Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

Кафедра КБ

КУРСОВОЙ ПРОЕКТ

«Клиент-серверный чат»

Выполнил: Федоров Н.С

Группа: 5362

Факультет: КТИ

Проверил: Пелевин М.С.

Оценка:

Дата: « »

Санкт-Петербург, 2016 г.

**Оглавление**

Введение.........................................................................................................................3

Техническая спецификация ..........................................................................................3

Графический интерфейс...................................... .........................................................3

Работа с сетью со стороны сервера................................................................................4

QHttp

QSocket

QWebSocket

ChatServer

WebSocket…………………………............................................................................................5

Заключение ....................................................................................................................6

**Введение**

Целью курсового проекта стало получение практического навыка в разработке

веб приложений.

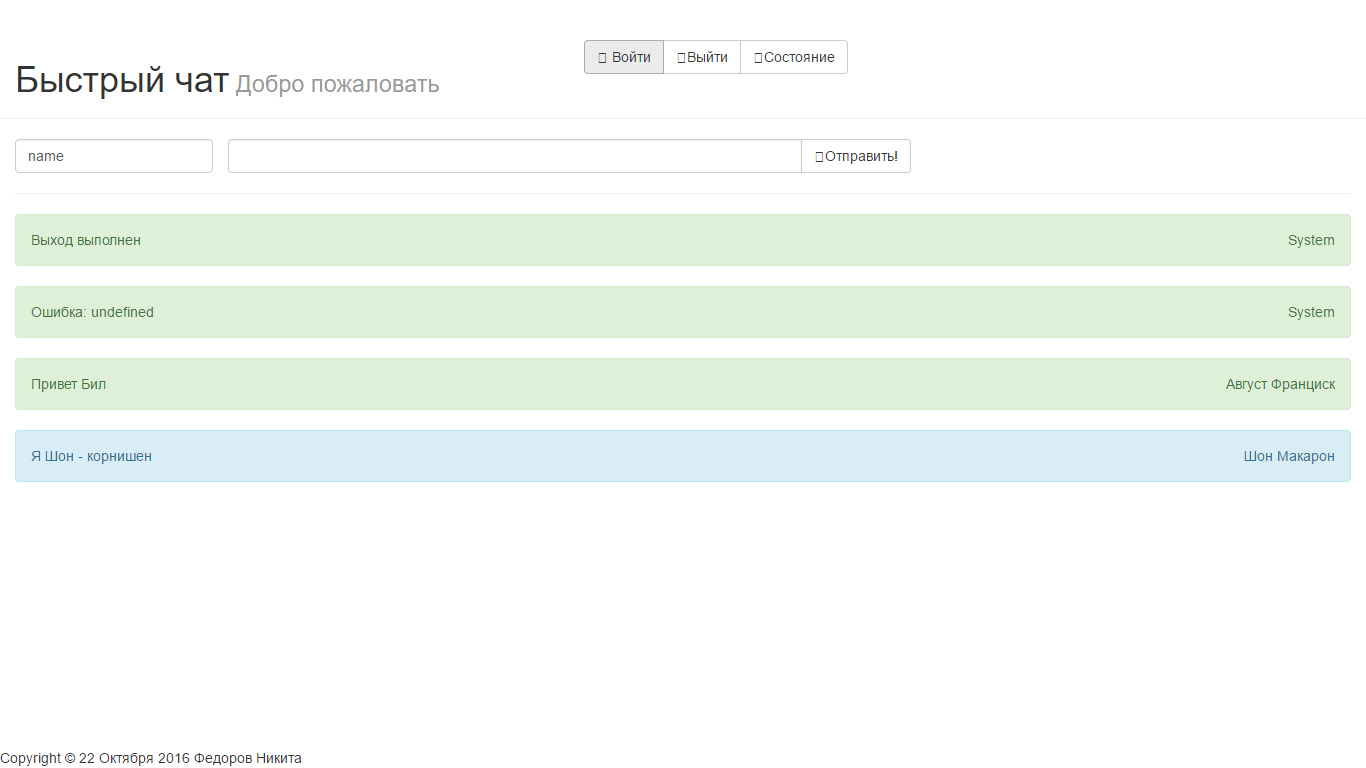
Результатом курсового проекта стал многопользовательский чат.

**Техническая спецификация**

В качестве основных инструментов для работы были выбраны:

* Qt Creator
* язык программирования C++
* Бутстрап
* Qt lib
* язык программирования js
* язык разметки html5

**Графический интерфейс**



**Работа с сетью**

Для работы с протоколами FTP и HTTP в библиотеке Qt имеются классы QFtp и QHttp. Они достаточно удобны для организации обмена файлами по сети.

Классы QFtp и QHttp основаны на низкоуровневом классе QSocket, который реализует представление сокетов TCP. Протокол TCP работает в терминах потоков данных, передаваемых между узлами сети. Класс QSocket, в свою очередь, реализован поверх QSocketDevice -- тонкой "обертки" вокруг платформо-зависимого сетевого API операционной системы. Класс QSocketDevice поддерживает протоколы TCP и UDP.

# Класс QHttp

Класс QHttp предназначен для создания клиентских приложений, работающих с протоколом HTTP. Он реализует набор функций, для выполнения наиболее распространенных операций этого протокола, включая get() и post(). Класс QHttp очень похож на QFtp.

Как и QFtp, объекты класса QHttp работают асинхронно. Функции get() и post() сразу же возвращают управление вызывающей программе, а собственно передача данных осуществляется в цикле обработки событий.

# Класс QSocket

Класс QSocket может использоваться при разработке приложений серверов и клиентов, работающих по протоколу TCP. TCP -- это протокол транспортного уровня, который является базой для множества других протоколов Интернет, включая FTP и HTTP, а так же может служить основой для разработки нестандартных протоколов обмена данными.

Протокол TCP ориентирован на потоки. Протоколы более высокого уровня, работающие поверх TCP, обычно подразделяются на строко-ориентированные и блочно-ориентированные:

* Строко-ориентированные протоколы передают данные в виде текстовых строк, каждая из которых завершается символом перевода строки.
* Блочно-ориентированные протоколы передают данные в виде блоков. Размер каждого блока содержится в отдельном поле, внутри блока.

Класс QSocket порожден от класса QIODevice, поэтому он в состоянии читать и писать данные из/в экземпляры классов QDataStream илиQTextStream. Одно важное отличие чтения данных из сети от чтения данных из файла состои в том, что перед вызовом оператора ">>" необходимо убедиться в том, что от удаленного хоста получены все данные. В случае ошибки мы можем получить непредсказуемый результат.

# Класс QWebSocket

Что же касается этого сокета, то он реализует модель «клиент-сервер» и является

обёрткой над QtspSocket

# Класс ChatServer

Основной класс. Реализует серверные методы

При создании соединения вызывается метод void onNewConnection(); где создается соединение с клиентом по средствам сокета QWebSocket \*pSocket и обрабатываются его состояния:  
connect(pSocket, SIGNAL(textMessageReceived(QString)), this, SLOT(processMessage(QString)));

connect(pSocket, SIGNAL(disconnected()), this, SLOT(socketDisconnected()));

Метод void processMessage(QString message); обрабатывает сообщения.

Метод void socketDisconnected(); завершает соединение.

WebSocket

Протокол WebSocket (стандарт [RFC 6455](http://tools.ietf.org/html/rfc6455)) предназначен для решения любых задач и снятия ограничений обмена данными между браузером и сервером.

Он позволяет пересылать любые данные, на любой домен, безопасно и почти без лишнего сетевого трафика.

Для открытия соединения достаточно создать объект WebSocket, указав в нём специальный протокол ws.:

var socket = new WebSocket("ws://javascript.ru/ws");

Пример запроса от браузера при создании нового объекта new WebSocket("ws://server.example.com/chat"):

GET /chat HTTP/1.1

Host: server.example.com

Upgrade: websocket

Connection: Upgrade

Origin: http://javascript.ru

Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==

Sec-WebSocket-Version: 13

Описания заголовков:

**GET, Host**

Стандартные HTTP-заголовки из URL запроса

**Upgrade, Connection**

Указывают, что браузер хочет перейти на websocket.

**Origin**

Протокол, домен и порт, откуда отправлен запрос.

**Sec-WebSocket-Key**

Случайный ключ, который генерируется браузером: 16 байт в кодировке [Base64](http://ru.wikipedia.org/wiki/Base64).

**Sec-WebSocket-Version**

Версия протокола. Текущая версия: 13.

Все заголовки, кроме GET и Host, браузер генерирует сам, без возможности вмешательства JavaScript.

**Заключение**

Чат позволяет обмениваться сообщениями.

Выполнение данного проекта позволило получить опыт разработки приложения, а так же на практике изучить ключевые классы необходимые для работы с сетью.