data = data;
            listView.BackColor =
System.Drawing.SystemColors.ControlLight;
            listView.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
                columnHeader1,
                columnHeader2,
                columnHeader3});
        }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

82

```

listView.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Nonclickable;
    listView.FullRowSelect = true;
    listView.GridLines = true;
    listView.Location = new System.Drawing.Point(145,
52);

    listView.Size = new System.Drawing.Size(372, 111);
    listView.View = System.Windows.Forms.View.Details;
    listView.MultiSelect = false;
    listView.HideSelection = false;
    listView.ItemSelectionChanged += new
ListViewItemSelectionChangedEventHandler(listView_itemSelectionC
hanged);

    columnHeader1.Text = "Триггер";
    columnHeader1.Width = 95;

    columnHeader2.Text = "Описание";
    columnHeader2.Width = 177;

    columnHeader3.Text = "Состояние";
    columnHeader3.Width = 89;

    createbtn.BackColor =
System.Drawing.SystemColors.ButtonFace;
    createbtn.Enabled = true;
    createbtn.FlatAppearance.BorderSize = 0;
    createbtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(152)))),
((int)(((byte)(208)))), ((int)(((byte)(195)))));
    createbtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(139)))),
((int)(((byte)(191)))), ((int)(((byte)(179)))));
    createbtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    createbtn.Font = font;
    createbtn.Location = new System.Drawing.Point(145,
169);

    createbtn.Size = new System.Drawing.Size(85, 21);
    createbtn.Text = "Создать";
    createbtn.UseVisualStyleBackColor = false;
    createbtn.Click += new
System.EventHandler(createbtn_Click);

    changebtn.BackColor = System.Drawing.SystemColors.ButtonFace;
    changebtn.Enabled = false;
    changebtn.FlatAppearance.BorderSize = 0;
    changebtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(152)))),
((int)(((byte)(208)))), ((int)(((byte)(195)))));

```

```

changebtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(139)))),
((int)((byte)(191)))), ((int)((byte)(179))));
    changebtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    changebtn.Font = font;
    changebtn.Location = new System.Drawing.Point(241,
169);

    changebtn.Size = new System.Drawing.Size(85, 21);
    changebtn.Text = "Изменить";
    changebtn.UseVisualStyleBackColor = false;
    changebtn.Click += new
System.EventHandler(changebtn_Click);

    deletebtn.BackColor =
System.Drawing.SystemColors.ButtonFace;
    deletebtn.Enabled = false;
    deletebtn.FlatAppearance.BorderSize = 0;
    deletebtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(152)))),
((int)((byte)(208)))), ((int)((byte)(195))));
    deletebtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(139)))),
((int)((byte)(191)))), ((int)((byte)(179))));
    deletebtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    deletebtn.Font = font;
    deletebtn.Location = new System.Drawing.Point(337,
169);

    deletebtn.Size = new System.Drawing.Size(85, 21);
    deletebtn.Text = "Удалить";
    deletebtn.UseVisualStyleBackColor = false;
    deletebtn.Click += new
System.EventHandler(deletebtn_Click);

    form.panell1.Controls.Add(listView);
    form.panell1.Controls.Add(createbtn);
    form.panell1.Controls.Add(changebtn);
    form.panell1.Controls.Add(deletebtn);
    form.materialLabel1.Text = "Триггеры";
    form.label5.Text = "Условия запуска задачи (не
обязательны) ";
    form.label1.ForeColor = Color.Black;
    form.label2.ForeColor = Color.Orange;
    form.label3.ForeColor = Color.Black;

    if (taskindex != -1)
    {
        for (int i = 0; i < data.T.Count; i++)
        listView.Items.Add(new ListViewItem(new string[] {

```

```

data.T.ElementAt(i).repeat,
                        data.T.ElementAt(i).tableDescription,
data.T.ElementAt(i).enable }, 0));
        }
    }
}
public void TrigRemove(TaskForm form)
{
    listView.Columns.Remove(columnHeader1);
    listView.Columns.Remove(columnHeader2);
    listView.Columns.Remove(columnHeader3);
    form.panell1.Controls.Remove(listView);
    form.panell1.Controls.Remove(createbtn);
    form.panell1.Controls.Remove(changebtn);
    form.panell1.Controls.Remove(deletebtn);

    form.Refresh();
}

private void createbtn_Click(object sender, EventArgs e)
{
    if (n == true)
    {
        TriggerForm form4 = new TriggerForm(this, data,
index);

        form4.Show();
        n = false;
    }
}

private void changebtn_Click(object sender, EventArgs e)
{
    if (m == true)
    {
        TriggerForm form4 = new TriggerForm(this, data,
index);

        form4.Show();
        m = false;
    }
}

private void deletebtn_Click(object sender, EventArgs e)
{
    index = listView.SelectedIndices[0];
    data.T.RemoveAt(index);
    listView.Items.RemoveAt(index);
}

private void listView_itemSelectionChanged(object sender,
EventArgs e)
{
    if (listView.SelectedItems.Count != 0)
    {

```

```

        index = listView.SelectedIndices[0];
        changebtn.Enabled = true;
        deletebtn.Enabled = true;
        createbtn.Enabled = false;
    }
    else
    {
        index = -1;
        changebtn.Enabled = false;
        deletebtn.Enabled = false;
        createbtn.Enabled = true;
    }
}
}
}

```

Приложение А.10

Файл «Actions.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Client
{
    public class Actions
    {
        Font font = new Font("Candara", 9.75f);
        public ListView listView = new ListView();
        ColumnHeader columnHeader1 = new ColumnHeader();
        ColumnHeader columnHeader2 = new ColumnHeader();
        public Button createbtn = new Button();
        public Button changebtn = new Button();
        public Button deletebtn = new Button();
        Data.Data data;
        public bool n = true, m = true;
        int index = -1;

        public void ActInit(TaskForm form, Data.Data data, int
taskindex)
        {
            this.data = data;
            listView.BackColor =
System.Drawing.SystemColors.ControlLight;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

86

```

listView.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
    columnHeader1,
    columnHeader2});
listView.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Nonclickable;
listView.FullRowSelect = true;
listView.GridLines = true;
listView.Location = new System.Drawing.Point(145,
52);

listView.Size = new System.Drawing.Size(372, 111);
listView.View = System.Windows.Forms.View.Details;
listView.ItemSelectionChanged += new
ListViewItemSelectionChangedEventHandler(listView_itemSelectionC
hanged);

columnHeader1.Text = "Действие";
columnHeader1.Width = 135;

columnHeader2.Text = "Подробности";
columnHeader2.Width = 237;

createbtn.BackColor =
System.Drawing.SystemColors.ButtonFace;
createbtn.Enabled = true;
createbtn.FlatAppearance.BorderSize = 0;
createbtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(152)))),
((int)(((byte)(208)))), ((int)(((byte)(195)))));
createbtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(139)))),
((int)(((byte)(191)))), ((int)(((byte)(179)))));
createbtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
createbtn.Font = font;
createbtn.Location = new System.Drawing.Point(145,
169);

createbtn.Size = new System.Drawing.Size(85, 21);
createbtn.Text = "Создать";
createbtn.UseVisualStyleBackColor = false;
createbtn.Click += new
System.EventHandler(createbtn_Click);

changebtn.BackColor = System.Drawing.SystemColors.ButtonFace;
if (listView.Items.Count != 0)
{ changebtn.Enabled = true; }
else { changebtn.Enabled = false; }
changebtn.FlatAppearance.BorderSize = 0;
changebtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(152)))),
((int)(((byte)(208)))), ((int)(((byte)(195)))));

```

```

changebtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(139)))),
((int)((byte)(191)))), ((int)((byte)(179))));
    changebtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    changebtn.Font = font;
    changebtn.Location = new System.Drawing.Point(241,
169);

    changebtn.Size = new System.Drawing.Size(85, 21);
    changebtn.Text = "Изменить";
    changebtn.UseVisualStyleBackColor = false;
    changebtn.Click += new
System.EventHandler(changebtn_Click);

    deletebtn.BackColor =
System.Drawing.SystemColors.ButtonFace;
    if (listView.Items.Count != 0)
    { deletebtn.Enabled = true; }
    else { deletebtn.Enabled = false; }
    deletebtn.FlatAppearance.BorderSize = 0;
    deletebtn.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(152)))),
((int)((byte)(208)))), ((int)((byte)(195))));
    deletebtn.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)((byte)(139)))),
((int)((byte)(191)))), ((int)((byte)(179))));
    deletebtn.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    deletebtn.Font = font;
    deletebtn.Location = new System.Drawing.Point(337,
169);

    deletebtn.Size = new System.Drawing.Size(85, 21);
    deletebtn.Text = "Удалить";
    deletebtn.UseVisualStyleBackColor = false;
    deletebtn.Click += new
System.EventHandler(deletebtn_Click);

    form.panell1.Controls.Add(listView);
    form.panell1.Controls.Add(createbtn);
    form.panell1.Controls.Add(changebtn);
    form.panell1.Controls.Add(deletebtn);
    form.button2.Text = "Далее";
    form.materialLabel1.Text = "Действия";
    form.label5.Text = "Действия, которые будут
выполняться данной задачей";
    form.label2.ForeColor = Color.Black;
    form.label3.ForeColor = Color.Orange;
    form.label4.ForeColor = Color.Black;

    if (taskindex != -1)
    {

```

```

for (int i = 0; i < data.A.Count; i++)
{
    listView.Items.Add(new ListViewItem(new
string[] { "Запуск программы или открытие файла",
data.A.ElementAt(i).path }, 0));
}
}
}
public void ActRemove(TaskForm form)
{
    listView.Columns.Remove(columnHeader1);
    listView.Columns.Remove(columnHeader2);
    form.panell1.Controls.Remove(listView);
    form.panell1.Controls.Remove(createbtn);
    form.panell1.Controls.Remove(changebtn);
    form.panell1.Controls.Remove(deletebtn);

    form.Refresh();
}

private void createbtn_Click(object sender, EventArgs e)
{
    if (n == true)
    {
        ActionForm form5 = new
ActionForm(this,data,index);
        form5.Show();
        n = false;
    }
}

private void changebtn_Click(object sender, EventArgs e)
{
    if (m == true)
    {
        ActionForm form5 = new ActionForm(this,
data,index);
        form5.Show();
        m = false;
    }
}

private void deletebtn_Click(object sender, EventArgs e)
{
    index = listView.SelectedIndices[0];
    data.A.RemoveAt(index);
    listView.Items.RemoveAt(index);
}

private void listView_itemSelectionChanged(object sender,
EventArgs e)
{

```

```

if (listView.SelectedItems.Count != 0)
{
    index = listView.SelectedIndices[0];
    changebtn.Enabled = true;
    deletebtn.Enabled = true;
    createbtn.Enabled = false;
}
else
{
    index = -1;
    changebtn.Enabled = false;
    deletebtn.Enabled = false;
    createbtn.Enabled = true;
}
}
}
}

```

Приложение А.11

Файл «Safety.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Client
{
    public class Safety
    {
        Font font = new Font("Candara", 9.75f);
        Label label1 = new Label();
        Label label2 = new Label();
        Label label3 = new Label();
        public Label label4 = new Label();
        public TextBox textBox1 = new TextBox();
        public TextBox textBox2 = new TextBox();
        public bool passval = false;

        private void textBox1_lostFocus(object sender, EventArgs
e)
        {
            CheckPassword();
        }
    }
}

```

```

private void CheckPassword()
{
    if (textBox1.Text != "" &&
!textBox1.Text.ToString().All(c => char.IsLetterOrDigit(c)))
        { label4.Text = "Пароль содержит недопустимые
символы"; passval = false; }
        else if (textBox1.Text != "" && textBox1.Text.Length
< 4)
            { label4.Text = "Пароль слишком короткий"; passval =
false; }
            else if (textBox1.Text != "")
                { label4.Text = ""; passval = true;
textBox2.ReadOnly = false; }
}

private void textBox1_gotFocus(object sender, EventArgs
e)
{
    textBox2.ReadOnly = true;
}

public void SafeInit(TaskForm form, int taskindex)
{
    label4.Font = font;
    label4.Text = "";
    label4.ForeColor = Color.Red;
    label4.BackColor = Color.Transparent;
    label4.Size = new Size(350, 30);
    label4.Location = new Point(158, 155);

    label3.Text = "Допустимая длина пароля: 4-16
символов, допустимые символы: А-я, А-з, 0-9";
    label3.Font = font;
    label3.BackColor = Color.Transparent;
    label3.Size = new Size(350, 30);
    label3.Location = new Point(158, 52);

    label1.Text = "Пароль";
    label1.Font = font;
    label1.BackColor = Color.Transparent;
    label1.Size = new Size(100, 20);
    label1.Location = new Point(158, 90);

    label2.Text = "Повторный ввод" + "\n" + "пароля";
    label2.Font = font;
    label2.BackColor = Color.Transparent;
    label2.Size = new Size(100, 40);
    label2.Location = new Point(158, 116);
    textBox1.Size = new Size(185, 20);
    textBox1.Location = new Point(270, 88);
    textBox1.PasswordChar = '•';

```

```

textBox1.Text = "";
        textBox1.Font = font;
        textBox1.MaxLength = 16;
        textBox1.GotFocus += new
System.EventHandler(this.textBox1_gotFocus);
        textBox1.LostFocus += new
System.EventHandler(this.textBox1_lostFocus);

        textBox2.Size = new Size(185, 20);
        textBox2.Location = new Point(270, 121);
        textBox2.PasswordChar = '•';
        textBox2.Text = "";
        textBox2.Font = font;
        textBox2.MaxLength = 16;
        textBox2.ReadOnly = true;

        form.panell1.Controls.Add(label1);
        form.panell1.Controls.Add(label2);
        form.panell1.Controls.Add(label3);
        form.panell1.Controls.Add(label4);
        form.panell1.Controls.Add(textBox1);
        form.panell1.Controls.Add(textBox2);
        form.button2.Text = "Готово";
        form.button2.Enabled = true;
        form.materialLabel1.Text = "Безопасность";
        form.label5.Text = "Защита задачи паролем от
изменения или удаления (не обязательна)";
        form.label4.ForeColor = Color.Orange;
        form.label3.ForeColor = Color.Black;

        if (taskindex != -1)
        {
            textBox1.Text = form.data.password;
            textBox2.Text = form.data.password;
        }
    }

    public void SafeRemove(TaskForm form)
    {
        form.panell1.Controls.Remove(label1);
        form.panell1.Controls.Remove(label2);
        form.panell1.Controls.Remove(label3);
        form.panell1.Controls.Remove(label4);
        form.panell1.Controls.Remove(textBox1);
        form.panell1.Controls.Remove(textBox2);

        form.Refresh();
    }
}

```

Приложение А.12

Файл «ClientSocket.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Runtime.Serialization.Formatters.Binary;

using System.IO;
using System.Net;
using System.Net.Sockets;

namespace Client
{
    public class ClientSocket
    {
        public TcpClient clientSocket;
        NetworkStream writerStream;
        NetworkStream readerStream;
        IPAddress ip;
        int port;
        MainForm form;

        public void CreateClientSocket(String ip, String port, MainForm
form, TcpClient clSock)
        {
            this.ip = IPAddress.Parse(ip);
            this.port = Convert.ToInt32(port);
            this.form = form;
            this.clientSocket = clSock;
            try
            {
                clientSocket.Connect(this.ip, this.port);
                form.label2.Text = ip;
                writerStream = clientSocket.GetStream();
                readerStream = clientSocket.GetStream();
            }
            catch (SocketException)
            {
                MessageBox.Show("Не удалось подключиться к
серверу", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        public void CreateMessage(Data.Data data)
        {
            BinaryFormatter formatter = new BinaryFormatter();
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

93

```

using (FileStream fs = new FileStream("1task.dat",
    FileMode.Create))
    {
        formatter.Serialize(fs, data);
        fs.Close();
    }
}

public void CreateTasks(List<Data.Data> ReceivedTasks)
{
    BinaryFormatter formatter = new BinaryFormatter();
    using (FileStream fs = new FileStream("tasks.dat",
        FileMode.Open))
    {
        List<Data.Data> received_data =
        (List<Data.Data>)formatter.Deserialize(fs);
        ReceivedTasks.Clear();
        for (int i = 0; i < received_data.Count; i++)
        {
            ReceivedTasks.Add(received_data.ElementAt(i));
        }

        fs.Close();
    }
    System.IO.File.Delete(@"tasks.dat");
}

public void DeleteTask(int taskindex)
{
    String message;
    Byte[] msg;
    message = "del:" + Convert.ToString(taskindex) +
    ":";

    msg = System.Text.Encoding.ASCII.GetBytes(message);
    writerStream.Write(msg, 0, msg.Length);
}

public void SendMessage(int taskindex)
{
    String message;
    Byte[] msg;
    if (taskindex != -1)
        message = "edit:" + Convert.ToString(taskindex)
    + ":";
    else
        message = "receive_";
    msg =
    System.Text.Encoding.ASCII.GetBytes(message);
    writerStream.Write(msg, 0, msg.Length);
}

```

```

BinaryFormatter formatter = new BinaryFormatter();
    byte[] buf = new byte[1024];
    int count;
    FileStream fs = new FileStream("1task.dat",
FileMode.Open);
    BinaryReader br = new BinaryReader(fs);
    long k = fs.Length;
    formatter.Serialize(writerStream, k.ToString());
    while ((count = br.Read(buf, 0, 1024)) > 0)
    {
        formatter.Serialize(writerStream, buf);
    }
    br.Close();
    fs.Close();
    System.IO.File.Delete(@"1task.dat");
}

public void ReceiveTasks(List<Data.Data>
ReceivedTasks,int taskindex)
{
    int count;
    String message;
    Byte[] msg;
    if (taskindex == -1)
    {
        message = "sendtask";
        msg =
System.Text.Encoding.ASCII.GetBytes(message);
        readerStream.Write(msg, 0, msg.Length);
    }
    BinaryFormatter formatter = new BinaryFormatter();
    FileStream fs = new FileStream("tasks.dat",
FileMode.Create);
    BinaryWriter bw = new BinaryWriter(fs);
    count =
int.Parse(formatter.Deserialize(readerStream).ToString());
    int i = 0;
    for (; i < count; i += 1024)
    {
        byte[] buf =
(byte[])(formatter.Deserialize(readerStream));
        bw.Write(buf);
    }
    bw.Close();
    fs.Close();
    CreateTasks(ReceivedTasks);
}
}
}

```

Приложение В

Листинги программы сервера

Приложение В.1

Файл «Program.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceProcess;
using System.Text;

namespace Server
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        static void Main()
        {
            ServiceBase[] ServicesToRun;
            ServicesToRun = new ServiceBase[]
            {
                new TaskManager()
            };
            ServiceBase.Run(ServicesToRun);
        }
    }
}
```

Приложение В.2

Файл «TaskManager.cs»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.ServiceProcess;
using System.Text;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.IO;
```

```

namespace Server
{
    public partial class TaskManager : ServiceBase
    {
        const int port = 770;
        static TcpListener listener;
        static Client clientSocket;
        FileStream fs = new FileStream("C:/log.txt",
        FileMode.OpenOrCreate);

        protected override void OnStart(string[] args)
        {
            try
            {
                TaskHandler taskHandler = new
TaskHandler(clientSocket, fs);
                taskHandler.LoadTasks();
                listener = new TcpListener(IPAddress.Any, port);
                listener.Start();
                String log = "Служба TaskManager успешно
запущена.";
                byte[] array =
System.Text.Encoding.Default.GetBytes(log);
                fs.Write(array, 0, array.Length);
                while (true)
                {
                    TcpClient client =
listener.AcceptTcpClient();
                    clientSocket = new Client(client,
taskHandler, fs);
                    Thread clientThread = new Thread(new
ThreadStart(clientSocket.Process));
                    clientThread.Start();
                }
            }
            catch (Exception ex)
            {
                String exc = Convert.ToString(ex);
                byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
                fs.Write(array, 0, array.Length);
            }
        }

        protected override void OnStop()
        {
            listener.Stop();
            fs.Close();
        }
    }
}

```

Приложение В.3

Файл «Client.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Serialization.Formatters.Binary;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace Server
{
    class Client
    {
        public TcpClient client;
        public TaskHandler taskHandler;
        FileStream fs;
        public Client(TcpClient tcpClient, TaskHandler
taskhandler, FileStream fs)
        {
            this.client = tcpClient;
            this.taskHandler = taskhandler;
            this.fs = fs;
            taskHandler.client = client;
        }
        int index = -1;

        public void Process()
        {
            NetworkStream readerStream = null;
            Byte[] bytes = new Byte[8];
            while (true)
            {
                try
                {
                    readerStream = null;
                    readerStream = client.GetStream();
                    readerStream.Read(bytes, 0, bytes.Length);
                    String msg =
System.Text.Encoding.ASCII.GetString(bytes);
                    if (msg == "receive_")
                    {
                        taskHandler.ReceiveTask(readerStream);
                        taskHandler.AddTask(index);
                        taskHandler.SendTasks();
                    }
                }
            }
        }
    }
}
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

98

```

if (msg == "sendtask")
{
    taskHandler.SendTasks();
}
if (msg.Contains("edit"))
{
    String[] msg_parts = msg.Split(':');
    index = Convert.ToInt32(msg_parts[1]);
    taskHandler.ReceiveTask(readerStream);
    taskHandler.DeleteTask(index);
    taskHandler.AddTask(index);
    taskHandler.SendTasks();
}
if (msg.Contains("del"))
{
    String[] msg_parts = msg.Split(':');
    index = Convert.ToInt32(msg_parts[1]);
    taskHandler.DeleteTask(index);
    taskHandler.SendTasks();
}
}
catch (Exception ex)
{
    String exc = Convert.ToString(ex);
    byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
    fs.Write(array, 0, array.Length);
}
finally
{
    if (readerStream != null)
        readerStream.Close();
    if (client == null)
        client.Close();
}
}
}
}

```

Приложение В.4

Файл «TaskHandler.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Net.Sockets;
using System.Diagnostics;
```

					<p style="text-align: center;">ПГУ 2.090301.26.001 ПЗ</p>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		99

```

using System.Globalization;
using System.Threading;

namespace Server
{
    class TaskHandler
    {
        public Client clientSocket;
        public TcpClient client;
        public List<Data.Data> Tasks = new List<Data.Data>();
        Thread doingtasks;
        FileStream fs;
        public int tasknum;
        int count;
        public TaskHandler(Client clientSocket, FileStream fs)
        {
            this.clientSocket = clientSocket;
            this.fs = fs;
            doingtasks = new Thread(new
ThreadStart(DoingTasks));
            doingtasks.Start();
        }

        public void LoadTasks()
        {
            try
            {
                if (System.IO.File.Exists("tasks.dat"))
                {
                    String[] str =
File.ReadAllLines("tasks.dat");
                    if (str.Length != 0)
                    {
                        BinaryFormatter formatter = new
BinaryFormatter();
                        using (FileStream fs = new
FileStream("tasks.dat", FileMode.Open))
                        {
                            List<Data.Data> openedTasks =
(List<Data.Data>)formatter.Deserialize(fs);
                            fs.Close();
                            for (int i = 0; i <
openedTasks.Count; i++)
                            {
                                Tasks.Add(openedTasks.ElementAt(i));
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

100

```

catch (Exception ex)
{
    String exc = Convert.ToString(ex);
    byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
    fs.Write(array, 0, array.Length);
}

}

public void ReceiveTask(NetworkStream readerStream)
{
    BinaryFormatter formatter = new BinaryFormatter();
    FileStream fs = new FileStream("ltask.dat",
    FileMode.Create);
    BinaryWriter bw = new BinaryWriter(fs);
    count =
int.Parse(formatter.Deserialize(readerStream).ToString());
    int i = 0;
    for (; i < count; i += 1024)
    {
        byte[] buf =
(byte[]) (formatter.Deserialize(readerStream));
        bw.Write(buf);
    }
    bw.Close();
    fs.Close();
}

public void AddTask(int index)
{
    try
    {
        Monitor.Enter(doingtasks);
        BinaryFormatter formatter = new
BinaryFormatter();
        using (FileStream fs = new
FileStream("ltask.dat", FileMode.Open))
        {
            Data.Data received_data =
(Data.Data)formatter.Deserialize(fs);
            if (index == -1)
            {
                Tasks.Add(received_data);
                tasknum = Tasks.Count - 1;
            }
            else
            {
                Tasks.Insert(index, received_data);
                tasknum = index;
            }
            fs.Close();
            Monitor.Exit(doingtasks);
        }
    }
}

```

```

CountTime(tasknum);
        System.IO.File.Delete(@"1task.dat");
        using (FileStream fs2 = new
FileStream("tasks.dat", FileMode.OpenOrCreate))
        {
            formatter.Serialize(fs2, Tasks);
            fs2.Close();
        }
    }
    catch (Exception ex)
    {
        String exc = Convert.ToString(ex);
        byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
        fs.Write(array, 0, array.Length);
    }
}

public void DeleteTask(int delindex)
{
    try
    {
        Monitor.Enter(doingtasks);
        BinaryFormatter formatter = new
BinaryFormatter();
        Tasks.RemoveAt(delindex);
        using (FileStream fs = new
FileStream("tasks.dat", FileMode.OpenOrCreate))
        {
            formatter.Serialize(fs, Tasks);
            fs.Close();
        }
        Monitor.Exit(doingtasks);
    }
    catch (Exception ex)
    {
        String exc = Convert.ToString(ex);
        byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
        fs.Write(array, 0, array.Length);
    }
}

public void SendTasks()
{
    try
    {
        BinaryFormatter formatter = new
BinaryFormatter();
        NetworkStream writerStream = client.GetStream();
        byte[] buf = new byte[1024];

```

```

FileStream fs2 = new FileStream("tasks.dat",
    FileMode.OpenOrCreate);
    BinaryReader br = new BinaryReader(fs2);
    long k = fs2.Length;
    formatter.Serialize(writerStream, k.ToString());
    while ((count = br.Read(buf, 0, 1024)) > 0)
    {
        formatter.Serialize(writerStream, buf);
    }
    br.Close();
    fs2.Close();
}
catch (Exception ex)
{
    String exc = Convert.ToString(ex);
    byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
    fs.Write(array, 0, array.Length);
}

}

public void CountTime(int tasknum)
{
    try
    {
        if (Tasks.Count == 0)
        {
            return;
        }
        if (Tasks.ElementAt(tasknum).T.Count == 0)
        {
            Tasks.ElementAt(tasknum).lasttime = null;
            Tasks.ElementAt(tasknum).nexttime =
Convert.ToString(DateTime.Now);
        }
        else
        {
            for (int i = 0; i <
Tasks.ElementAt(tasknum).T.Count; i++)
            {
                if
                (Tasks.ElementAt(tasknum).T.ElementAt(i).enable == "Включен")
                {
                    if (Tasks.ElementAt(tasknum).T.ElementAt(i).doing.date != null)
                    {
                        DateTime dt;

                        DateTime.TryParse(Tasks.ElementAt(tasknum).T.ElementAt(i).doing.
date + " " +

Tasks.ElementAt(tasknum).T.ElementAt(i).doing.time, out dt);

```

```

if (DateTime.Now >= dt)
    {
        DeleteTask(tasknum);
        break;
    }
    }
    if
(TasksWithElementAt(tasknum).T.WithElementAt(i).repeat ==
"Однократный")
    {

TasksWithElementAt(tasknum).lasttime = null;

TasksWithElementAt(tasknum).nexttime =
TasksWithElementAt(tasknum).T.WithElementAt(i).begin.date + " "
+
TasksWithElementAt(tasknum).T.WithElementAt(i).begin.time;
        DateTime dt;

DateTime.TryParse(TasksWithElementAt(tasknum).nexttime, out dt);
        if (DateTime.Now >= dt)
        {

TasksWithElementAt(i).T.RemoveAt(i);

                break;
            }
        }

        if
(TasksWithElementAt(tasknum).T.WithElementAt(i).repeat == "Ежедневный")
        {
            if
(TasksWithElementAt(tasknum).lasttime == null)
            {

TasksWithElementAt(tasknum).nexttime =
TasksWithElementAt(tasknum).T.WithElementAt(i).begin.date + " "
+
TasksWithElementAt(tasknum).T.WithElementAt(i).begin.time;
                DateTime dt2;

DateTime.TryParse(TasksWithElementAt(tasknum).T.WithElementAt(i).begin.
date + " "
+
TasksWithElementAt(tasknum).T.WithElementAt(i).begin.time, out dt2);

TasksWithElementAt(tasknum).times.Add(dt2);
            }
        }
    else
    {
        DateTime outtime;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

104

```

DateTime.TryParse (Tasks.ElementAt (tasknum).nexttime, out
outtime);

                                Calendar myCal =
CultureInfo.InvariantCulture.Calendar;
                                outtime =
myCal.AddDays (outtime,
Convert.ToInt32 (Tasks.ElementAt (tasknum).T.ElementAt (i).day));

Tasks.ElementAt (tasknum).nexttime = Convert.ToString (outtime);
                                }
                                }

                                if
(Tasks.ElementAt (tasknum).T.ElementAt (i).repeat ==
"Еженедельный")
                                {
                                Data.Data.TriggerData trd =
Tasks.ElementAt (tasknum).T.ElementAt (i);
                                List<String> DaysofWeek =
Tasks.ElementAt (tasknum).T.ElementAt (i).week.days.Split ('
').ToList ();
                                List<String> EngDaysofWeek = new
List<String> ();
                                for (int n = 0; n <
DaysofWeek.Count (); n++)
                                {
                                if (DaysofWeek.ElementAt (n)
== "Пн") { EngDaysofWeek.Add ("Monday"); }
                                if (DaysofWeek.ElementAt (n)
== "Вт") { EngDaysofWeek.Add ("Tuesday"); }
                                if (DaysofWeek.ElementAt (n)
== "Ср") { EngDaysofWeek.Add ("Wednesday"); }
                                if (DaysofWeek.ElementAt (n)
== "Чт") { EngDaysofWeek.Add ("Thursday"); }
                                if (DaysofWeek.ElementAt (n)
== "Пт") { EngDaysofWeek.Add ("Friday"); }
                                if (DaysofWeek.ElementAt (n)
== "Сб") { EngDaysofWeek.Add ("Saturday"); }
                                if (DaysofWeek.ElementAt (n)
== "Вс") { EngDaysofWeek.Add ("Sunday"); }
                                }
                                if (Tasks.ElementAt (tasknum).lasttime == null)

DateTime.TryParse (Tasks.ElementAt (tasknum).T.ElementAt (i).begin.
date + " "
                                +
Tasks.ElementAt (tasknum).T.ElementAt (i).begin.time, out
trd.startweek);
                                Calendar myCal = CultureInfo.InvariantCulture.Calendar;
trd.endweek = trd.startweek;
trd.endweek = myCal.AddDays (trd.endweek, 7);

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

105

```

for (DateTime strt = trd.startweek; strt <= trd.endweek; )
{
    for (int j = 0; j <
EngDaysofWeek.Count(); j++)
    {
        if
(Convert.ToString(myCal.GetDayOfWeek(strt)) ==
EngDaysofWeek.ElementAt(j))
        {

Tasks.ElementAt(tasknum).nexttime = Convert.ToString(strt);

EngDaysofWeek.RemoveAt(j);

DaysofWeek.RemoveAt(j);

                                goto end_cycle;
                                }
                                }
                                strt = myCal.AddDays(strt,
1);
                                }
                                end_cycle:
                                if (EngDaysofWeek.Count() == 0)
                                {
                                    trd.startweek =
myCal.AddWeeks(trd.startweek, Convert.ToInt32(trd.week.week));
                                }
                                else
                                {
                                    trd.DaysofWeek =
String.Empty;
                                    for (int k = 0; k <
DaysofWeek.Count(); k++)
                                    {
                                        trd.DaysofWeek +=
DaysofWeek.ElementAt(k) + " ";
                                    }
                                }

Tasks.ElementAt(tasknum).T.Insert(i, trd);

Tasks.ElementAt(tasknum).T.RemoveAt(i + 1);
                                }
                                if (Tasks.ElementAt(tasknum).T.ElementAt(i).repeat ==
"Ежемесячный")
                                {
                                    Data.Data.TriggerData trd =
Tasks.ElementAt(tasknum).T.ElementAt(i);
                                    List<String> Month =
Tasks.ElementAt(tasknum).T.ElementAt(i).month.month.Split('
').ToList();

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

105

```

List<int> MonthNum = new List<int>();
List<String> Days;
for (int n = 0; n <
Month.Count(); n++)
{
    if (Month.ElementAt(n) ==
"Янв") { MonthNum.Add(1); }
    if (Month.ElementAt(n) ==
"Фев") { MonthNum.Add(2); }
    if (Month.ElementAt(n) ==
"Мар") { MonthNum.Add(3); }
    if (Month.ElementAt(n) ==
"Апр") { MonthNum.Add(4); }
    if (Month.ElementAt(n) ==
"Май") { MonthNum.Add(5); }
    if (Month.ElementAt(n) ==
"Июн") { MonthNum.Add(6); }
    if (Month.ElementAt(n) ==
"Июл") { MonthNum.Add(7); }
    if (Month.ElementAt(n) ==
"Авг") { MonthNum.Add(8); }
    if (Month.ElementAt(n) ==
"Сен") { MonthNum.Add(9); }
    if (Month.ElementAt(n) ==
"Окт") { MonthNum.Add(10); }
    if (Month.ElementAt(n) ==
"Ноя") { MonthNum.Add(11); }
    if (Month.ElementAt(n) ==
"Дек") { MonthNum.Add(12); }
}
if
(Tasks.ElementAt(tasknum).lasttime == null)
{

DateTime.TryParse(Tasks.ElementAt(tasknum).T.ElementAt(i).begin.
date + " "
+
Tasks.ElementAt(tasknum).T.ElementAt(i).begin.time, out
trd.startweek);

Days =
Tasks.ElementAt(tasknum).T.ElementAt(i).month.days.Split('
').ToList();
}
else
{
Days =
trd.DaysofWeek.Split(' ').ToList();
}

Calendar myCal = CultureInfo.InvariantCulture.Calendar;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

107

```

for (int j = 0; j < MonthNum.Count; j++)
{
    if
    (Convert.ToInt32(myCal.GetMonth(trd.startweek)) ==
    MonthNum.ElementAt(j))
    {
        if
        (Tasks.ElementAt(tasknum).T.ElementAt(i).month.days != null)
        {
            for (int n = 1; n <=
            DateTime.DaysInMonth(myCal.GetYear(trd.startweek),
            MonthNum.ElementAt(j)); n++)
            {
                for (int m = 0;
                m < Days.Count() - 1; m++)
                {
                    if (n ==
                    Convert.ToInt32(Days.ElementAt(m)))
                    {
                        trd.startweek.AddMonths(MonthNum.ElementAt(j) -
                        trd.startweek.Month);

                        trd.startweek.AddDays(n);

                        Tasks.ElementAt(tasknum).nexttime =
                        Convert.ToString(trd.startweek) +
                        " "
                        + Tasks.ElementAt(tasknum).T.ElementAt(i).begin.time;

                        Days.RemoveAt(m);

                        trd.DaysofWeek += Days.ToString() + " ";

                        goto
                        end_cycle;
                    }
                }
            }
        }
        if
        (Tasks.ElementAt(tasknum).T.ElementAt(i).month.weeks != null)
        {
            List<String> Weeks =
            Tasks.ElementAt(tasknum).T.ElementAt(i).month.weeks.Split('
            ').ToList();

            List<String>

            DaysofWeek =
            Tasks.ElementAt(tasknum).T.ElementAt(i).month.wdays.Split('
            ').ToList();

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

108

```

for (int n = 0; n < Weeks.Count - 1; n++)
{

trd.startweek.AddMonths (MonthNum.ElementAt(j) -
trd.startweek.Month);

List<DateTime>
dt_list = new List<DateTime>();
for (int m = 1;
m <= DateTime.DaysInMonth(myCal.GetYear(trd.startweek),
MonthNum.ElementAt(j)); m++)
{

trd.startweek.AddDays (m);

for (int l =
0; l < DaysOfWeek.Count; l++)
{
if
(Convert.ToString (myCal.GetDayOfWeek (trd.startweek)) ==
DaysOfWeek.ElementAt (l))
{

dt_list.Add(trd.startweek);

goto
end_2;

}

}

}
end_2:

Tasks.ElementAt (tasknum).nexttime =
Convert.ToString (dt_list.ElementAt (n));

}

}

end_cycle:
if (Days.Count == 0)
{
MonthNum.RemoveAt (j);
trd.DaysOfWeek =
Tasks.ElementAt (tasknum).T.ElementAt (i).month.days;
}

}

Tasks.ElementAt (tasknum).T.Insert (i, trd);

Tasks.ElementAt (tasknum).T.RemoveAt (i + 1);
}

if (Tasks.ElementAt (tasknum).T.ElementAt (i).delay != null)

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

109

```

        {
            String delay =
Tasks.ElementAt(tasknum).T.ElementAt(i).delay;
            String num = String.Empty;
            for (int j = 0; j < delay.Length;
j++)
            {
                if (Char.IsDigit(delay[j]))
                {
                    num += delay[j];
                }
            }
            DateTime t;

DateTime.TryParse(Tasks.ElementAt(tasknum).nexttime, out t);
            if (delay.Contains("сек. "))
            {

t.AddSeconds(Convert.ToInt32(num));
            }
            if (delay.Contains("мин. "))
            {

t.AddMinutes(Convert.ToInt32(num));
            }
            if (delay.Contains("ч. "))
            {

t.AddHours(Convert.ToInt32(num));
            }
            if (delay.Contains("д. "))
            {
                t.AddDays(Convert.ToInt32(num));
            }
            Tasks.ElementAt(tasknum).nexttime =
Convert.ToString(t);
        }
    }
}
catch (Exception ex)
{
    String exc = Convert.ToString(ex);
    byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
    fs.Write(array, 0, array.Length);
}

public void DoingTasks()
{

```

```

try
{
    while (true)
    {
        if (Tasks.Count > 0)
        {
            for (int i = 0; i < Tasks.Count; i++)
            {
                for (int j = 0; j <
Tasks.ElementAt(i).A.Count; j++)
                {
                    if
(Tasks.ElementAt(i).A.ElementAt(j).enable == "Включена" &&
Tasks.ElementAt(i).nexttime
!= null)
                        {
                            DateTime dt;

DateTime.TryParse(Tasks.ElementAt(i).nexttime, out dt);
                            if (dt <= DateTime.Now)
                            {
                                String path =
Tasks.ElementAt(i).A.ElementAt(j).path;
                                ProcessStartInfo si =
new ProcessStartInfo(path);

                                si.ErrorDialog = true;
                                Process.Start(si);
                                if (Tasks.Count > 0)

Tasks.ElementAt(i).lasttime = Tasks.ElementAt(i).nexttime;
                                    CountTime(i);
                                }
                                if
(Tasks.ElementAt(i).T.Count == 0) DeleteTask(i);
                                    }
                                    if (Tasks.Count == 0) break;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        String exc = Convert.ToString(ex);
        byte[] array =
System.Text.Encoding.Default.GetBytes(exc);
        fs.Write(array, 0, array.Length);
    }
}
}

```

Приложение С

Листинг библиотеки Data

Приложение С.1

Файл «Data.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Data
{
    [Serializable]
    public class Data
    {
        [Serializable]
        public struct GeneralData
        {
            public String name;
            public String author;
            public String description;
        }
        public GeneralData genData;

        [Serializable]
        public struct TriggerData
        {
            public String repeat;

            [Serializable]
            public struct Begin
            {
                public String time;
                public String date;
            }
            public Begin begin;

            public String day;

            [Serializable]
            public struct Week
            {
                public String week;
                public String days;
            }
            public Week week;
        }
    }
}
```

```

[Serializable]
    public struct Month
    {
        public String month;
        public String days;
        public String weeks;
        public String wdays;
    }
    public Month month;

    public String delay;

    [Serializable]
    public struct Doing
    {
        public String time;
        public String date;
    }
    public Doing doing;

    public String enable;

    public String tableDescription;

}
public List<TriggerData> T = new List<TriggerData>();

[Serializable]
public struct ActionData
{
    public String path;
    public String enable;
}
public List<ActionData> A = new List<ActionData>();

public String password;

public String lasttime;
public String nexttime;

}
}

```

Приложение D

Результаты работы программы

Приложение D.1 – Вид начального экрана приложения

Название	Описание	Последний з...	Следующи...	Триггеры
Задача по ...	Запуск Ia...	07.06.2020 5:...	09.06.2020 ...	В 5:55:07...
Калькулятор	Открыть ...	Задача ещё ...	10.06.2020 ...	В 6:16:00...
Изображен...	Открыть ...	Задача ещё ...	08.06.2020 ...	В 12:30:0...

Рисунок 34 – Вид главного окна приложения

Приложение D.2 – Окно создания задачи. Вкладка «Общие».

Рисунок 35 – Окно создания задачи. Вкладка «Общие».

Приложение D.3 – Окно создания задачи. Вкладка «Триггеры»

Создание новой задачи

Триггеры

Условия запуска задачи (не обязательны)

Триггер	Описание	Состояние

Создать Изменить Удалить

Назад Далее

Рисунок 36 – Окно создания задачи. Вкладка «Триггеры».

Приложение D.4 – Окно создания триггера.

Создание триггера

Начать задачу: По расписанию

Параметры

☒ Однократно
☐ Ежедневно
☐ Еженедельно
☐ Ежемесячно

Начать: 6:26:41 04.06.2020

Дополнительные параметры

☐ Отложить на (произвольная задержка):
☐ Срок действия: 6:26:41 04.06.2020
☒ Включено

Отмена Сохранить

Рисунок 37 – Окно создания триггера.

Приложение D.5 – Окно создания задачи. Вкладка «Действия»

Создание новой задачи

Действия

Действия, которые будут выполняться данной задачей

Действие	Подробности

Создать Изменить Удалить

Назад Далее

Рисунок 38 – Окно создания задачи. Вкладка «Действия».

Приложение D.6 – Окно создания действия.

Создание действия

Укажите действие для данной задачи.

Программа или файл:

Обзор

☒ Включено

Отмена Сохранить

Рисунок 39 – Окно создания действия.

Приложение D.7 – Окно создания задачи. Вкладка «Безопасность»

Создание новой задачи

Безопасность

Защита задачи паролем от изменения или удаления (не обязательна)

Допустимая длина пароля: 4-16 символов, допустимые символы: A-Я, A-Z, 0-9

Пароль

Повторный ввод пароля

Назад Готово

Рисунок 40 – Окно создания задачи. Вкладка «Безопасность».

Приложение D.8 – Окно изменения задачи.

Изменение задачи

Общие

Общая информация о задаче

Имя

Автор

Описание

Назад Далее

Рисунок 41 – Окно изменения задачи.

Приложение Е.1 – UML–диаграмма классов клиента



Приложение Е.2 – UML–диаграмма классов сервера.

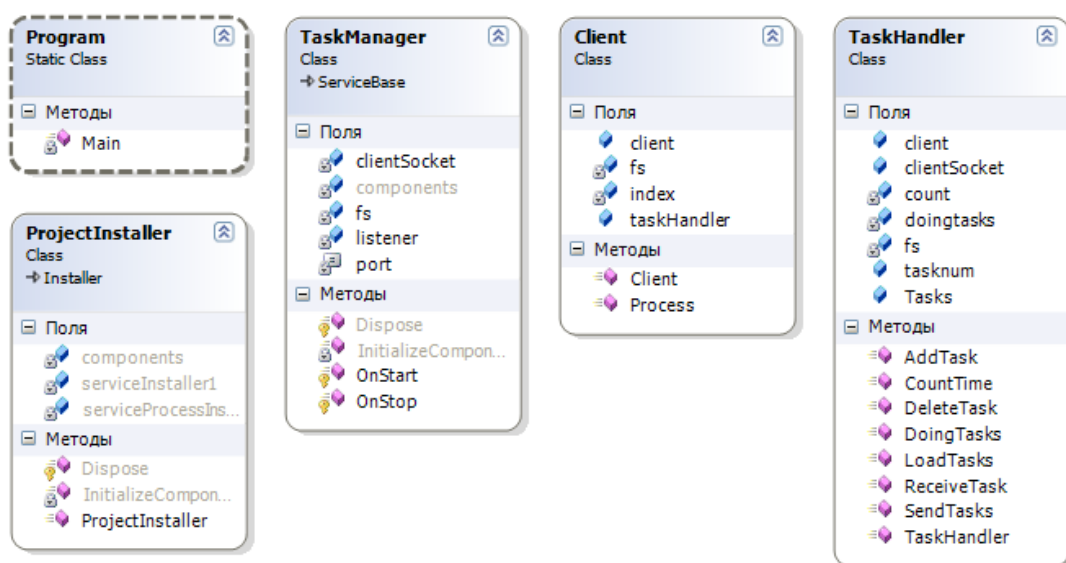


Рисунок 43 – UML–диаграмма классов сервера.

Приложение Е.3 – UML–диаграмма вариантов использования клиента.



Рисунок 44 – UML–диаграмма вариантов использования клиента.

Приложение Е.4 – UML–диаграмма вариантов использования сервера.



Рисунок 45 – UML–диаграмма вариантов использования сервера.

Приложение Е.5 – UML–диаграмма последовательности.

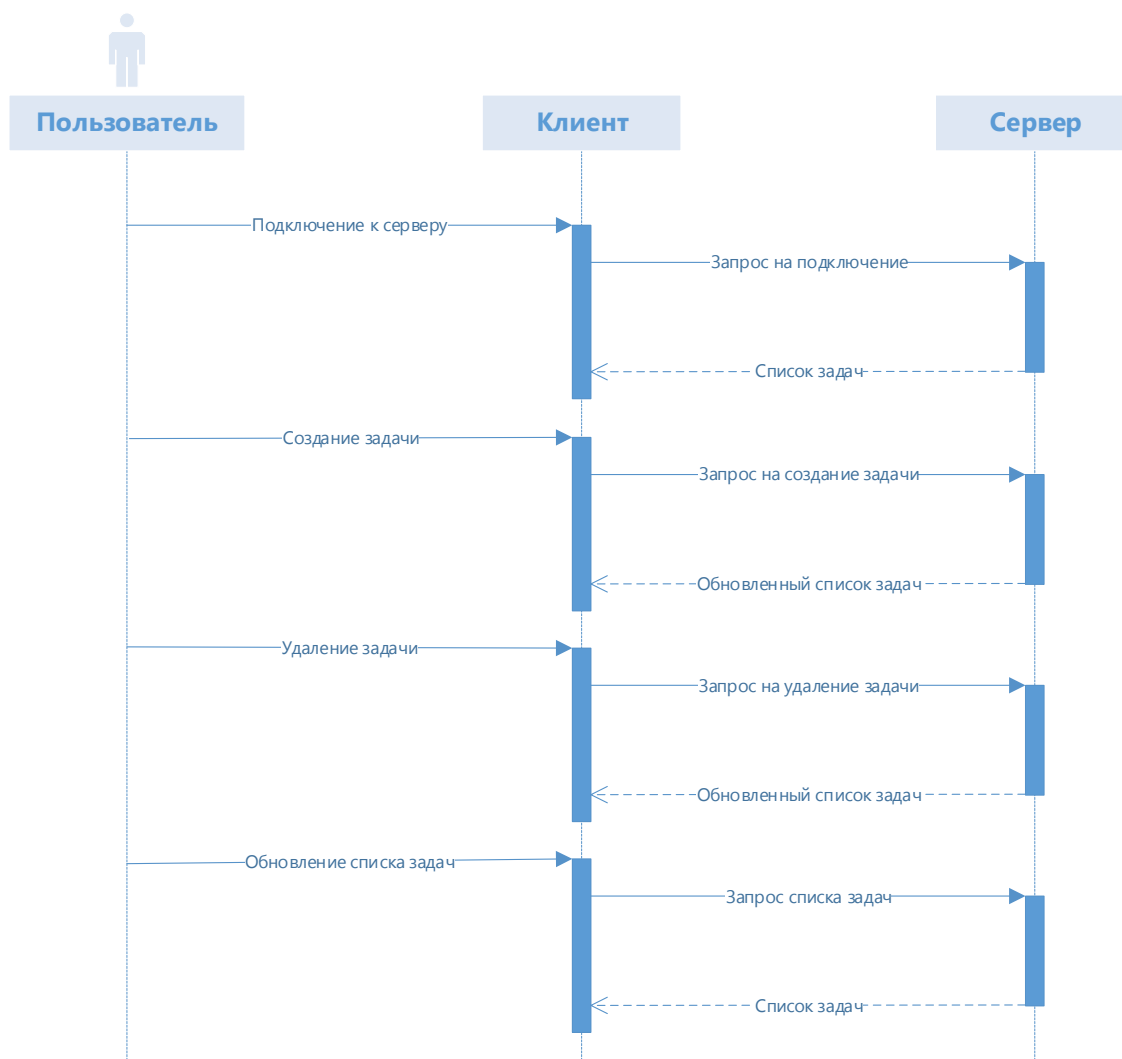


Рисунок 46 – UML–диаграмма последовательности.

Приложение Е.6 – UML–диаграмма деятельности клиента.

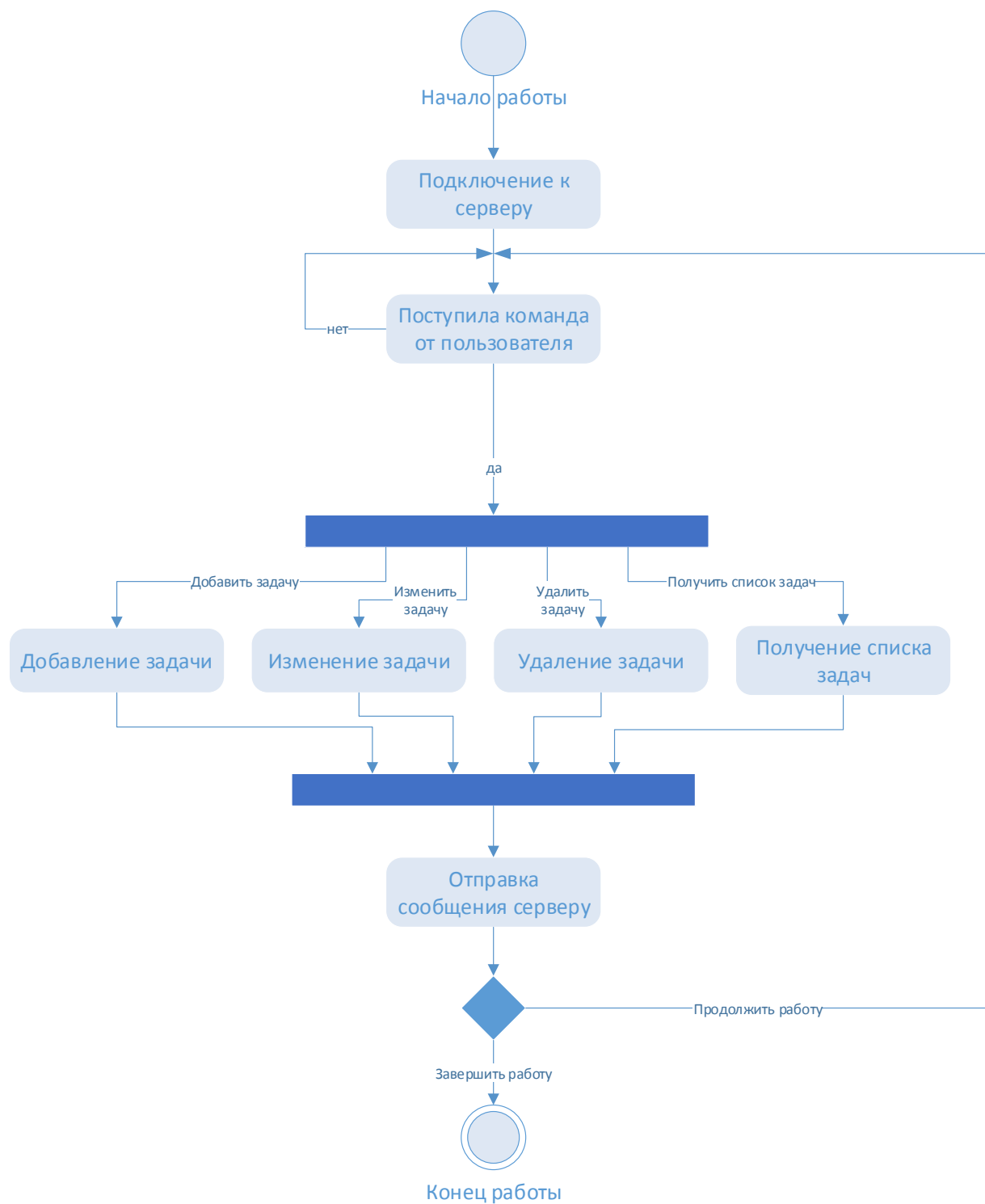


Рисунок 47 – UML–диаграмма деятельности клиента.

Приложение Е.7 – UML–диаграмма деятельности сервера.

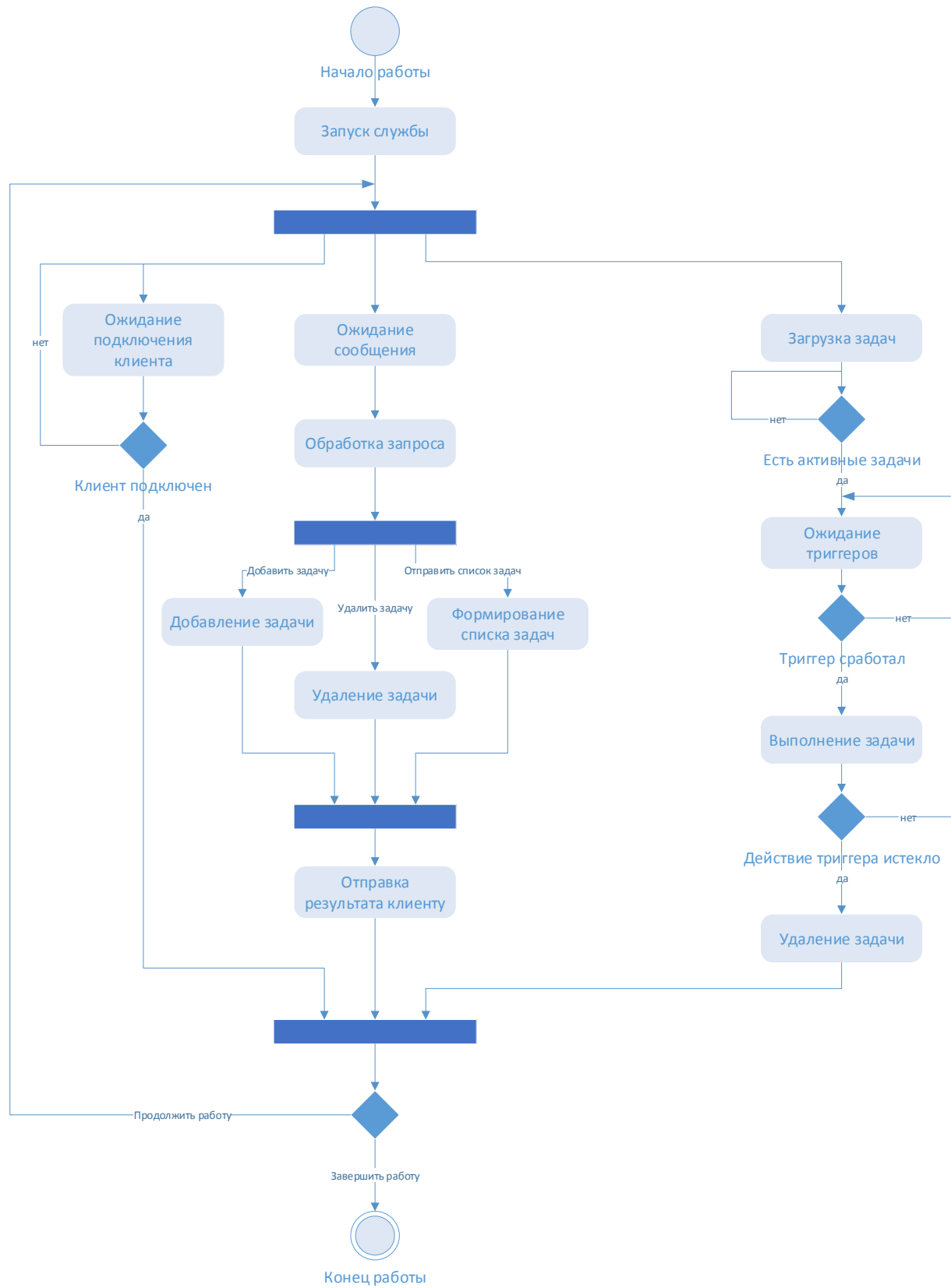


Рисунок 48 – UML–диаграмма деятельности сервера.

Приложение Е.8 – UML–диаграмма развертывания

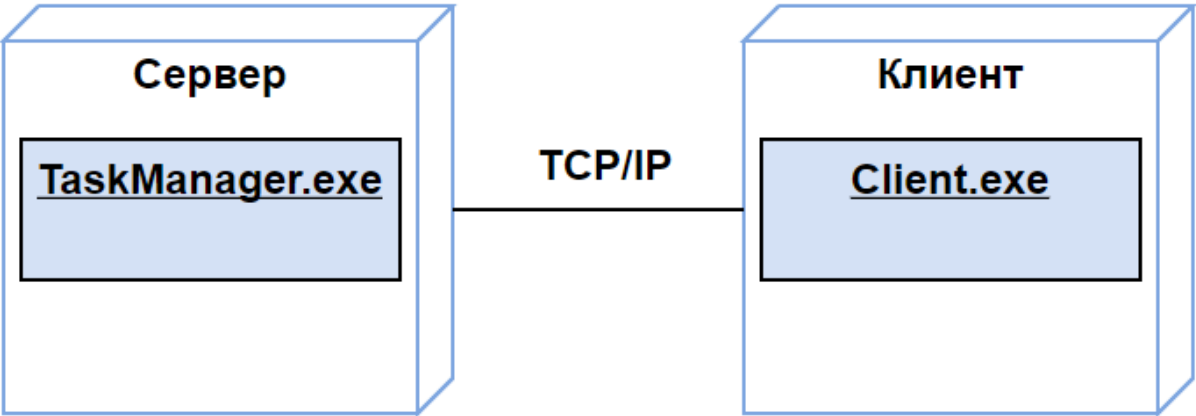


Рисунок 49 – UML–диаграмма развертывания.

Приложение F

Презентация

Клиент-серверное приложение «Планировщик задач»

Презентацию подготовила:
студентка группы 16BV2 Мультяева И.И.

Руководитель бакалаврской работы:
к.т.н., доцент Карамышева Н.С.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

124

Постановка задачи

Необходимо разработать клиент–серверное приложение, позволяющее создавать, изменять, удалять и в установленное время выполнять задачи на локальных и удаленных ЭВМ.

- Клиент должен подключаться к локальному или удаленному серверу для управления его задачами.
- Сервер должен выполнять требуемые запросы клиента, а также запускать созданные задачи.
- Сетевое взаимодействие между клиентом и сервером должно осуществляться с использованием протокола TCP/IP.

Диаграмма вариантов использования клиента

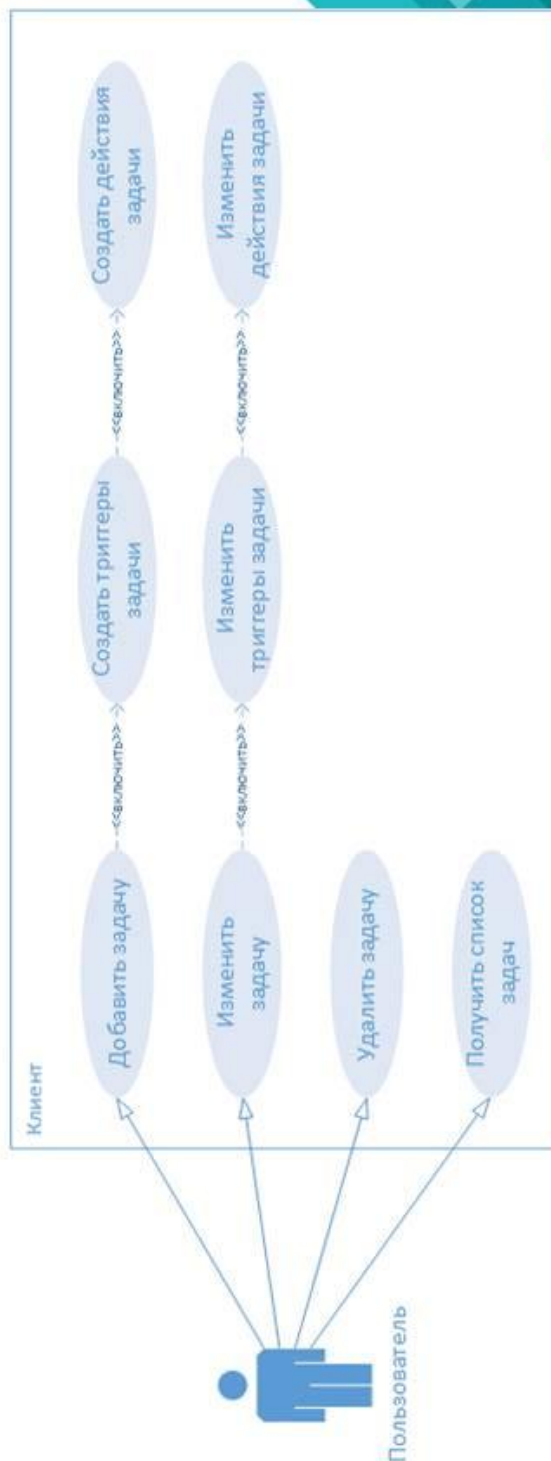


Диаграмма вариантов использования сервера

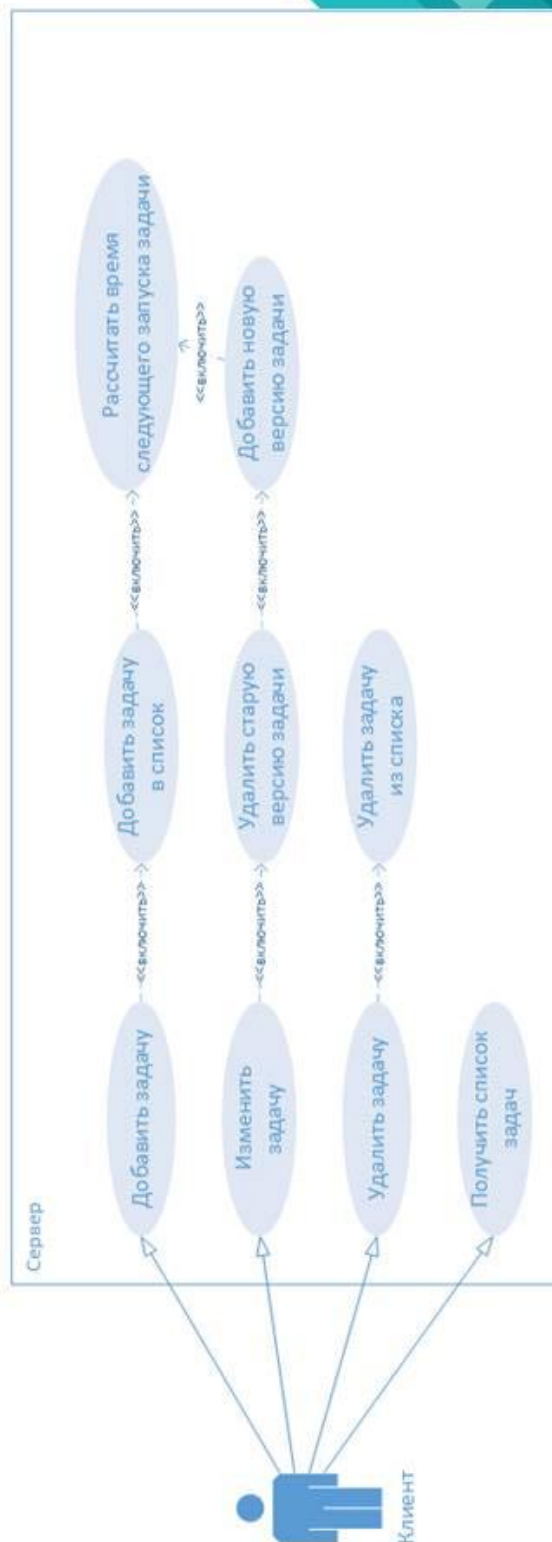


Диаграмма последовательности

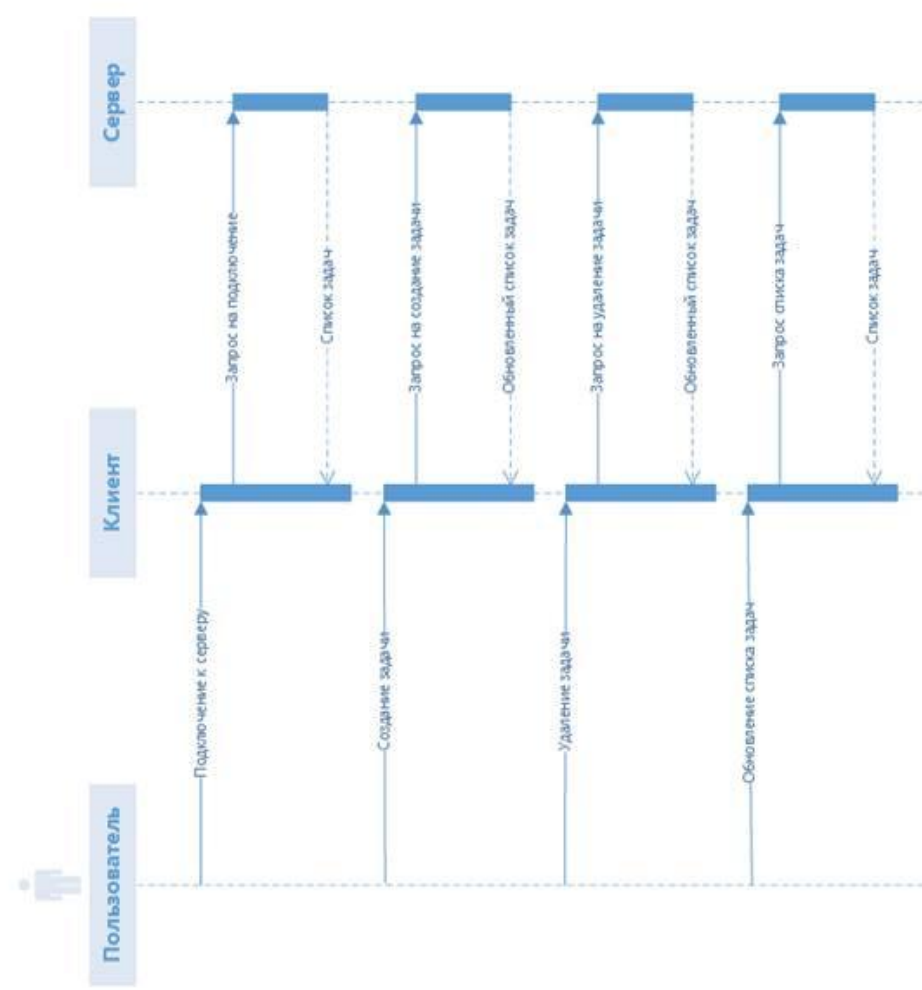




Диаграмма классов сервера





Интерфейс клиентского приложения

Планировщик задач

Добавить задачу

Изменить задачу

Удалить задачу

Список активных задач:
Всего: 0

Название

Описание

Последний з...

Следующий...

Триггеры

Обновить

Вы подключены к серверу:

... ..

настройка сервера



Создание задачи

Создание новой задачи

Общие

Общая информация о задаче

Общие Триггеры

Общие Действия

Безопасность

Имя

Автор

Описание

User

Назад

Далее

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

132

Обновление списка задач

Планировщик задач

Добавить задачу

Изменить задачу

Удалить задачу

Вы подключены к серверу:
127.0.0.1
настройка сервера

Обновить

Список активных задач:
Всего: 3

Название	Описание	Последний з...	Следующий...	Триггеры
Задача по ...	Запуск Ia...	07.06.2020 5:...	09.06.2020 ...	В 5:55:07 ...
Калькулятор	Открыть ...	Задача ещё ...	10.06.2020 ...	В 6:16:00...
Изображен...	Открыть ...	Задача ещё ...	08.06.2020 ...	В 12:30:0...



Заключение

В ходе работы было создано клиент-серверное приложение, полностью соответствующее заявленным требованиям.

- Пользователь может создавать, изменять и удалять задачи на локальных и удаленных компьютерах.
 - Каждая задача имеет свой список триггеров и действий.
 - Сервер принимает и обрабатывает запросы пользователя, параллельно занимаясь выполнением задач.
- 



Спасибо за внимание!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.26.001 ПЗ

Лист

135

