СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc451340224)

[2 Этапы проектирования задачи 4](#_Toc451340225)

[2.1 Программные средства 4](#_Toc451340226)

[2.2 Обзор предметой области 6](#_Toc451340227)

[2.3 Программирование 9](#_Toc451340229)

[3 Отладка и тестирование 11](#_Toc451340230)

[4 Инструкция программиста 17](#_Toc451340231)

[4.1 Выбор кодека 18](#_Toc451340232)

[4.2 Опытная эксплуатация и внедрение проекта 19](#_Toc451340233)

[5 Инструкция пользователя 20](#_Toc451340234)

[Заключение 20](#_Toc451340234)

1 Описание исходных данных

Исходными данными для проектирования являются техническое задание и материал, который был предоставлен предприятием «МИКОС».

Требуется создать многопоточный сервис для общения, поддерживающий возможность работы в локальной и интернет сетях.

Техническое задание содержит:

* цель разработки;
* анализ предметной области;
* требования к приложению;
* используемое программное обеспечение;

Материал предоставленный программистом компании, содержит:

* Основные библиотеки для создания приложения;

Все библиотеки находятся на общедоступном ресурсе – Hoogle, который является подобием Google, но его возможности ограничены библиотеками языка Haskell.

* Интернет ресурсы с информацией;
* Книги по программированию на Haskell;

Основной материал по программированию на языке Haskell взят с сайта «anton-k.github.io», руководство-книга по работе с потоками и многопоточными приложениями – «http://chimera.labs.oreilly.com/books/1230000000929». Также для поиска использовался сайт «cyberforum.ru».

2.1 Этапы проектирования задачи

Для создания программы необходимо ознакомится со средствами для работы в мультипоточной среде.

Процесс построения приложения был разбит на следующие этапы:

* обучение концепции языка Haskell;
* настройка среды разработчика под конкретное приложение(подключение нужных функций, библиотек, шаблонов);
* построение программы.

Для написания кода на языках «HTML», «CSS», «JavaScript», «Haskell» была использована программа «Sublime Text 3».

Для оценки затрат ресурсов компьютера использовалась программа «ThreadScope» .

Для компиляции и тестирования работы потоковых каналов программы использовались несколько компьютеров(стационарный и два портативных).

2.1.1 Программные средства

Sublime Text – бесплатный редактор с открытым кодом для [веб–разработчиков](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0). Brackets ориентирован на работу с [HTML](https://ru.wikipedia.org/wiki/HTML), [CSS](https://ru.wikipedia.org/wiki/CSS) и [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). Эти же технологии лежат в основе самого редактора, что обеспечивает его кроссплатформенность т.е. совместимость с операционными системами Mac, Windows и Linux. Sublime Text создан и развивается [Adobe Systems](https://ru.wikipedia.org/wiki/Adobe_Systems) под лицензией [MIT License](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT) и поддерживается на [GitHub](https://ru.wikipedia.org/wiki/GitHub).

На сегодняшний день сообществом создано множество расширений, добавляющих большинство необходимых инструментов для работы над кодом, таких как система контроля версий [Git](https://ru.wikipedia.org/wiki/Git), просмотр HTML–кода в браузере в реальном времени (Live Preview), синхронизация с [FTP](https://ru.wikipedia.org/wiki/FTP) (Git–FTP). Принять участие в разработке и поддержке расширений может любой желающий.

На «Sublime Text» был установлен следующий ряд расширений.

Code Folding – расширение, позволяющее сворачивать код в блоке согласно его структуре.

Emmet – расширение, позволяющее ускорить набор HTML тегов записывая их краткую форму и при нажатии на «tab» разворачивая их в полноценный HTML код.

Documents Toolbar – расширение, добавляющие к стандартному интерфейс программы панель вкладок.

Beautify, Beautifer – расширение, которое добавляет пробелы и переносы на новую строку в код, делая его более читабельным.

QuickSearch – расширение, которое при двойном клике на выражение подсвечивает все его вхождения в документ.

HTML – стандартный язык разметки вэб–страниц. Интерпретируется браузерами, полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства. На сайте используется для разметки страницы.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

JavaScript – прототипно–ориентированный сценарный язык программирования, обычно используется как встраиваемый язык для программного доступа к объектам приложений. На сайте используется для отправки AJAX запросов при нажатии кнопок в управляющей версии, AJAX запросы – это технология обращения к серверу без перезагрузки страницы.

Stack – это новый инструмент от команды FP Complete, предназначенный для работы с Haskell-проектами. Он предназначен для удовлетворения потребностей больших групп программистов, любителей, и компаний, думающих о том, чтобы начать использовать haskell. Программа предназначена, для упрощения программирования средой, используемой новичками, обеспечивая возможности настройки и широкий функционал для опытных разработчиков.

ThreadScope – программа позволяет следить и оценивать загруженность производительности параллельных программ на haskell. Используя ThreadScope можно проверить приложение, чтобы увидеть, что созданная программа сбалансировано обрабатывается всеми доступными процессорами и найти ошибки, связанных с неправильной работой сборщика мусора или плохой балансировки нагрузки.

2.1.2 Обзор программных компонентов

Проектирование программных каналов началось с исследования многопоточных инструментов программирования.

Виды инструментов:

* Взаимоисключения (mutex, мьютекс) — это объект синхронизации, который устанавливается в особое сигнальное состояние, когда не занят каким-либо потоком. Только один поток владеет этим объектом в любой момент времени, отсюда и название таких объектов (от английского mutually exclusive access — взаимно исключающий доступ) — одновременный доступ к общему ресурсу исключается. После всех необходимых действий мьютекс освобождается, предоставляя другим потокам доступ к общему ресурсу. Объект может поддерживать рекурсивный захват второй раз тем же потоком, увеличивая счетчик, не блокируя поток, и требуя потом многократного освобождения. Такова, например, критическая секция в Win32. Тем не менее, есть и такие реализации, которые не поддерживают такое и приводят к взаимной блокировке потока при попытке рекурсивного захвата. Это FAST\_MUTEX в ядре Windows.
* Семафоры представляют собой доступные ресурсы, которые могут быть приобретены несколькими потоками в одно и то же время, пока пул ресурсов не опустеет. Тогда дополнительные потоки должны ждать, пока требуемое количество ресурсов не будет снова доступно. Семафоры очень эффективны, поскольку они позволяют одновременный доступ к ресурсам.
* События. Объект, хранящий в себе 1 бит информации «просигнализирован или нет», над которым определены операции «просигнализировать», «сбросить в непросигнализированное состояние» и «ожидать». Ожидание на просигнализированном событии есть отсутствие операции с немедленным продолжением исполнения потока. Ожидание на непросигнализированном событии приводит к приостановке исполнения потока до тех пор, пока другой поток (или же вторая фаза обработчика прерывания в ядре ОС) не просигнализирует событие. Возможно ожидание нескольких событий в режимах «любого» или «всех». Возможно также создание события, автоматически сбрасываемого в непросигнализированное состояние после пробуждения первого же — и единственного — ожидающего потока (такой объект используется как основа для реализации объекта «критическая секция»). Активно используются в MS Windows, как в режиме пользователя, так и в режиме ядра. Аналогичный объект имеется и в ядре Linux под названием kwait\_queue.
* Критические секции обеспечивают синхронизацию подобно мьютексам, за исключением того, что объекты, представляющие критические секции, доступны в пределах одного процесса. События, мьютексы и семафоры также можно использовать в однопроцессном приложении, однако реализации критических секций в некоторых ОС (например, Windows NT) обеспечивают более быстрый и более эффективный механизм взаимно-исключающей синхронизации — операции «получить» и «освободить» на критической секции оптимизированы для случая единственного потока (отсутствия конкуренции) с целью избежать любых ведущих в ядро ОС системных вызовов. Подобно мьютексам объект, представляющий критическую секцию, может использоваться только одним потоком в данный момент времени, что делает их крайне полезными при разграничении доступа к общим ресурсам.
* Условные переменные (condvars). Сходны с событиями, но не являются объектами, занимающими память — используется только адрес переменной, понятие «содержимое переменной» не существует, в качестве условной переменной может использоваться адрес произвольного объекта. В отличие от событий, установка условной переменной в просигнализированное состояние не влечет за собой никаких последствий в случае, если на данный момент нет потоков, ожидающих на переменной. Установка события в аналогичном случае влечет за собой запоминание состояния «просигнализировано» внутри самого события, после чего следующие потоки, желающие ожидать события, продолжают исполнение немедленно без остановки. Для полноценного использования такого объекта необходима также операция «освободить mutex и ожидать условную переменную атомарно». Активно используются в UNIX-подобных ОС. Дискуссии о преимуществах и недостатках событий и условных переменных являются заметной частью дискуссий о преимуществах и недостатках Windows и UNIX.
* Мьютекс, поддерживающий рекурсивный захват, с семантикой разделяемого или эксклюзивного захвата. Семантика: объект может быть либо свободен, либо захвачен произвольным числом потоков разделяемым образом, либо захвачен всего одним потоком эксклюзивным образом. Любые попытки осуществить захваты, нарушающее это правило, приводят к блокировке потока до тех пор, пока объект не освободится так, чтобы сделать захват разрешенным. Также есть операции вида TryToAcquire — никогда не блокирует поток, либо захватывает, либо (если нужна блокировка) возвращает FALSE, ничего не делая. Используется в ядре Windows, особенно в файловых системах — так, например, любому кем-то открытому дисковому файлу соответствует структура FCB, в которой есть 2 таких объекта для синхронизации доступа к размеру файла. Один из них — paging IO resource — захватывается эксклюзивно только в пути обрезания файла, и гарантирует, что в момент обрезания на файле нет активного ввода-вывода от кэша и от отображения в память.
* Rundown protection. Полудокументированный (вызовы присутствуют в файлах-заголовках, но отсутствуют в документации) объект в ядре Windows. Счетчик с операциями «увеличить», «уменьшить» и «ждать». Ожидание блокирует поток до тех пор, пока операции уменьшения не уменьшат счетчик до нуля. Кроме того, операция увеличения может отказать, и наличие активного в данный момент времени ожидания заставляет отказывать все операции увеличения.

2.1.3 Программирование

Программирование – процесс создания компьютерных программ. Программирование основывается на использовании языков программирования, на которых записываются инструкции для компьютера. Современное приложение содержит множество таких инструкций, связанных между собой.

Текстовый редактор среды программирования может иметь специфичную функциональность, такую, как индексация имен, отображение документации, подсветка синтаксиса, средства визуального создания пользовательского интерфейса. С помощью текстового редактора программист производит набор и редактирование текста создаваемой программы, который называют исходным кодом. Язык программирования определяет синтаксис и изначальную семантику исходного кода.

В настоящее время активно используются интегрированные среды разработки, включающие в свой состав также редактор для ввода и редактирования текстов программ, отладчики для поиска и устранения ошибок, трансляторы с различных языков программирования, компоновщики для сборки программы из нескольких модулей и другие служебные модули.

Большая часть работы программистов связана с написанием исходного кода, тестированием и отладкой программ на одном из языков программирования. Исходные тексты и исполняемые файлы программ являются объектами авторского права и являются интеллектуальной собственностью их авторов и правообладателей. Различные языки программирования поддерживают различные стили программирования (парадигмы программирования). Отчасти искусство программирования состоит в том, чтобы выбрать язык программирования, наиболее полно подходящий для решения поставленной задачи. Разные языки требуют от программиста различного уровня внимания к деталям при реализации алгоритма, результатом чего часто бывает компромисс между простотой и производительностью (или между временем программиста и временем пользователя).

Виды языков программирования:

* Языки низкого уровня программирования;

Язык программирования, близкий к программированию непосредственно в машинных кодах используемого реального или виртуального (например, Java, Microsoft .NET) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

Иногда одно мнемоническое обозначение соответствует целой группе машинных команд, выполняющих одинаковое действие над разными ячейками памяти процессора. Кроме машинных команд языки программирования низкого уровня могут предоставлять дополнительные возможности, такие как макроопределения (макросы). При помощи директив есть возможность управлять процессом трансляции машинных кодов, предоставляя возможность заносить константы и литеральные строки, резервировать память под переменные и размещать исполняемый код по определенным адресам. Часто эти языки позволяют работать вместо конкретных ячеек памяти с переменными.

Как правило, использует особенности конкретного семейства процессоров. Общеизвестный пример низкоуровневого языка — язык ассемблера, хотя правильнее говорить о группе языков ассемблера. Более того, для одного и того же процессора существует несколько видов языка ассемблера. Они совпадают в машинных командах, но различаются набором дополнительных функций (директив и макросов);

* Языки программирования высокого уровня.

Данные виды языков программирования, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания.

Высокоуровневые языки программирования были разработаны для платформенной независимости сути алгоритмов. Зависимость от платформы перекладывается на инструментальные программы — трансляторы, компилирующие текст, написанный на языке высокого уровня, в элементарные машинные команды (инструкции). Поэтому, для каждой платформы разрабатывается платформенно-уникальный транслятор для каждого высокоуровневого языка, например, переводящий текст, написанный на Delphi в элементарные команды микропроцессоров семейства x86.

Так, высокоуровневые языки стремятся не только облегчить решение сложных программных задач, но и упростить портирование программного обеспечения. Использование разнообразных трансляторов и интерпретаторов обеспечивает связь программ, написанных при помощи языков высокого уровня, с различными операционными системами программируемыми устройствами и оборудованием, и, в идеале, не требует модификации исходного кода (текста, написанного на высокоуровневом языке) для любой платформы.

Такого рода оторванность высокоуровневых языков от аппаратной реализации компьютера помимо множества плюсов имеет и минусы. В частности, она не позволяет создавать простые и точные инструкции к используемому оборудованию. Программы, написанные на языках высокого уровня, проще для понимания программистом, но менее эффективны, чем их аналоги, создаваемые при помощи низкоуровневых языков. Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня (язык ассемблера) в ряд современных профессиональных высокоуровневых языков программирования.

Примеры: C++, C#, Java, JavaScript, Python, PHP, Ruby, Perl, Паскаль, Delphi, Лисп. Языкам высокого уровня свойственно умение работать с комплексными структурами данных. В большинстве из них интегрирована поддержка строковых типов, объектов, операций файлового ввода-вывода.

2.2 Отладка и тестирование

При разработке программ наиболее трудоемким является этап отладки и тестирования программ. Цель тестирования, т.е. испытания программы, заключается в выявлении имеющихся в программе ошибок. Цель отладки состоит в выявлении и устранении причин ошибок.

Отладку программы начинают с составления плана тестирования. Такой план должен представлять себе любой программист. Составление плана опирается на понятие об источниках и характере ошибок. Основными источниками ошибок являются недостаточно глубокая проработка математической модели или алгоритма решения задачи; нарушение соответствия между схемой алгоритма или записью его на алгоритмическом языке и программой, записанной на языке программирования; неверное представление исходных данных на программном бланке; невнимательность при наборе программы и исходных данных на клавиатуре устройства ввода.

Нарушение соответствия между детально разработанной записью алгоритма в процессе кодирования программы относится к ошибкам, проходящим вследствие невнимательности программиста. Отключение внимания приводит и ко всем остальным ошибкам, возникающим в процессе подготовки исходных данных и ввода программы в ЭВМ. Ошибки, возникающие вследствие невнимательности, могут иметь непредсказуемые последствия, так как наряду с потерей меток и описаний массивов, дублированием меток, нарушением баланса скобок возможны и такие ошибки, как потеря операторов, замена букв в обозначениях переменных, отсутствие определений начальных значений переменных, нарушение адресации в массивах, сдвиг исходных данных относительно полей значений, определенных спецификациями формата.

Тестирование происходило с помощью программы ThreadScope. Тестированию подвергались как многопоточные инструменты, так и алгоритмы работы.

Алгоритмом называется точное и понятное предписаниe исполнителю совершить последовательность действий, направленных на решение поставленной задачи. Слово «алгоритм» происходит от имени математика Аль Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмом понимали только правила выполнения четырех арифметических действий над числами. В дальнейшем это понятие стали использовать вообще для обозначения последовательности действий, приводящих к решению любой поставленной задачи. Говоря об алгоритме вычислительного процесса, необходимо понимать, что объектами, к которым применялся алгоритм, являются данные. Алгоритм решения вычислительной задачи представляет собой совокупность правил преобразования исходных данных в результатные.

Основными свойствами алгоритма являются:

* детерминированность (определенность). Предполагает получение однозначного результата вычислительного процecca при заданных исходных данных. Благодаря этому свойству процесс выполнения алгоритма носит механический характер;
* результативность. Указывает на наличие таких исходных данных, для которых реализуемый по заданному алгоритму вычислительный процесс должен через конечное число шагов остановиться и выдать искомый результат;
* массовость. Это свойство предполагает, что алгоритм должен быть пригоден для решения всех задач данного типа;
* дискретность. Означает расчлененность определяемого алгоритмом вычислительного процесса на отдельные этапы, возможность выполнения которых исполнителем (компьютером) не вызывает сомнений.

Алгоритм должен быть формализован по некоторым правилам посредством конкретных изобразительных средств. К ним относятся следующие способы записи алгоритмов: словесный, формульно–словесный, графический, язык операторных схем, алгоритмический язык.

Наибольшее распространение благодаря своей наглядности получил графический (блок–схемный) способ записи алгоритмов.

Блок–схемой называется графическое изображение логической структуры алгоритма, в котором каждый этап процесса обработки информации представляется в виде геометрических символов (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций. Перечень символов, их наименование, отображаемые ими функции, форма и размеры определяются ГОСТами.

При всем многообразии алгоритмов решения задач в них можно выделить три основных вида вычислительных процессов:

* линейный;
* ветвящийся;
* циклический.

Линейным называется такой вычислительный процесс, при котором все этапы решения задачи выполняются в естественном порядке следования записи этих этапов.

Ветвящимся называется такой вычислительный процесс, в котором выбор направления обработки информации зависит от исходных или промежуточных данных (от результатов проверки выполнения какого–либо логического условия).

Циклом называется многократно повторяемый участок вычислений. Вычислительный процесс, содержащий один или несколько циклов, называется циклическим. По количеству выполнения циклы делятся на циклы с определенным (заранее заданным) числом повторений и циклы с неопределенным числом повторений. Количество повторений последних зависит от соблюдения некоторого условия, задающего необходимость выполнения цикла. При этом условие может проверяться в начале цикла – тогда речь идет о цикле с предусловием, или в конце – тогда это цикл с постусловием.

Для построения был выбран способ блок–схема, на рисунках 2–4 представлены алгоритмы основных функций приложения.

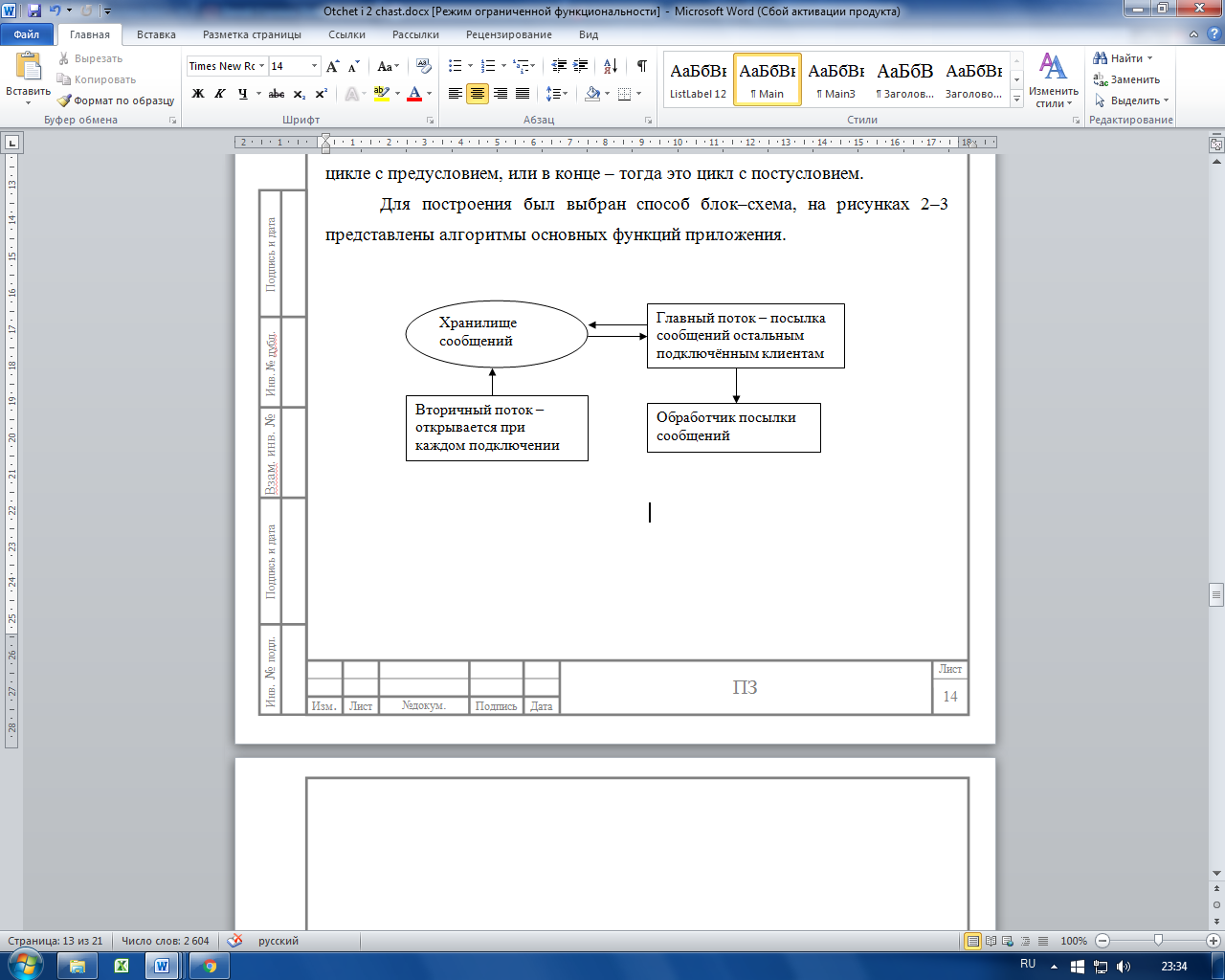


Рисунок 2 ­– работа серверной части

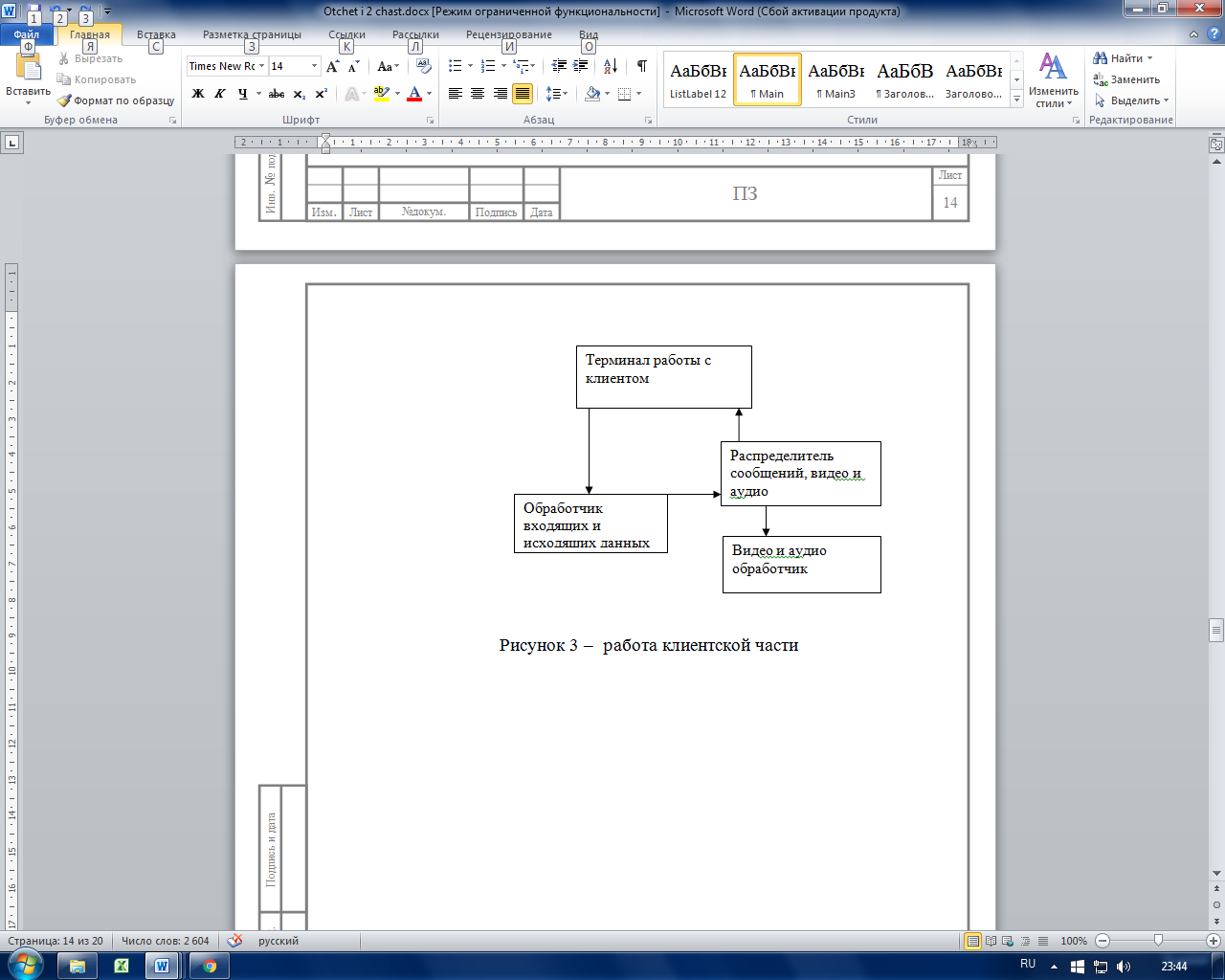


Рисунок 3 ­– работа клиентской части

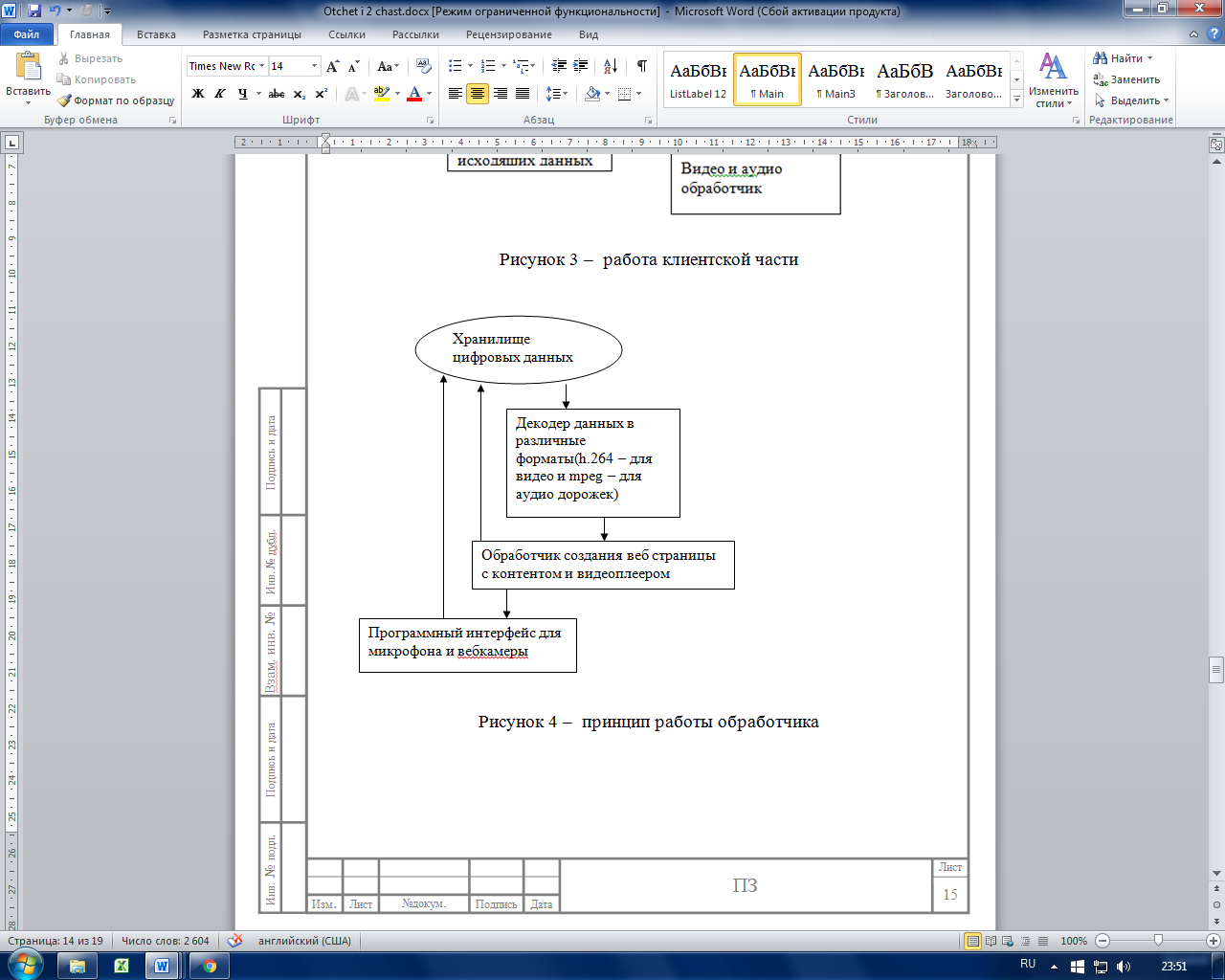


Рисунок 4 ­– принцип работы обработчика

Программа проходила тестовые испытания ресурсозатратности и производительности на предмет утечки памяти и работы многопоточности. Не было выявленно ошибок.

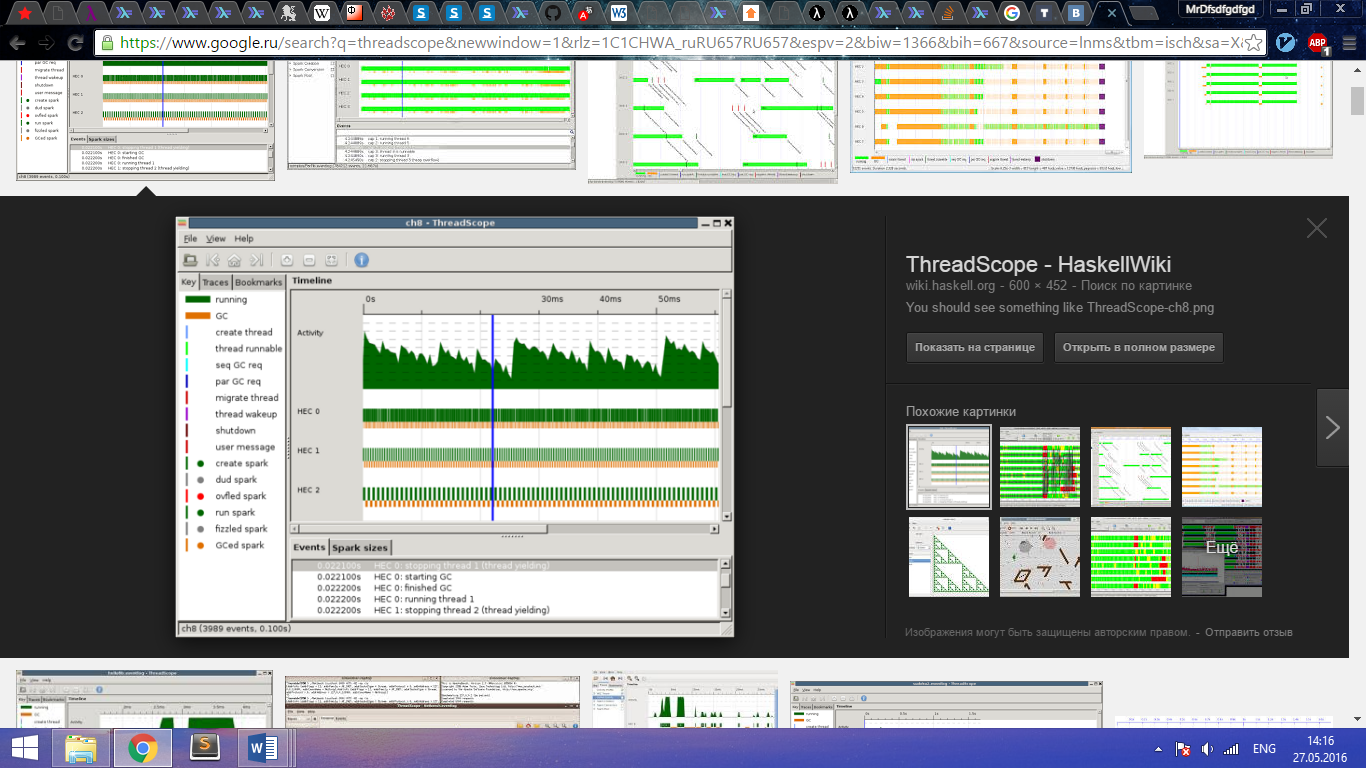


Рисунок 5 ­– тестирование многопоточности

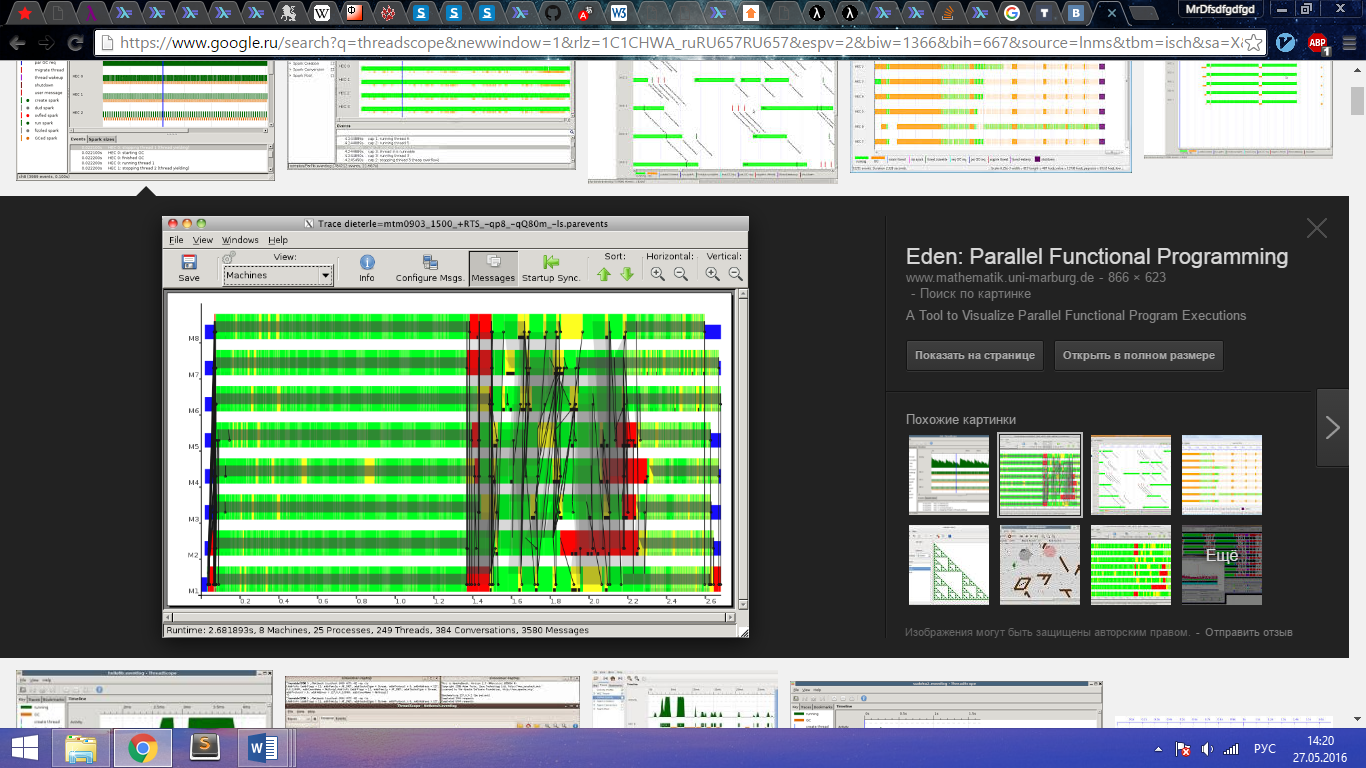


Рисунок 6 – тестирование сборщика мусора

2.4 Инструкция программиста(команды,обозначения)

Кома́нда — это указание компьютерной программе действовать как некий интерпретатор для решения задачи. В более общем случае, команда — это указание некоему интерфейсу командной строки, такому как shell.

В частности, термин команда используется в языках императивного программирования. Эти языки так названы, потому что их операторы, как правило, пишутся наподобие глаголам в повелительном наклонении, используемому во многих естественных языках. Если посмотреть на оператор императивного языка как на предложение естественного языка, то команда, в общем, подобна глаголу.

Список команд:

* !connect [псевдоним хостинга] – подключение к каналу;
* !host [псевдоним] [порт] – создание канала на компьютере. Под портом понимается четырёхзначное число выше 3000, обозначающее номер порта-получателя сообщений;
* !settings [цвет] [цвет] – настройки строки. Первый цвет обозначает цвет строк, а второй – цвет текста;
* !video – открытие браузера и трансляция видео. Функция работает при наличии микрофона, вебкамеры и при передачи кем-либо видеопотока.

Канал связи (англ. channel, data line) — система технических средств и среда распространения сигналов для односторонней передачи данных (информации) от отправителя (источника) к получателю (приёмнику). В случае использования проводной линии связи, средой распространения сигнала может являться оптическое волокно или витая пара. Канал связи является составной частью канала передачи данных. Существует множество видов каналов связи, среди которых наиболее часто выделяют каналы проводной связи (воздушные, кабельные, световодные и др.) и каналы радиосвязи (тропосферные, спутниковые и др.). Такие каналы в свою очередь принято квалифицировать на основе характеристик входного и выходного сигналов, а также по изменению характеристик сигналов в зависимости от таких явлений, происходящих в канале, как замирания и затухание сигналов. == По типу среды распространения каналы связи делятся на проводные, акустические, оптические, инфракрасные и радиоканалы.

2.4.1 Выбор кодека

Ко́дек (англ. codec, от coder/decoder — шифратор/дешифратор — кодировщик/декодировщик или compressor/decompressor) — устройство или программа, способная выполнять преобразование данных или сигнала.

Для хранения, передачи или шифрования потока данных или сигнала его кодируют с помощью кодека, а для просмотра или изменения — декодируют. Кодеки часто используются при цифровой обработке видео и звука.

В кодеках могут использоваться два вида сжатия данных: сжатие с потерями и сжатие без потерь.

Многие аудио- и видеокодеки используют сжатие с потерями, что существенно уменьшает объём данных для хранения или передачи, но приводит к ухудшению качества звука или видео при воспроизведении. Аудиокодеки можно настроить таким образом, что человеческий слух не замечает ухудшения качества звука.

В случае, если потеря качества неприемлема, то используются кодеки со сжатием без потерь. Например, сжатие без потерь используется, если планируется последующее редактирование файла, так как сжатие с потерями будет быстро ухудшать качество при каждом последующем сохранении результата.

Выбор кодека осуществлялся по следующим критериям:

* возможность ухудшать качество дорожки;
* видео с переменной кадровой частотой;
* аудио с переменным битрейтом;
* поддержка html5 видеоплеером;

Также выбор основывался на данной таблице сравнения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Медиаконтейнер | Владелец стандарта | Поддержка B-frame | Аудио с переменным битрейтом | Видео с переменной кадровой частотой |
| 3GP | 3GPP | Да | Да | Да |
| [Advanced Systems Format](https://ru.wikipedia.org/wiki/Advanced_Systems_Format)(.asf, .wma, .wmv) | Microsoft | Да | Да | Да |
| AVI | Microsoft | Да | Да | Нет |
| DivX Media Format (.divx) | DivX | Да | Да | Да |
| EVO | MPEG | Неизвестно | Неизвестно | Неизвестно |
| Flash Video (.flv) | Adobe Systems | Неизвестно | Неизвестно | Неизвестно |
| MPEG-2 TS Transport Stream (.ts) | MPEG | Да | Да | Да |

Был выбран кодек MPEG4 , состоящий из 2 стандартов: видео – h264, аудио – mp3.

2.4.2 Опытная эксплуатация и внедрение проекта

Под опытной эксплуатацией следует понимать эксплуатацию системы заказчиком параллельно с основной информационной системой, которая находится в промышленной эксплуатации. В ходе опытной эксплуатации в систему может вводиться заведомо некорректная информация, поэтому результаты ее работы в опытной эксплуатации не используются для принятия бизнес–решений.

В ходе внедрения необходимо строго придерживаться утвержденных плана и графика, игнорируя возможность добавления в систему новых необязательных требований и возможностей, иначе реализация проекта затянется до бесконечности.

2.5 Инструкция пользователя

Программа начинается с открытия ярлыка Multichat.exe. Его расположение зависит от пользовательского корня файлов.

В начале происходит знакомство с пользователем. Приложение спрашивает псевдоним пользователя, который будет использоваться в программе.

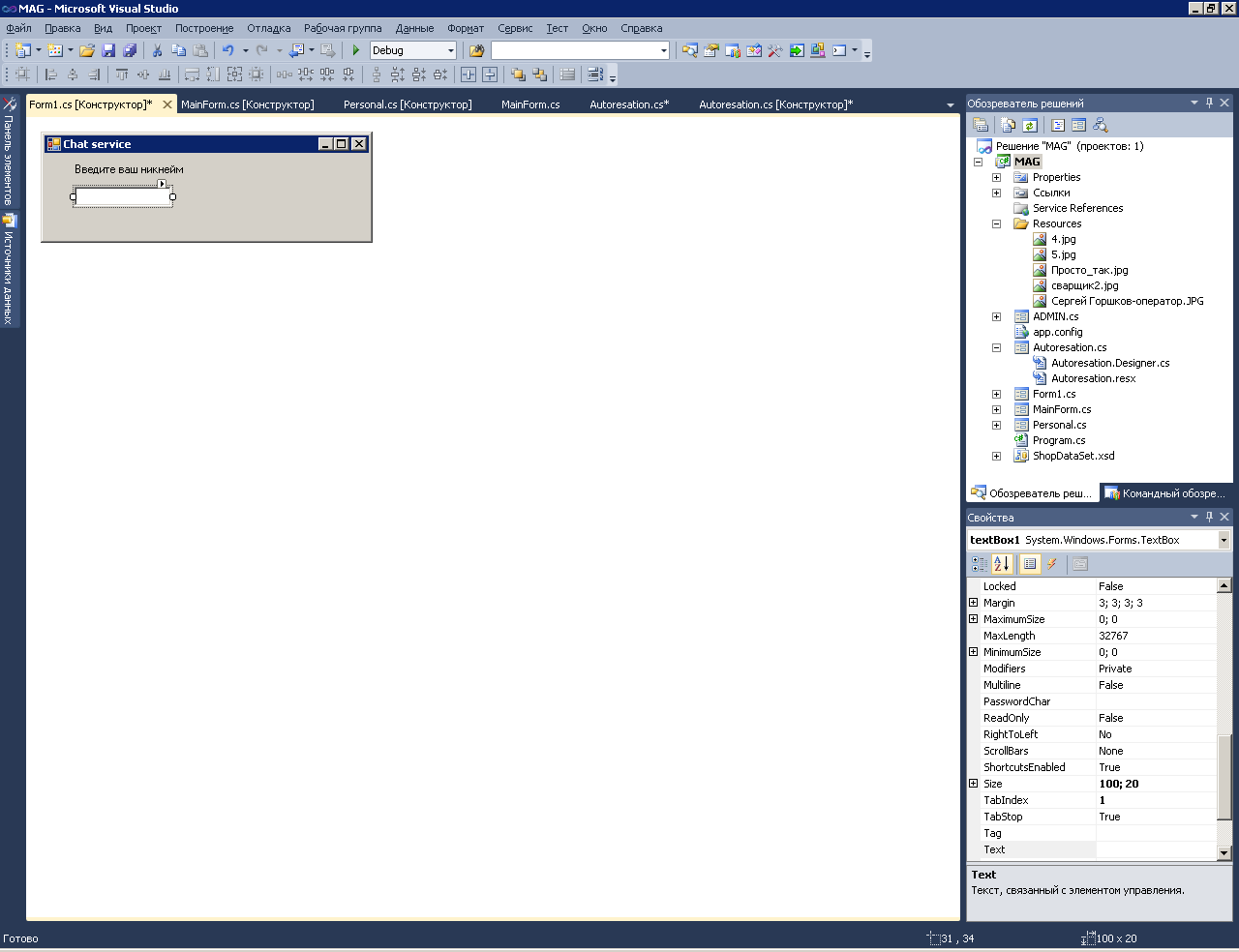


Рисунок 5 ­– приветствие

Затем открывается меню выбора команды. От команды зависит дальнейшая работа приложения.

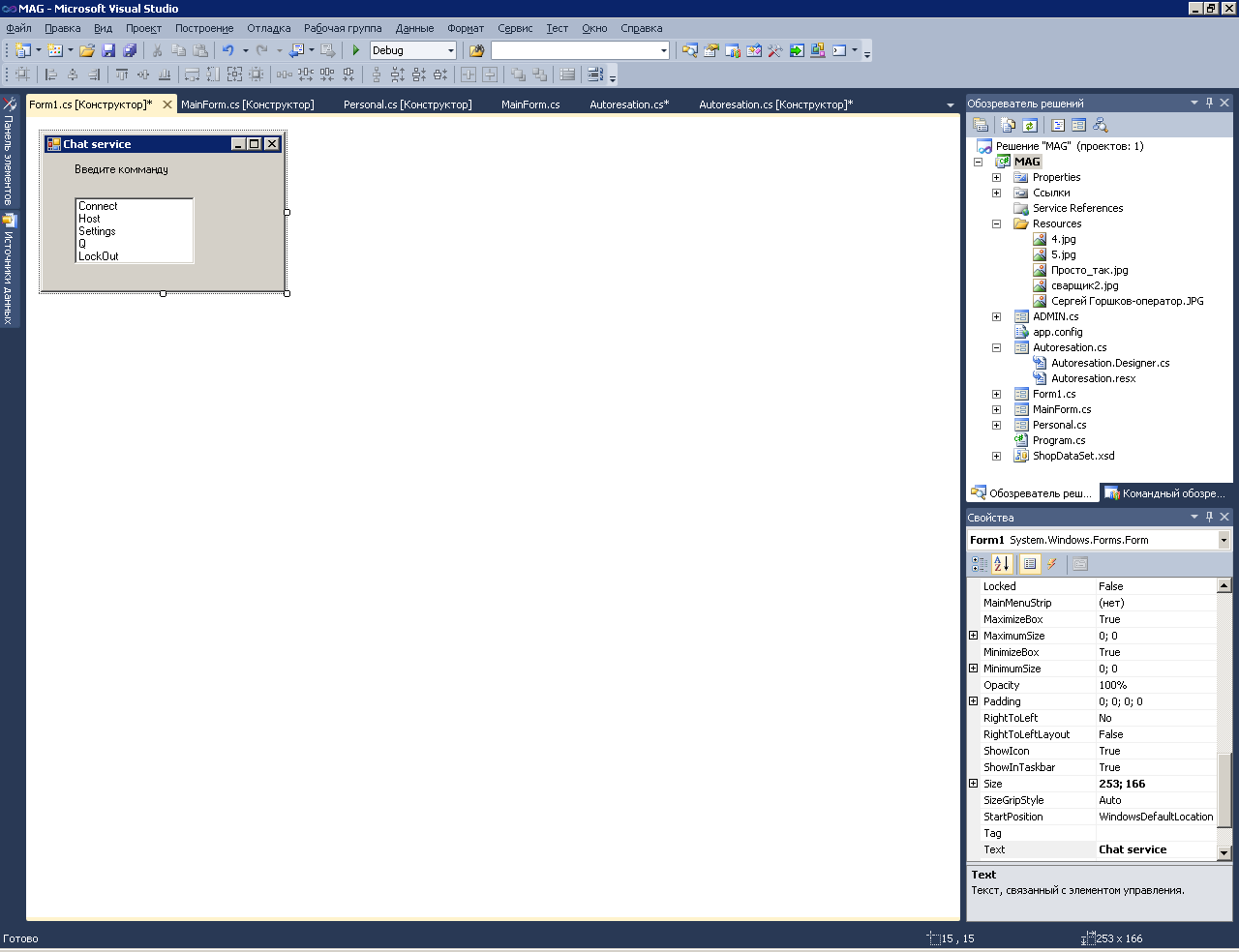


Рисунок 6 ­– выбор комманды

При выборе соответствующей команды определяется соответствующая работа приложения. Если connect, то происходит подключение к существующему каналу, host создаёт свой канал, к которому происходит подключение, settings выявляет вид приложения.

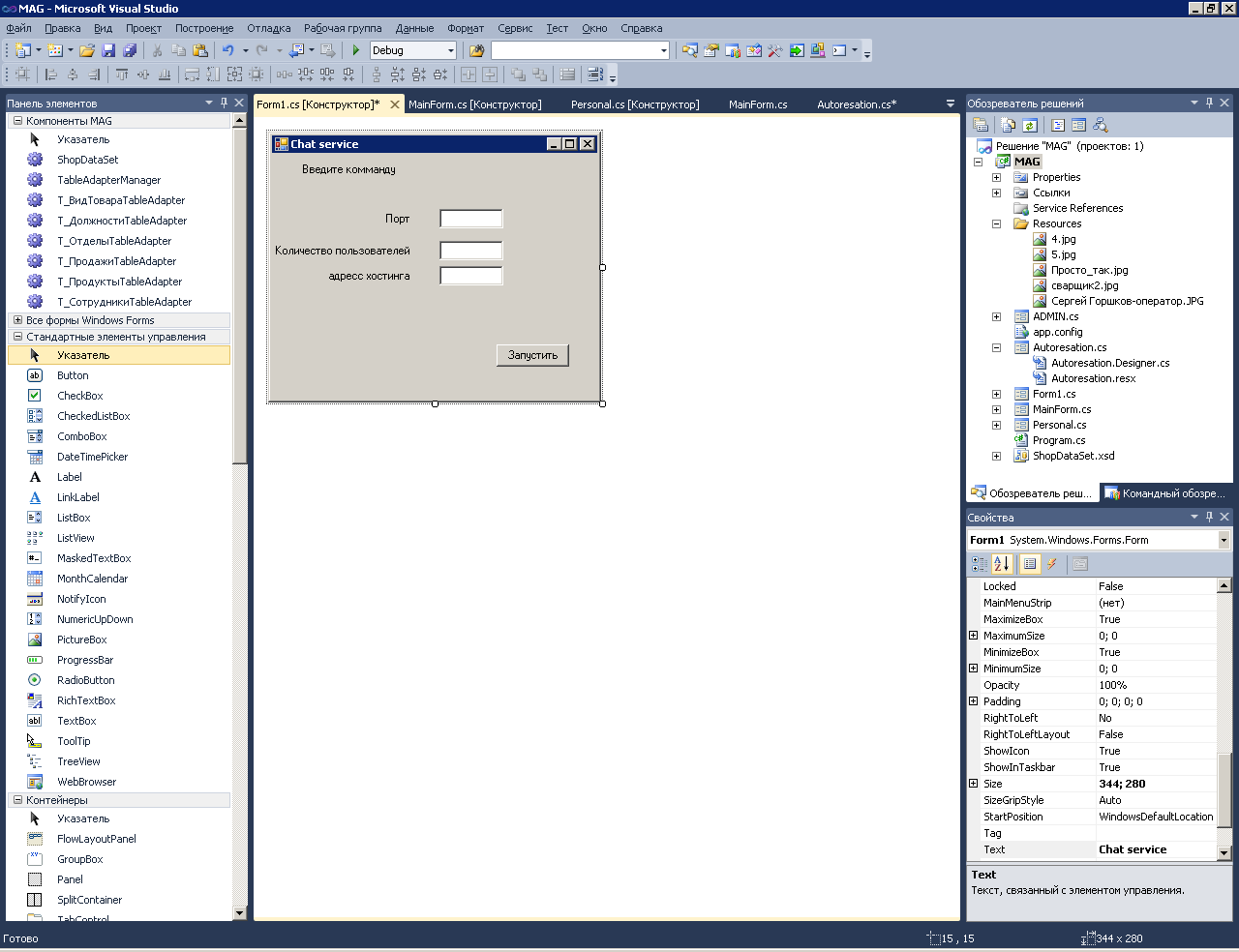


Рисунок 7 ­– создание канала

После выборе функции connect происходит подключение к некоторому каналу: либо локальному, либо интернет. После подключения и происходит общение между посетителями канала.

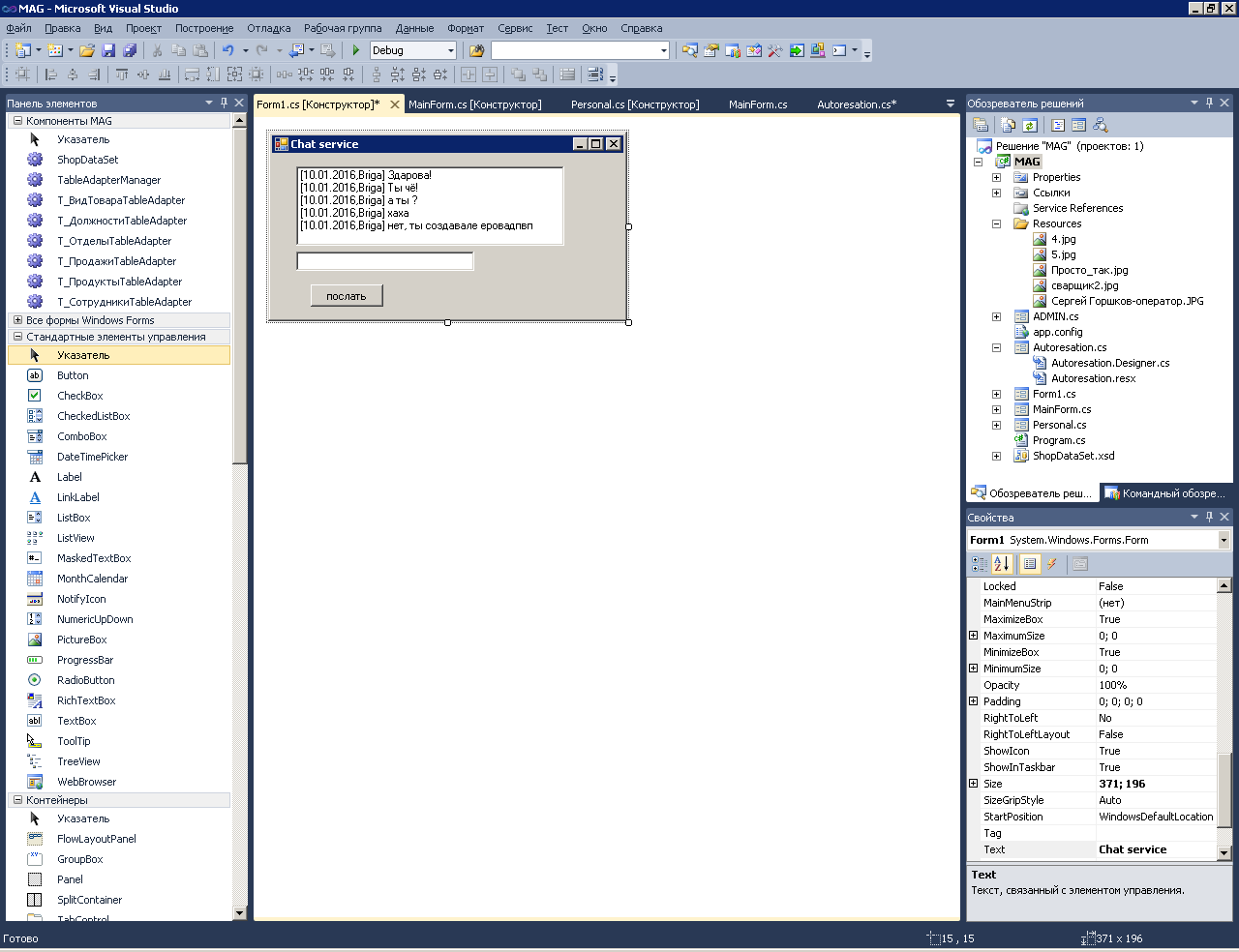


Рисунок 8 ­– общение

# Заключение

В результате работы было создан сервис для общения, имеющая возможность работы в локальной сети.

В ходе работы были решены следующие задачи:

* изучена предметная область;
* разработана архитектура программы;
* выбрано программное обеспечение;
* написан стабильный код;
* разработан пакет программ для обработки видео и аудио;
* протестировано приложение;
* подготовлена соответствующая документация.

Система разработана на языках Haskell, JavaScript, HTML, CSS. Использована блочная вёрстка.