ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

28366

регистрационный номер

Секция: **Системы обработки информации**

название секции

Сайт для эффективного обмена данными между устройствами

название работы

Автор: **Карпушин Георгий Михайлович**

фамилия, имя, отчество

ГБОУ школа №2086, 11 “А” класс

наименование учебного заведения, класс

Москва - 2024

Аннотация

Ежедневно многие люди из разных сфер деятельности сталкиваются с потребностью в передаче информации между устройствами. Проведя исследование этой проблемы, я пришёл к выводу, что необходимо разработать и внедрить универсальный кроссплатформенный веб-инструмент для эффективной передачи данных.

В работе рассматриваются исследование, основанное на эмпирическом методе интервью, доказывающее актуальность проекта, методы реализации программ для сайта, учитывая тенденции современных технологий и потребностей потенциальных пользователей, и результаты внедрения решения с их последующим анализом.

Главным результатом работы над проектом стало веб-решение. Оно расположено по ссылке https://fastchat.space. Основываясь на данных из Яндекс Метрики, подключенной к сервису, выясним, что ежемесячно сайт посещают в среднем 87 человек. Из этого следует, что гипотеза о необходимости подобного сервиса была верной.

Содержание

Сайт для эффективного обмена данными между устройствами 1
Аннотация 2
Содержание 3
Введение 4
Основная часть 6
Цель 6
Задачи 6
Гипотеза 6
Теоретическая часть 7
Выявление актуальности и анализ существующих решений 7
Прогноз наиболее востребованного функционала 7
Практическая часть 9
Выбор языков программирования 9
Выбор и подключение базы данных к проекту 10
Выбор технологии и разработка схемы клиенто-серверного общения 12
Автоматическая работа сервера 15
Автоматическая работа клиента 16
Минимальные системные требования 16
Необходимые элементы для корректной работы сервиса на VPS/VDS 17
Результаты работы 17
Заключение 19
Список используемой источников 20
Приложение 21
Блок-схема создания чата 21
Блок-схема обработки текстовых и бинарных сообщений 22
Максимальное количество запросов в минуту 23
Максимальное количество запросов в секунду 23
Онлайн сайта 23

Введение

Ежедневно многие люди сталкиваются с потребностью в передаче информации между устройствами. Приведу несколько жизненных примеров, подтолкнувших меня начать работу над проектом:

1. В моей школе на уроках информатики и программирования используется тестирующая система, для авторизации в которой необходима уникальная длинная ссылка. Ранее, чтобы передать эту ссылку, мы использовали почту, но для этого требовалось осуществить вход в аккаунт на компьютере, к которому имели доступ все учащиеся школы. Если забыть выйти из личного аккаунта после урока, другие учащиеся могли бы воспользоваться этой оплошностью и получить доступ не только к письмам, но и к другим аккаунтам, привязанным к этой почте, где хранятся ещё более важные персональные данные.
2. Помимо длинной ссылки, периодически мне и моим одноклассникам было необходимо писать код или делать презентации дома и приносить его на уроки в качестве домашнего задания. Единственным реалистичным способом является приносить задание на флешке, что не очень удобно, ведь помимо редких проблем с совместимостью или вероятности заразить флешку вирусом, носитель можно банально забыть при выходе из дома или при уходе из компьютерного класса.
3. Мне также известна проблема, с которой сталкиваются преподаватели при раздаче электронного материала для работы на класс. Самым распространенным методом является копирование необходимых файлов на каждый компьютер при помощи флешки. Однако, этот процесс занимает много времени. Другим способом является рассылка архива с необходимыми файлами на почту. Об опасностях такого способа я рассказал в предыдущем пункте.
4. В ходе работы над учебным проектом в рамках урока программирования нам потребовалось объединиться в группы и распределить обязанности по

написанию программы. Однако возникла проблема: как объединить все части кода в одну программу? Мы решили передавать код друг другу по электронной почте. Однако, кроме опасностей входа в почту на общественном компьютере, описанных выше, это было крайне неудобно и заняло много времени. Кроме того, возникали несостыковки между модулями, так как мы не имели доступа к функциям, написанным другими участниками, во время программирования собственных.

Проанализировав эти ситуации и проведя устный опрос класса методом интервью, методом индукции я пришёл к выводу, что проблема актуально, и, пользуясь гипотетическим методом, предположил, что эту проблему можно решить, если передавать данные через специальный сервис, оформленный в виде кроссплатформенных временных каналов общего доступа с возможностью дополнения данных в реальном времени без регистрации или, проще говоря, чатов, которые будут фактически анонимны и лишены персональных данных пользователя. Это не только поможет предотвратить проблемы с конфиденциальностью, но и даст возможность кратковременно хранить там файлы, раздавать их на большую аудиторию и иметь доступ с абсолютно разных устройств без предварительной подготовки. Наиболее логичным и удобным способом реализовать такой проект было создание сайта. Я назвал свой проект Fast Chat.

Основная часть

Цель

Разработать эффективный способ обмена информацией между устройствами, позволяющий пользователям передавать различные типы данных кроссплатформенно без необходимости регистрации.

Задачи

- Исследовать актуальность проекта
- Проанализировать существующие решения
- Спрогнозировать наиболее востребованные функции
- Разработать схему клиенто-серверного общения
- Разработать алгоритм обработки, хранения и очистки пользовательских данных
- Выбрать технологии для реализации
- Провести тестирование

Гипотеза

Люди нуждаются в удобном способе передачи информации с одного устройства на другое эффективно и с возможностью получения информации через промежуток времени после отправки.

Теоретическая часть

Выявление актуальности и анализ существующих решений

Для начала я решил выяснить, насколько моя идея актуальна. Я провел устный опрос среди участников моего класса, учителей и знакомых. Всего в опросе принял участие 41 человек. Проанализировав полученные ответы, я выяснил, что 87.8% респондентов считают проблему актуальной и хотели бы увидеть предложенное мной решение в реальности. Кроме того, я задал респондентам вопрос о том, чем они пользуются в подобных вышеприведённым ситуациях и какие недостатки и преимущества они видят в этом и составил таблицу по полученным результатам:

	Fast Chat	Bluetooth	Приложения-передатчики в локальной сети	Файлообменники онлайн	Мессенджеры, социальные сети, почта
Процент пользователей	87,8%	61%	12,2%	9,8%	100%
Можно передавать различные типы информации	+	-	-	-	+
Можно подключиться сразу с нескольких устройств	+	-	-	+	-
Хранится некоторое время	+	-	-	+	+
Не нужно устанавливать дополнительное ПО	+	+	-	+	-
Не нужно регистрироваться на публичном устройстве	+	+	-	+	-
Кроссплатформенность	+	+	-	+	+

Таблица 1. Сравнения аналогов, используемых респондентами для решения схожих с вышеприведёнными проблем

Прогноз наиболее востребованного функционала

Методом индукции, основываясь на данных опроса, логике, дополнительных опросах и приведённых выше ситуациях, я пришёл к выводу, что проекту

необходимы следующие базовые функции, представленные в порядке убывания актуальности:

- Возможность передавать текст и ссылки с возможностью прямого перехода в явном виде и файлы разных расширений, а также дополнять данные в чате в реальном времени. О необходимости такой функционала заявили все респонденты.
- О необходимости быстрого создания и входа в чат также заявили все респонденты. Для быстрого создания чата я решил отказаться от регистрации в пользу уникальных идентификаторов для каждого чата. Для быстрого входа в чаты мною были придуманы два способа: с помощью ручного ввода короткого уникального идентификатора чата на главной странице или по QR кода, генерируемого в каждом чате. Я решил, что допустимыми идентификаторами должны быть строки, содержащие только регистронезависимые буквы латинского алфавита, цифры, всего более 4 и менее 11 символов в длину. А стандартным идентификатором я выбрал допустимый идентификатор длинной в 5 символов. Всего может быть создано до 60.5 миллионов чатов со стандартным идентификатором, чего с головой хватит для моего проекта.
- Возможность установить пароль в двух режимах: полной и частичной блокировки. Первый режим позволяет ограничить доступ ко всему содержанию чата и может быть полезен для передачи конфиденциальной информации. Согласно проведенному опросу, каждый второй человек сталкивается с такой необходимостью. Второй режим предназначен для распространения данных, переданных в чат, на большую аудиторию, при этом исключается возможность совместного редактирования содержимого чата. Все опрошенные учителя, согласно данным опроса, выразили желание видеть такую возможность.

- Возможность установки так называемой “тёмной темы” чата. Согласно опросу, 17,1% респондентов принципиально не будут пользоваться сервисом без этой функции.
- Возможность установить имя отправителя, тему сообщения или иной комментарий для сообщения. Я решил объединить эту потребность в одну функцию и назвать её тэгами. Согласно опросу, 7% респондентов крайне необходима такая функция. Я также посчитал, что это будет полезно в случаях использования чата для работы над проектом несколькими людьми, для отправки ответов учителю учениками, для возможности распределения информации в чате по темам и других подобных случаях. Именно поэтому я решил внедрить эту функцию сразу после реализации основных. В дальнейшем, в ходе первого же открытого тестирования в школе, когда в один чат зашли 17 человек, необходимость функции ещё раз подтвердилась.

Практическая часть

Выбор языков программирования

В качестве языка программирования для клиента был выбран JavaScript. Этот язык поддерживается абсолютным большинством браузеров, хорошо документирован и имеет активное сообщество пользователей (свыше 314 тысяч тем на stackoverflow.com). Важно отметить, что я уже имел большой опыт работы с ним, что тоже стало весомым аргументом в его пользу. Теоретической альтернативой был WebAssembly, но против его использования можно привести следующие аргументы: сложность разработки на низкоуровневом языке, необходимость в компиляции кода, поддержка со стороны значительно меньшего количества браузеров (согласно данным caniuse.com) и, наконец, мое полное незнакомство с синтаксисом данного языка.

При выборе языка программирования для создания сервера я рассматривал несколько популярных и современных вариантов, с каждым из которых я был знаком как минимум на базовом уровне: Python, Node JS и GoLang. Основными

требованиями к языку программирования для сервера считаю многопоточность и быстродействие:

- Многопоточность. GoLang, на фоне всех основных языков, выделяется самой простой реализацией многопоточности благодаря так называемым го-рутинам. В случае Node JS многопоточность больше похожа на симуляцию из-за метода Worker Threads, чем на реальную многозадачность. Именно поэтому безусловным победителем в этой категории я определил GoLang.
- Скорость работы. Согласно исследованию Туана Нгуена на сайте golinuxcloud.com, GoLang можно назвать самым быстрым языком, поэтому и в этой категории я выбрал его.

Именно поэтому серверная часть написана на GoLang.

Выбор и подключение базы данных к проекту

База данных Cassandra от Apache является “NoSQL базой данных с открытым исходным кодом, которой доверяют тысячи компаний за масштабируемость и высокую доступность без ущерба для производительности”, - написано на официальном сайте разработчика. Она отлично подходит для хранения файлов и текстовых сообщений. Кроме того, NoSQL база данных позволяет отказаться от приведения типов к текстовому формату, что заметно ускорит написание и выполнение кода. Кроме того, в отличие от некоторых других баз данных, Cassandra легко масштабируется, достаточно производительна благодаря шардированию и репликации и поддерживается GoLang. Также я составил схему базы данных, она приведена ниже (Рис.1):

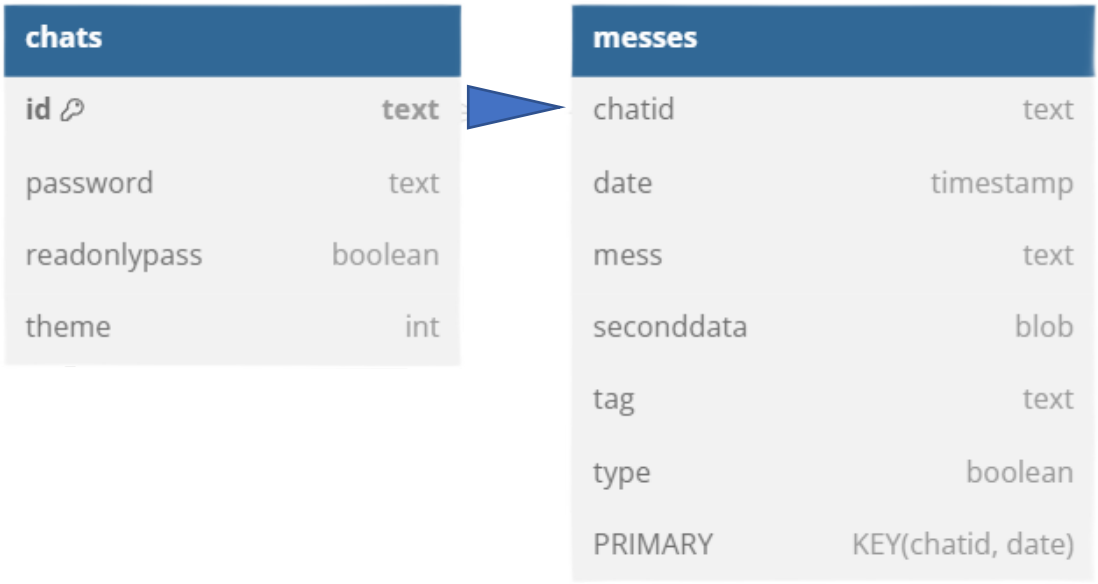


Рис 1 Схема базы данных

Описание таблиц:

Таблица chats хранит все чаты:

Столбец id - уникальный идентификатор чата

Столбец password - пароль

Столбец readonlypass - режим работы пароля

Столбец theme - номер выбранной в чате темы

Таблица messes хранит все сообщения:

Столбец chatid - идентификатор чата

Столбец date - время и дата отправки сообщения

Столбец mess - сообщение или название файла

Столбец seconddata - бинарные данные файла

Столбец tag - тэг сообщения

Столбец type - обозначает тип сообщения: если значение false, значит сообщение - обычный текст, если true, значит в текстовом поле - название файла, а в seconddata бинарные данные файла.

Выбор технологии и разработка схемы клиенто-серверного общения

Существует несколько распространённых технологий клиенто-серверного общения, вот таблица сравнения:

Особенность	**AJAX**	**Fetch API**	**WebSocket**	**Server-Sent Events (SSE)**	**WebRTC**
Тип связи	Request-Response	Request-Response	Full-Duplex	Server to Client	Peer-to-Peer
Поддерживаемые типы танных	Текст и XML	Текст и JSON	Текст и бинарные данные	Текст	Бинарные данные
Возможность работы в реальном времени	Нет	Нет	Есть	Есть	Есть

Таблица 2. сравнения различных технологий клиент-сверверного общения

Учитывая необходимость возможности дополнения чатов данными в реальном времени и двухстороннего доступа к каналу, можно отбросить использование AJAX, Fetch API и SSE. Также, учитывая ограничения WebRTC в передаче текста или JSON/XML, можно отказаться от его использования для экономии времени. Именно поэтому было решено использовать протокол WebSocket, который наиболее подходит для достижения поставленных целей.

WebSocket обеспечивает возможность установления постоянного соединения между клиентом и сервером, позволяя обмениваться данными в режиме реального времени с обеих сторон. Это позволяет достичь мгновенной доставки сообщений и обновления чата без необходимости постоянного обновления страницы.

При разработке алгоритма общения между клиентом и сервером, я разделил потенциально передаваемые данные на две ключевые группы с логической точки зрения: инструкции и команды, текст (включая названия файлов), ссылки и бинарные данные файлов. Отметим, что особых различий между текстом и ссылками нет, поэтому их можно не разделять. Поэтому я решил, что и текст, и ссылки, и инструкции могут быть переданы в формате JSON в однообразной структуре сообщений. Значит остаётся две группы - JSON в текстовом формате и

бинарные данные файлов. Обе эти группы уже обособлены на уровне протокола, поэтому не нужно придумывать дополнительный алгоритм по их разделению.

Как я и написал, в JSON формате также будут передаваться инструкции от сервера клиенту и наоборот: налаживание подключения, установка пароля, переход в другой чат или изменение его идентификатора, смена темы чата, запрос клиентом файла или сообщений и так далее. Было решено создать коды для обработки и единую структуру сообщений, чтобы было легче обрабатывать их на GoLang. Структура сообщений, созданная с помощью синтаксиса Go следующая:

```
type Message struct {
    Code int    `json:"code"`
    Text string `json:"text"`
    Tag  string `json:"tag"`
    Time string `json:"time"`
}
```

структура сообщения

Теперь мне было необходимо разработать коды и способ их обработки. На текущий момент существует 8 клиенто-серверных и 7 серверо-клиентских кодов. Стоит сразу сказать, что при полной блокировке чата паролем клиенту доступны только 0 и 5 код, а при частичной - 0, 4, 5, 6. Описание всех кодов:

- Клиенто-серверные коды предназначены для оповещения сервера о соответствующих действиях клиента:

  0. Запрос на подключение. Может содержать или не содержать идентификатор чата. Схема обработки кода сервером изображена в приложении в блок-схеме создания чата.

  1, 2. Эти коды отвечают за обычное текстовое сообщение и за название последующего за этим сообщением файла. Схема обработки кода сервером изображена в приложении в блок-схеме обработки текстовых и бинарных сообщений.

  3. Установка нового пароля. Проверяется, что режим пароля установлен на полную блокировку или пользователь авторизован в качестве администратора чата. В текстовом поле сообщения содержится новый пароль, а в тэге - режим работы. Также, если новый пароль пустой - он

снимается с чата полностью. Результат сохраняется в базу данных chats в соответствующие столбцы.

4. Запрос старых сообщений. Для оптимизации работы я решил, что буду отправлять клиенту только 50 последних сообщений. Когда клиент будет доходить до самого раннего сообщения, будет сделан запрос, содержащий в поле time дату и время этого сообщения. В ответ сервер добавит клиенту ещё 50 сообщений в хронологическом порядке.

5. Когда на чате установлен, клиент сможет ввести его с помощью этого когда. Если хеши паролей при проверке сервером совпадут - клиент будет авторизован в чате как администратор.

6. Запрос файла. Когда клиенту отправляются сообщения, в них содержатся лишь текст, оригинальное время отправки и тэг. Если в чате есть файл, то его сообщение не содержит бинарных данных для оптимизации скорости доставки и производительности сайта. В качестве текста сообщения формируется кнопка в виде ссылке, при нажатии на которую клиент отправит серверу запрос на файл, содержащий имя, дату и время отправки. Сервер вернёт клиенту имя файла и бинарные данные двумя сообщения в соответствующем порядке.

7. Оповещение сервера о смене темы в чате. В поле text будет содержаться номер темы, на данный момент их всего 2. После обработки всем клиентам онлайн будет отправлена текущая тема, а также произойдёт обновление соответствующего столбца в таблице chats.

- Серверо-клиентские коды являются своеобразными командами:

  0. Принудительная перезагрузка страницы с возможностью поменять идентификатор чата. Используется только при смене пароля.

  1. Добавить сообщение в чат в хронологическом порядке. Text содержит сообщение, tag - имя, time - время отправки для добавления в хронологическом порядке.

2. Отправка имени файла в качестве сообщения для расположения его в чате в хронологическом порядке. Text содержит имя файла, tag - тэг сообщения, time - время отправки для добавления в хронологическом порядке.

3. Смена идентификатора без перезагрузки страницы. Поле text содержит новый идентификатор чата, а tag – true или false в зависимости от режима пароля. True, если режим работы с частичным доступом, иначе false.

5. Запрос пароля от клиента. Посылается при заходе в чат с полной блокировкой или при нажатии на кнопку авторизации в чате с частичной блокировкой. Поле tag содержит true или false в зависимости от режима пароля. Если false - клиент не сможет закрыть диалоговое окно без ввода верного пароля. Иначе у клиента будет такая возможность, если он нажмёт на крест справа сверху.

6. Назначение имени файла для последующих после сообщения бинарных данных. Поле text содержит имя файла.

7. Назначение темы чата. Поле text содержит номер темы.

Автоматическая работа сервера

Сервер создаёт защищённое соединение на 443 порту, а все незащищённые входящие запросы на 80-й порт принудительно переводит на защищённую версию, принимает запросы на подключение по протоколу Web Socket по адресу speaker и обрабатывает их. Также, в 4:00 по локальному времени сервера производится очистка пустых и старых чатов (в которых не было сообщений более 7 дней) с помощью утилиты cron и дополнительной подпрограммы cleaner, а при перезагрузке системы сервер будет повторно включён в течение 45 секунд после активации сервера.

Автоматическая работа клиента

Изучив необходимый материал, я добавил на сайт счётчик Яндекс Метрики, чтобы иметь доступ к детальной статистике пользования сайтом. Кроме того, я изучил документацию поисковых систем Яндекс и Google об индексировании и успешно сверстал подходящую для обходов роботами страницу, что помогло добиться среднего ежемесячного онлайна в 87 человек [Приложение: онлайн сайта].

Если соединение с сервером пропало или не было установлено - сайт автоматически пытается переподключиться, пока не получится.

Если клиент ввёл несуществующий идентификатор - создаётся новый пустой чат с введённым идентификатором.

Если клиент ввёл зарезервированный идентификатор undefined, то для него создаётся новый пустой чат со сгенерированным идентификатором из 5 символов.

Если клиент ввел идентификатор, не соответствующий требованиям или зарезервированный идентификатор EOF или зарезервированный идентификатор chats, его перебрасывает на https://fastchat.space/chat/EOF - зарезервированный чат для отображения ошибки ввода идентификатора клиентом.

Сайт автоматически создаёт QR код для своей страницы в левом нижнем углу страницы.

Минимальные системные требования

Поскольку бюджет моего проекта был ограничен, я пытался создать систему, которая будет работать с минимальным финансированием. В итоге я нашёл виртуальную машину, имеющую 1 ГБ оперативной памяти, 1 ядро процессора и 14 ГБ места на жёстком диске, что должно хватить для развёртывания прототипа. Мне удалось правильно настроить базу данных, чьи минимальные системные требования гораздо выше характеристик VDS, благодаря чему стало возможным поддерживать стабильную работу сервиса довольно продолжительное время до забивания так называемой кучи (heap limit), при которой будет срабатывать oom-killer. С помощью Яндекс Метрики, подключенной к сайту, я выяснил, что сайт может выдерживать

как минимум до 751 запроса страницы в минуту [Приложение: Максимальное количество запросов в минуту] и с нагрузкой до 9 запросов в секунду [Приложение: Максимальное количество запросов в секунду].

Необходимые элементы для корректной работы сервиса на VPS/VDS

- Установленная Ubuntu 22.04 (Операционная система виртуальной машины)
- Установленный GoLang версия 1.19 (Для работы сервера и возможности редактировать и собирать исходные файлы сервера)
- Установленная Java версия 1.8.0_352 (Для корректной работы Apache Cassandra)
- Установленный Python версия 3.10.6 (Для корректной работы Apache Cassandra)
- Установленный Apache Cassandra версия 4.1.0 (База данных)
- Установленный cron и nohup (Для запуска сервера и очистки в фоновом автоматическом режиме)

Для корректной работы клиентской части сайта необходим браузер, который, с учётом современных тенденций технологий и запросов пользователей, поддерживает все необходимые функции. Например, браузеры на Chromium удовлетворяют этим условиям.

Результаты работы

В результате получился довольно удобный, гибкий и понятный инструмент по передаче, распространению и хранению различных данных без регистрации.

Пользовательский интерфейс предоставляет обычному человеку доступ к функциям и возможностям программ, осуществляющих передачу данных между устройствами. Он был разработан интуитивно понятным с расчетом на действия “в один клик”. Благодаря нескольким открытым тестам были выявлены и исправлены его основные проблемы.

Чтобы создать чат, нужно нажать на кнопку "Создать Чат!" на главной странице или ввести желаемый идентификатор в поле "ID чата (Если есть)" на главной странице и нажать на кнопку "Перейти в Чат!".

Чтобы войти в чат по идентификатору, нужно ввести его в поле "ID чата (Если есть)" на главной странице и нажать на кнопку "Перейти в Чат!" или воспользоваться QR кодом, отобразив его по нажатию на кнопку сверху справа страницы чата. При желании пользователя ссылку на чат можно найти в браузерной строке.

В чате наверху слева - название проекта, справа - допустимый идентификатор после “ID: “ и кнопка отображения QR кода.

В самом центре страницы чата - невидимое поле сообщений (в нём видны только исходящие и входящие сообщения): Бóльшим шрифтом - сообщение, меньшим снизу и справа - дата отправки сообщения по локальному времени.

Снизу по центру - поле ввода сообщения, правее - кнопка прикрепления файлов, далее по направлению - отправка введённого сообщения и всех прикреплённых файлов по очереди. Также для отправки сообщений можно нажать сочетание клавиш Enter на клавиатуре. Слева от поля ввода - меню управления чатом, которое содержит в себе настройки пароля, тэга сообщений и темы чата.

Заключение

Созданный сервис обладает рядом конкурентных функций и удовлетворяет всем целям проекта. Fast Chat способен создавать временные чаты с допустимым идентификатором, сохранять и распространять среди участников чата сообщения и файлы не превышающие 10 мегабайт в реальном времени, чаты удаляются в течение 8-ми дней после отправки последнего сообщения (пустые чаты тоже удаляются). Сайт находится по адресу https://fastchat/space и защищён с помощью ssl сертификата и, в основном, https и wss протоколов. Сайтом ежемесячно пользуется в среднем 87 человек, согласно данным Яндекс Метрики [Приложение: онлайн сервиса]. Видео работы проекта - https://youtu.be/_dv4H_FFSD0 (рис 3).

Рис. 3 QR-код на видео-демонстрацию работы сайта

Рис.4 QR-код на сайт проекта

Список используемой источников

Официальная спецификация Bluetooh – URL: https://www.bluetooth.com/specifications/ (дата обращения: 30.09.2022) Текст: электронный

Сравнение Python и Go – URL: https://hackr.io/blog/golang-vs-python (Дата обращения: 2.10.2022) Текст: электронный

Сравнение C++ и Go – URL: https://blog.boot.dev/golang/go-vs-c-plus-plus-golang/ (Дата обращения: 3.10.2022) Текст: электронный

Golang Performance Comparison | Why is GO Fast? – URL: https://caniuse.com/wasm (дата обращения: 3.10.2022) Текст: электронный

Поддержка Webassambly – URL: https://www.golinuxcloud.com/golang-performance/ (дата обращения: 3.10.2022) Текст: электронный

Официальная документация Go – URL: https://go.dev/ (дата обращения: на протяжении всего проекта) Текст: электронный

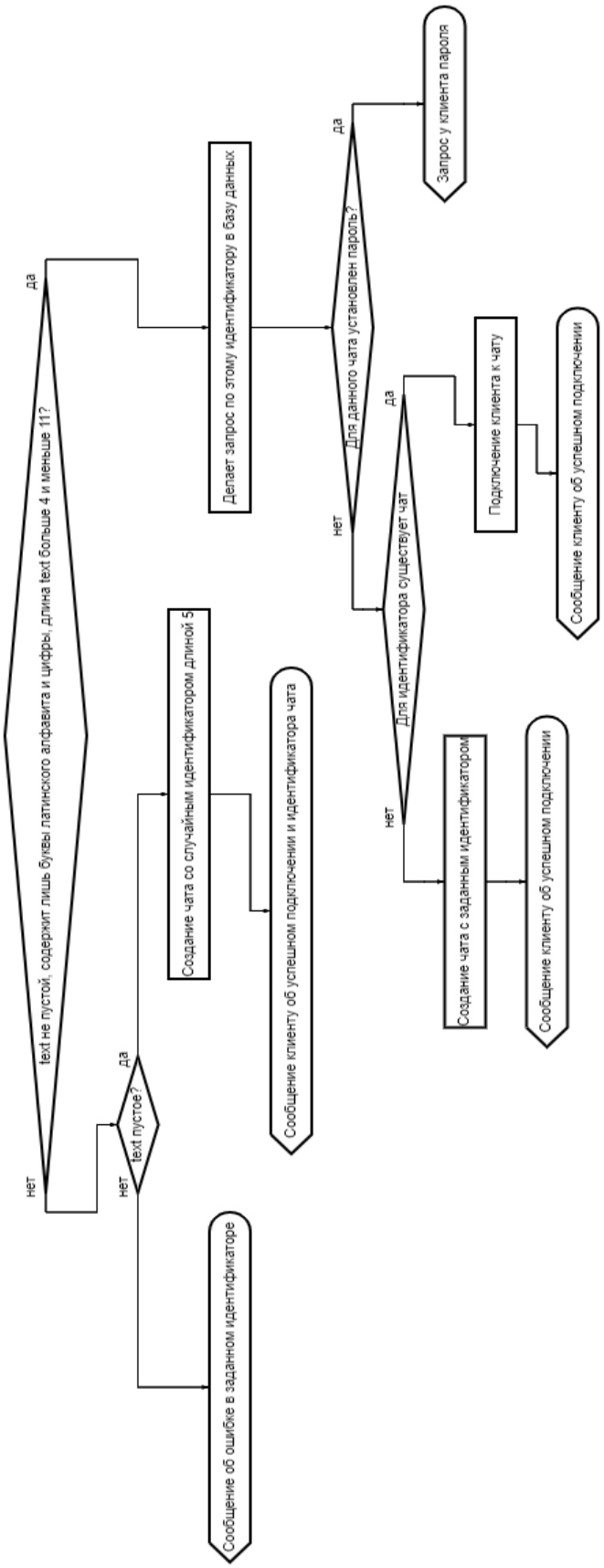
Документация Java Script – URL: https://learn.javascript.ru/ (дата обращения: на протяжении всего проекта) Текст: электронный

Документация Java Script – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: на протяжении всего проекта) Текст: электронный

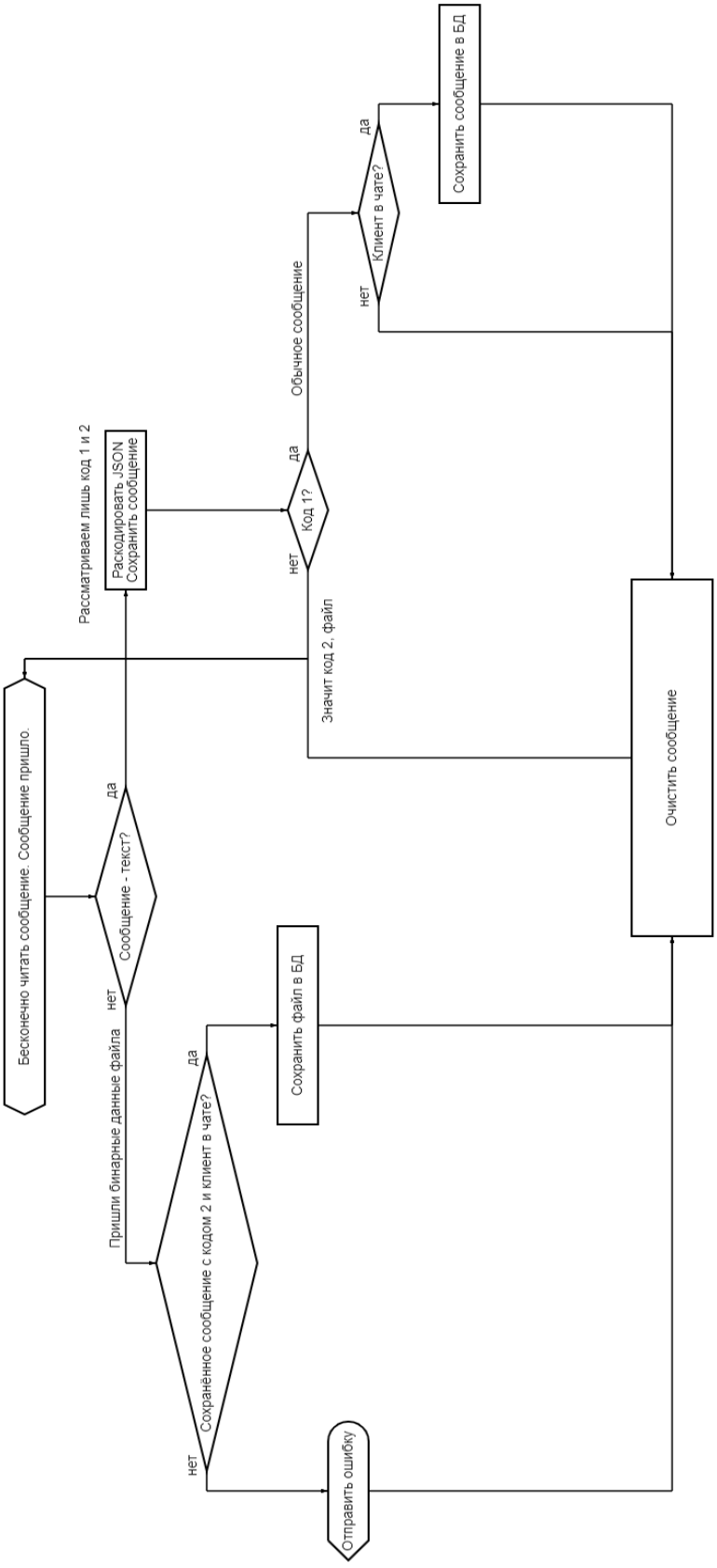
Документация Apache Cassandra – URL: https://cassandra.apache.org/doc/latest/ (дата обращения: 7.02.2023) Текст: электронный

Форумы по программированию – URL: https://stackoverflow.com/ (дата обращения: на протяжении всего проекта) Текст: электронный

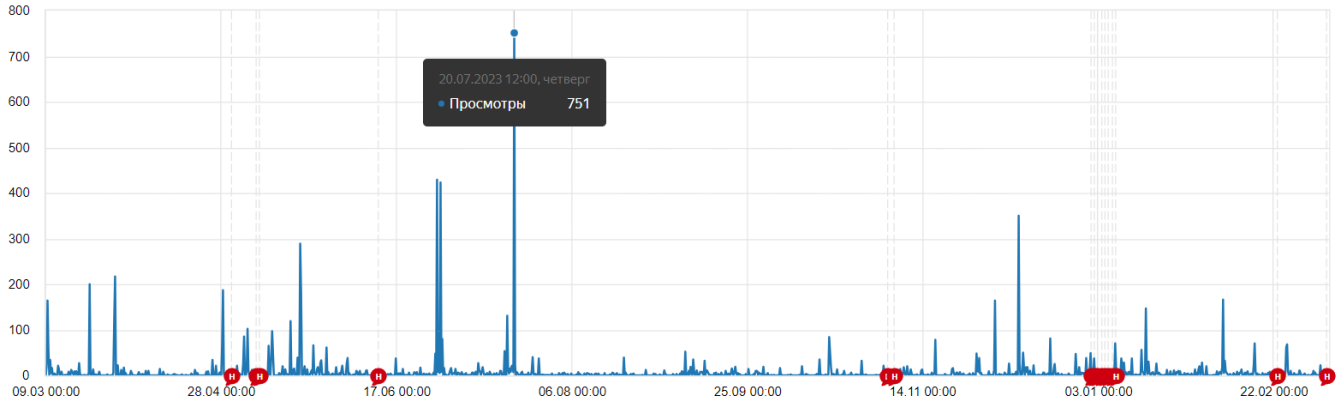
Приложение



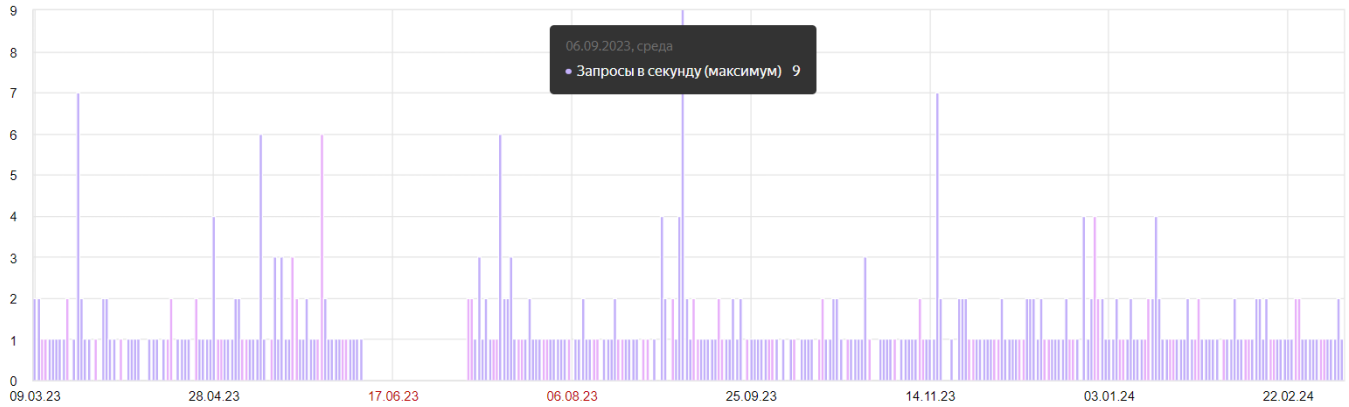
Блок-схема создания чата



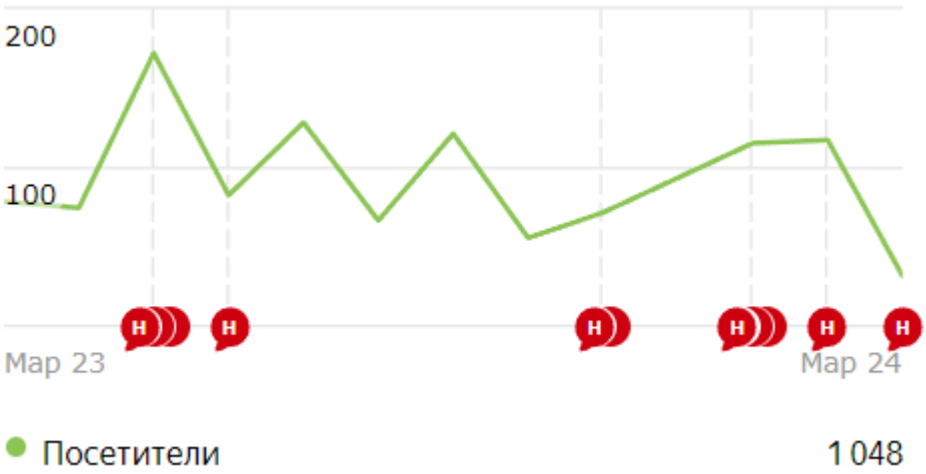
Блок-схема обработки текстовых и бинарных сообщений



Максимальное количество запросов в минуту



Максимальное количество запросов в секунду



Онлайн сайта