# РЕФЕРАТ

Пояснительная записка 72с., 24 рис., 3 табл., 3 прил., 5 частей, 10 источников.

Перечень ключевых слов: СИСТЕМА GPS-СЛЕЖЕНИЯ, GPS-ТРЕКЕР, СИСТЕМА GPS МОНИТОРИНГА, СЕРВИС ПРОСМОТРА ПУТИ, ОТОБРАЖЕНИЕ ПУТИ НА КАРТАХ.

Целью работы являлась разработка клиента и сервера отображения местоположения и пути перемещения объектов.

Проведен сравнительный обзор четырех аналогичных систем GPS-мониторинга.

В качестве исходных материалов были использованы некоторые готовые решения: библиотека получения координат с модуля GPS – Microsoft.Location, JavaScript фрэймворк jQuery, коннектор к MySQL базе данных MySQL Connector. Также были использованы документация с описанием API карт двух систем – Google maps и Яндекс карты и описание NMEA протокола, используемого для передачи координат от спутника к устройству-приемнику.  
 Компоненты системы реализованы на разных языках программирования. Компоненты «Клиент» и «Сервер данных» написаны на языке C#, «Web-интерфейс» на РНР. Для хранения данных используется БД MySQL.

# СОДЕРЖАНИЕ

[РЕФЕРАТ 2](#_Toc359622408)

[СОДЕРЖАНИЕ 3](#_Toc359622409)

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 6](#_Toc359622410)

[ОПРЕДЕЛЕНИЯ 7](#_Toc359622411)

[ВВЕДЕНИЕ 8](#_Toc359622412)

[1 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ 9](#_Toc359622413)

[1.1 Обследование предметной области 9](#_Toc359622414)

[1.2 Требования пользователя к системе 12](#_Toc359622415)

[2 РАЗРАБОТКА КОНЦЕПЦИИ 13](#_Toc359622416)

[2.1 Обзор существующих аналогов 14](#_Toc359622417)

[2.1.1 АNTOR MonitorMaster 14](#_Toc359622418)

[2.1.2 Geostron 14](#_Toc359622419)

[2.1.3 АвтоТрекер 14](#_Toc359622420)

[2.1.4 CityPoint 15](#_Toc359622421)

[3 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 16](#_Toc359622422)

[3.1 Общие положения 16](#_Toc359622423)

[3.1.1 Полное наименование системы и ее условное обозначение 16](#_Toc359622424)

[3.1.2 Наименования организаций-участников работ 16](#_Toc359622425)

[3.1.3 Плановые сроки начала и окончания работы по созданию системы 16](#_Toc359622426)

[3.1.4 Порядок оформления и предъявления заказчику результатов работ по созданию системы 16](#_Toc359622427)

[3.2. Назначение и цели создания системы 17](#_Toc359622428)

[3.2.1 Назначение системы 17](#_Toc359622429)

[3.2.2 Цели создания системы 17](#_Toc359622430)

[3.2.3 Характеристика объекта автоматизации 17](#_Toc359622431)

[3.3 Требования к системе 18](#_Toc359622432)

[3.3.1 Требования к системе в целом 18](#_Toc359622433)

[3.3.2 Требования к функциям системы 22](#_Toc359622443)

[3.3.3 Требования к видам обеспечения 23](#_Toc359622446)

[3.4 Порядок контроля и приемки системы 25](#_Toc359622452)

[4 ТЕХНИЧЕСКИЙ ПРОЕКТ 26](#_Toc359622453)

[4.1 Общесистемные решения 26](#_Toc359622454)

[4.1.1 Назначение системы 26](#_Toc359622455)

[4.1.2 Анализ требований и решения по компонентной структуре 27](#_Toc359622456)

[4.1.3 Описание вариантов использования 29](#_Toc359622457)

[4.2 Описание информационного обеспечения 35](#_Toc359622465)

[4.2.1 Состав информационного обеспечения 35](#_Toc359622466)

[4.2.2 Структура базы данных 35](#_Toc359622467)

[4.3 Выбор средств разработки 37](#_Toc359622470)

[4.4 Формат строк обмена сообщениями 39](#_Toc359622471)

[4.5 Описание программного обеспечения 40](#_Toc359622472)

[4.5.1 Структура программного обеспечения 40](#_Toc359622473)

[4.6 Программа и методика испытаний 42](#_Toc359622476)

[4.6.1 Объект испытаний 42](#_Toc359622477)

[4.6.2 Цель испытаний 42](#_Toc359622478)

[4.6.3 Условия и порядок проведения испытаний 43](#_Toc359622479)

[4.6.4 Контрольный пример 44](#_Toc359622480)

[5 РАБОЧАЯ ДОКУМЕНТАЦИЯ 48](#_Toc359622481)

[5.1 Руководство пользователя 48](#_Toc359622482)

[5.1.1 Введение 48](#_Toc359622483)

[5.1.2 Назначение и условия применения 49](#_Toc359622484)

[5.1.3 Описание операций 50](#_Toc359622485)

[ЗАКЛЮЧЕНИЕ 56](#_Toc359622510)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 57](#_Toc359622511)

[ПРИЛОЖЕНИя 58](#_Toc359622512)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Таблица 1 – Перечень определений, обозначений и сокращений

|  |  |
| --- | --- |
| Обозначение | Расшифровка |
| API | Интерфейс программирования приложений |
| EDGE | Цифровая технология беспроводной передачи данных для мобильной связи. |
| GPRS | пакетная радиосвязь общего пользования |
| GPS | Система глобального позиционирования |
| GSM | Глобальный стандарт цифровой мобильной сотовой связи, с разделением каналов по времени и частоте |
| GUI | Графический пользовательский интерфейс |
| MSDN | Подразделение компании Майкрософт, ответственное за взаимодействие фирмы с разработчиками. |
| NMEA | Текстовый протокол связи (как правило, навигационного) оборудования между собой. |
| TDMA | Множественный доступ с разделением по времени |
| URL | Единый указатель ресурсов (определитель местонахождения) ресурса. |
| UML | Унифицированный язык моделирования |
| WWAN | Беспроводная глобальная вычислительная сеть |
| WWW | Распределённая система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к Интернету. |
| XML | расширяемый язык разметки |
| БД | База данных |
| ИС | Информационная система |
| ПО | Программное обеспечение |
| СУБД | Система управления базой данных |

# ОПРЕДЕЛЕНИЯ

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования.

Web – Для обозначения Всемирной паутины используют слово веб (англ. web «паутина») и аббревиатуру WWW.

Геопозиционирование - представляет собой процесс определения местоположения объекта на Земле.

Сокеты (англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой [сетью](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C).

Фреймворк (англ. framework — каркас, структура) — структура программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

# ВВЕДЕНИЕ

В далеком 1957 произошел запуск первого искусственного спутника земли, это событие стало прорывом для науки и всего человечества и позволило сделать очередной шаг навстречу эре цифровых технологий.

В настоящее время каждому доступны технологии, которые 50 лет назад описывались лишь писателями-фантастами. Благодаря усилиям ученых тех лет стало возможно создать сеть спутников которые могли решать множество задач: передачу сигнала с одного континента на другой, спутниковое телевидение и телефония, даже фотографии поверхности планеты. Среди этих достижений и разработка системы глобального позиционирования. Изначально созданная в военных целях разработка, как это часто бывает, со временем нашла применение в мирных целях.

Сейчас GPS (Global Position System) модуль есть во всех современных телефонах, что позволяет владельцу определить свое местоположение относительно земли. С течением времени увеличивалась популярность данной технологии, а цены на GPS приемники падали – это сделало их доступными и позволило применять данную технологию во многих отраслях, что привело к спросу на специализированный софт.

# 1 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ

## Обследование предметной области

Процесс получения и отображения пути можно представить в виде следующей последовательности действий:

* определение географических координат текущего местоположения
* передача этих координат обрабатывающей части
* формирование пути
* отображение пути на карте

Для определения географических координат существуют две системы глобального позиционирования: GPS и ГЛОНАСС

GPS (англ. Global Positioning System — система глобального позиционирования, читается Джи Пи Эс) — спутниковая система навигации, обеспечивающая измерение расстояния, времени и определяющая местоположение. Позволяет в любом месте Земли (не включая приполярные области), почти при любой погоде, а также в космическом пространстве вблизи планеты определить местоположение и скорость объектов. Система разработана, реализована и эксплуатируется Министерством обороны США.

GPS состоит из трёх основных сегментов: космического, управляющего и пользовательского Спутники GPS транслируют сигнал из космоса, и все приёмники GPS используют этот сигнал для вычисления своего положения в пространстве по трём координатам в режиме реального времени.

Космический сегмент состоит из 32 спутников, вращающихся на средней орбите Земли.

Управляющий сегмент представляет собой главную управляющую станцию и несколько дополнительных станций, а также наземные антенны и станции мониторинга, ресурсы некоторых из которых являются общими с другими проектами.

ГЛОНАСС – Глобальная навигационная спутниковая система — советская и российская спутниковая система навигации. ГЛОНАСС предназначена для оперативного навигационно-временного обеспечения неограниченного числа пользователей наземного, морского, воздушного и космического базирования. Доступ к гражданским сигналам ГЛОНАСС в любой точке земного шара, на основании указа Президента РФ, предоставляется российским и иностранным потребителям на безвозмездной основе и без ограничений.

Основой системы должны являться 24 спутника, движущихся над поверхностью Земли в трёх орбитальных плоскостях. Принцип измерения аналогичен американской системе навигации NAVSTAR GPS. Основное отличие от системы GPS в том, что спутники ГЛОНАСС в своем орбитальном движении не имеют резонанса (синхронности) с вращением Земли, что обеспечивает им большую стабильность. Таким образом, группировка КА ГЛОНАСС не требует дополнительных корректировок в течение всего срока активного существования. Тем не менее, срок службы спутников ГЛОНАСС заметно короче.

Основной принцип работы обеих систем — определение местоположения путём измерения моментов времени приема синхронизированного сигнала от навигационных спутников антенной потребителя. Для определения трёхмерных координат GPS-приёмнику нужно иметь четыре уравнения: «расстояние равно произведению скорости света на разность моментов приема сигнала потребителя и момента его синхронного излучения от спутников».

Обе системы используют для определения координат протокол NMEA 0183, который передает данные о местоположении в градусах. Описание протокола в Приложение 2.

Для передачи данных главным фактором является расстояние на которое можно их передавать. Поэтому стоит ограничится в рассмотрении средств сегментом средств WWAN (Беспроводные глобальные сети).

GPRS (англ. General Packet Radio Service) — «пакетная радиосвязь общего пользования») — надстройка над технологией мобильной связи GSM, осуществляющая пакетную передачу данных. GPRS позволяет пользователю сети сотовой связи производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе Интернет. GPRS предполагает тарификацию по объёму переданной/полученной информации, а не по времени, проведённому онлайн.

EDGE (EGPRS) (англ. Enhanced Data rates for GSM Evolution) — цифровая технология беспроводной передачи данных для мобильной связи, которая функционирует как надстройка над 2G и 2.5G (GPRS)-сетями. Эта технология работает в TDMA- и GSM-сетях. Для поддержки EDGE в сети GSM требуются определённые модификации и усовершенствования.

Для формирования пути необходимо накопление некоторых координат, а следовательно и их хранение. После накопления необходимого минимума координат местоположения объекта из них формируется путь. Под путем, в данном случае, понимается упорядоченное по времени регистрации в системе множество точек, заданных географическими координатами, которые при последовательном соединении отрезками образуют траекторию близкую к реальной траектории перемещения объекта с определенной погрешностью.

Для отображения маршрута используется карта, имеющая формат координатной сетки аналогичный формату координат которые передает клиент. На данном этапе стоит выделить следующие решения, которые имеют встроенную преогромно реализованную координатную сетку:

* Google maps – пожалуй, самая популярная система карт
* Яндекс карты – аналог от отечественных разработчиков
* Navitel карты – еще один отечественный аналог
* Leaflet – Open Source аналог предшественников

Все эти решения имеют JavaScript API для создания пользовательских элементов, поверх карты.

В современном мире бизнес использует технологии отслеживания для решения многих задач, таких как:

* слежение за путем перемещения транспорта фирмы
* отслеживание угнанного транспорта
* просмотр маршрута путешествия
* поиск пропавших туристов
* определение длины проделанного пути

## 1.2 Требования пользователя к системе

Основной задачей разрабатываемой системы является отслеживание пути пользователя.

Дополнительные требования пользователя к системе:

* хранение координат пути пользователя на протяжении не менее 30 дней
* обеспечение ограничения доступа к просмотру маршрута пользователя
* обеспечения доступа через веб-интерфейс
* отображение на картах пути и текущего местоположения пользователя
* отображение пользователю его текущих координат, скорости и других показателей.

# 2 РАЗРАБОТКА КОНЦЕПЦИИ

Разрабатываемая система будет представлять собой сервис отслеживания пути пользователя и отображения его на карте, который можно применить для слежения за личным транспортом или транспортом фирмы, а также в других целях требующих определение положения объекта

Исходя из требований к системе, она должна будет обеспечивать сбор, передачу, обработку данных о местоположении пользователя, формирование маршрута и отображать его на одной из карт.

Система должна состоять из двух частей – клиентской и серверной.

Клиентская часть должна получать координаты местоположения объекта и передавать их серверной части, под объектом в данном случае понимается GPS-приемник, подразумевается, что пользователь на протяжении всего пути будет находится рядом. Также для клиентской части должен быть разработан формат команд для общения с сервером. Клиент должен обеспечивать сохранность пароля пользователя и защиту от несанкционированного его получения. Помимо этого клиентская часть должна иметь интерфейс для ввода настроек (таких как логин, пароль, периодичность отправки данных и т.п.). Интерфейс должен быть простым и интуитивно понятным. Допустимы погрешности в определении координат в зависимости от выбранной системы, и технических средств позиционирования.

В задачи серверной части входит авторизация пользователя, накопление данных для составления пути и проецирование его на карту. Также серверная часть должна обладать многопоточностью для обеспечения получения данных от множества клиентов с минимальными задержками и без коллизий. Серверная части допустимо терять не долее 15% передаваемых координат и не заботиться о восстановлении соединения с клиентом в случае его(соединения) разрыва.

## 2.1 Обзор существующих аналогов

## 2.1.1 АNTOR MonitorMaster

Уникальным преимуществом системы мониторинга транспорта и мобильных объектов АNTOR MonitorMaster является возможность интеграции с разными типами приборов и датчиков, что предоставляет клиентам выбор различных режимов мониторинга в соответствии с их потребностями. Данные о местонахождении объекта (сотрудника), пробеге, времени в пути, остановках и т. д. со всех устройств поступают и одновременно обрабатываются в АNTOR MonitorMaster для дальнейшей аналитики и отчетов. Таким образом, для осуществления различных видов мониторинга клиенту не нужно каждый раз приобретать программное обеспечение.

## 2.1.2 Geostron

Компания «Geostron» — отечественный разработчик систем спутникового мониторинга подвижных объектов с использованием технологий GPS и Глонасс. Компания обладает многолетним опытом разработок прикладного компьютерного программного обеспечения, благодаря которому стало возможным создание эффективных, удобных и простых систем спутникового контроля за транспортом.

Интерфейс клиентской программы интуитивно понятен, поэтому любой пользователь в кратчайшие сроки сможет освоить основные функции программы. Программа дает возможность получать маршруты передвижения за необходимый период и формировать отчеты, в реальном времени контролировать перемещение мобильного объекта.

Цены на трекеры:

* Персональный трекер GL100 — 4 800 руб.
* Персональный трекер Teltonika GH3000 — 5 900 руб.
* Автомобильный трекер Teltonika FM1100 — 4 900 руб.

## 2.1.3 АвтоТрекер

«АвтоТрекер» — многофункциональная интеллектуальная система спутникового мониторинга автотранспорта, разработанная компанией «Русские Навигационные Технологии». Данная система спутникового контроля автотранспорта позволяет в режиме реального времени определять точное местоположение автомобиля и траекторию его движения, а также контролировать огромное количество параметров работы транспортного средства: от скорости и пробега до расхода топлива и температуры в кузове.

## 2.1.4 CityPoint

CityPoint – ГЛОНАСС/GPS система контроля транспорта, призванная повысить эффективность управления парком транспортных средств, снизить расход ГСМ и другие эксплуатационные затраты, исключить нецелевое использование транспорта, а также обеспечить высокий уровень транспортной безопасности: сокращения аварийности, сохранности грузов и безопасности водителей и пассажиров.

Стоимость комплекта оборудования CityPoint Lite - 9 995 рублей.мСтоимость активации одной единицы транспорта (за 1 год) - 1 995 рублей.

Стоит отметить что все представленные выше поставщики систем предлагают к использованию свое оборудование, стоимость которого либо уже заложена в тарифный план системы, либо указана фиксировано за единицу, либо обговаривается непосредственно при заказе системы и зависит от количества заказываемых модулей. Абонентская плата, в большинстве случаев, включает в себя расходы на GPRS-трафик, но значительно превышает его стоимость.

Подведя итог можно сказать, что на рынке присутствует множество коммерческих решений разного уровня и качества исполнения, многие из которых работают только с оборудованием конкретной фирмы-производителя и требуют значительных финансовых вложений для развертывания и поддержания функционирования.

Разработка данной системы позволит не только сэкономить на покупке готовых решений, но и создать аналогичный продукт, который при наличии качественных технических средств сможет составить конкуренцию на рынке описанным выше продуктам. Создание гибкой системы с широкой областью применения и возможностью легкой доработки под конкретные нужды позволит некоторое конкурентное преимущество перед жестко заточенными и неповоротливыми системами.

# 3 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 3.1 Общие положения

## 3.1.1 Полное наименование системы и ее условное обозначение

В данном документе создаваемая информационная система называется «Сервер и клиент отображения местоположения и пути перемещения объекта».

Условное обозначение GPSTracker.

Также допускается использовать термин «Система» для обозначения разрабатываемой ИС.

## 3.1.2 Наименования организаций-участников работ

Разработчик: студент группы 4-42, Масалин Александр Сергеевич.

## 3.1.3 Плановые сроки начала и окончания работы по созданию системы

Плановый срок начала работ по созданию ИС – 1 сентября 2012 года.

Плановый срок окончания работ по созданию ИС – 17 июня 2013 года.

Сроки, состав и очередность работ являются ориентировочными и могут изменяться по согласованию с заказчиком.

## 3.1.4 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Результаты работы должны быть представлены в виде:

* технической и отчетной документации к проекту;
* электронного носителя, содержащего исходные коды, откомпилированные программы, установочные файлы.

## 3.2. Назначение и цели создания системы

## 3.2.1 Назначение системы

Система предназначена для автоматизации получения координат местонахождения пользователя и последующего отображения его пути по накопленным координатам, а также отображение дополнительной информации о состоянии пользователя.

## 3.2.2 Цели создания системы

В процессе создания системы должны быть достигнуты следующие цели:

* Определение пути следования и текущего местоположения объекта при проведении поисково-спасательных работ;
* оперативное и долгосрочное отслеживание местонахождения объектов и их состояния для предотвращения использования служебных авто в личных целях;
* оперативное отслеживание местонахождения объектов и их состояния в качестве охранных функций.
* приобрести знания и навыки проектирования геоинформационных систем и получить опыт работы с новыми технологиями

## 3.2.3 Характеристика объекта автоматизации

Объектом автоматизации, в данном случае, являются процессы сборки, обработки, отправки данных о текущем местоположении объекта, их (данных) накопления для последующего формирования маршрута с отображением его на карте.

Для серверной и клиентской части стоит рассмотреть данные процессы в отдельности:

Процесс сбора данных клиентом включает в себя:

* Получение информации о местоположении клиента с модуля геопозиционирования в формате строк протокола NMEA 0183

Процесс обработки данных клиентом:

* Разбиение полученных в процессе сборки строк на составляющие и выбор только необходимых для геопозиционирования объекта данных.
* При необходимости, перевод значений из одной системы счисления в другую.

Процесс отправки данных клиентом включает в себя:

* Установление соединения с сервером
* Формирование строки обмена данными в разработанном формате и отправка строки серверу.

Процесс сбора данных сервером включает в себя:

* Получение данных сервером от клиентов с помощью разработанного формата обмена данными между клиентом и сервером.

Процесс обработки данных сервером:

* Определение типа полученной от клиента строки и соответствующая реакция
* Разбиение строки с координатами на составляющие и сохранение полученных значений
* Формирование на основе накопленных данных пути объекта
* Проекция пути на карту

## 3.3 Требования к системе

## 3.3.1 Требования к системе в целом

## 3.3.1.1 Требования к структуре и функционированию системы

Данная система должна состоять из централизованного сервера сбора данных и связанного с ним хранилища данных, средства отображения пути на карте и приложения на устройстве, которое оснащено GPS-приемником.

## 3.3.1.2 Перечень подсистем, их назначение и основные характеристики

Разрабатываемая система должна состоять из компонентов, которые должны обеспечивать функционирование следующих подсистем:

Со стороны клиентской части:

* Подсистема определения координат: получает данные с устройства геопозиционирования, преобразует их в формат пригодный для отправки.
* Подсистема обмена данными: обеспечивает взаимодействие клиентской и серверной части.

Со стороны серверной части:

* Подсистема обмена данными: принимает входящие подключения, обеспечивает аутентификацию пользователей, а также поддержание текущих соединений; прием и отправку управляющих сообщений.
* Подсистема управления сервером: обеспечивает изменение настроек работы сервера и сохранение их на случай перебоев с электропитанием.
* Подсистема управления пользователями: позволяет создавать, удалять, редактировать данные о пользователях системы и предоставлять им различные привилегии, касающиеся просмотра пути.
* Подсистема ввода/вывода: предназначена для отображения информации о текущем состоянии сервера, в функции данной подсистемы также входит сохранение и загрузка настроек сервера.
* Подсистема хранения данных: хранит данные полученные от пользователя и информацию о самих пользователях.
* Подсистема отображения пути: формирует путь и проецирует его на карту.

## 3.3.1.3 Требования к способам и средствам связи для информационного обмена между компонентами системы

Все компоненты обмениваются между собой данными по протоколу TCP/IP. Компоненты сервера и отображения пути обмениваются данными посредством базы данных.

## 3.3.1.4 Требования к численности и квалификации персонала системы

Для эксплуатации ИС определены следующие роли:

* Администратор;
* Пользователь.

Для первоначального конфигурирования и настройки системы администратор должен:

* Иметь навыки работы с СУБД MySQL и знать язык запросов SQL.
* Устанавливать ПО на ОС семейства Windows (в том числе мобильных платформ).
* Иметь опыт и навыки настройки веб-сервера.

Пользователь должен иметь представление о работе системы и базовые навыки использования сети Интернет и ОС семейства Windows.

## 3.3.1.5 Требования к надежности и нагрузке

Система должна быть защищена от уязвимостей типа: sql injection (внедрение вредоносного sql кода) и DDOS (атака на систему с целью добиться ее неработоспособности).

Серверная часть должна поддерживать не менее 80 одновременных клиентских подключений, не демонстрируя при этом существенного падения производительности.

## 3.3.1.6 Требования к эргономике и технической эстетике

Изменение настроек серверной и клиентской части должно быть реализовано с помощью GUI, к которому предъявляются следующие требования к эргономике и эстетике:

* Все элементы должны быть выполнены в едином стиле
* Для оформления не должны быть использованы резкие, раздражающие цвета
* GUI должно содержать только необходимые настройки и минимум оформления
* Все манипуляции при работе c GUI должны обеспечиваться двумя средствами: клавиатура и манипулятор «мышь».

Отображение пути и положения объекта должно обеспечиваться при помощи web-интерфейса. Дизайн web-интерфейса системы должен удовлетворять следующим требованиям к эргономике и эстетике:

* Иметь минимальный набор графических элементов и обеспечивать высокую скорость загрузки.
* Корректно отображаться при разрешениях экрана: 1024х768 1152х864 1280х800 1280х1024 1440х900 1680х1050 1920х1080 1920х1200
* Обеспечить идентификацию раздела где находится пользователь и легкую навигацию по разделам.
* Для оформления не должны быть использованы резкие, раздражающие цвета
* Элементы управления должны органично вписываться в дизайн интерфейса, а их назначение должно быть понятно без дополнительных манипуляций.

## 3.3.1.7 Требования к защите информации от несанкционированного доступа

Система должна обеспечивать доступ к БД только авторизованным в системе по логину и паролю пользователям, либо администратору, имеющему непосредственный доступ к компьютеру, содержащему БД.

Просмотр пути пользователя доступен только пользователям, имеющим соответствующие привилегии.

Доступ к настройке серверной части и конфигурационному файлу должен иметь только администратор системы.

Пароль пользователя должен храниться и передаваться в зашифрованном виде.

## 3.3.1.8 Требования к патентной чистоте

Установка системы в целом, как и установка отдельных частей системы не должна предъявлять дополнительных требований к покупке лицензий на программное обеспечение сторонних производителей и не нарушать действующего законодательства Российской Федерации.

В процессе разработки не должны нарушаться условия использования сторонних средств и компонентов разработки, использованных в процессе создания системы, используемые средства не должны вести к серьезному удорожанию проекта.

Лицензионные соглашения и условия использования должны накладывать минимум ограничений на функционал системы, или не накладывать таковых вовсе.

## 3.3.1.9 Требования по сохранности информации при авариях

Для предотвращения потери данных в процессе эксплуатации системы должны создаваться резервные копии данных, что является частью комплекса мер по технической поддержке системы и входит в обязанности администратора системы.

## 3.3.2 Требования к функциям системы

## 3.3.2.1 Требования к функциям клиентской части системы

* Изменение настроек подключения таких как имя пользователя, пароль, хост, порт.
* Управление частотой отправки сообщений серверу.
* Шифрование пароля пользователя и сохранение его на устройстве.
* Обмен сообщениями с серверной частью по беспроводному каналу связи.

## 3.3.2.2 Требования к функциям серверной части системы

* Управление пользователями системы (добавление, редактирование, удаление).
* Получение сведений о текущем состоянии подключений пользователей.
* Обработка управляющих сообщений соответствующее реагирование на них.
* Хранение полученных координат.
* Настройка отображения пути, в частности цвета и длины.
* Отображение последнего(текущего) местоположения объекта.
* Настройка внешнего вида карты и карты.
* Управление масштабом карты.
* Обмен сообщениями с клиентской частью по беспроводному каналу связи.

## 3.3.3 Требования к видам обеспечения

## 3.4.3.1 Требования к информационному обеспечению системы

В состав системы должны входить следующие виды информации:

* Информация о пользователях.
* Информация о пути перемещения.

Формат и структура данных, а также формат сообщений, которыми обмениваются клиент и сервер определяется на этапе технического проектирования.

Технические средства, обеспечивающие хранение информации, должны использовать современные технологии, позволяющие обеспечить повышенную надежность хранения данных и оперативную замену оборудования (распределенная избыточная запись/считывание данных; зеркалирование; независимые дисковые массивы; кластеризация).

Для хранения должна использоваться СУБД с поддержкой языка SQL.

## 3.3.3.2 Требования к лингвистическому обеспечению системы

Для реализации серверной и клиентской частей необходимо использовать язык C#.

Для реализации подсистемы отображения пути необходимо использовать языки PHP и JavaScript.

Для реализации алгоритмов манипулирования данными в подсистеме хранения данных необходимо использовать стандартный язык запроса к данным SQL.

Для взаимодействия с пользователем должен использоваться русский язык.

## 3.3.3.3 Требования к программному обеспечению системы

Использованное для создания системы программное обеспечение должно:

* Максимально облегчать работу по созданию системы.
* Иметь полную и понятную документацию.
* Быть полностью совместимым с другими компонентами, используемыми в разработке.
* Продукты, разрабатываемые при помощи выбранных средств, должны поддерживаться ОС семейства Windows.

## 3.3.3.4 Требования к техническому обеспечению

Техническое (аппаратное) обеспечение всех компонентов системы должно удовлетворять следующим требования:

* Обеспечивать полноценное функционирование описанного в требованиях к программному обеспечению ПО.
* Обеспечивать необходимую вычислительную мощность для функционирования системы и используемого ею программного обеспечения.
* Обеспечивать бесперебойное функционирование системы в случаях перебоев в электрическом питании технического обеспечения путем использования блоков бесперебойного питания.
* Иметь доступ к сети Интернет.
* Иметь средства коммутации компонентов системы, определенные в ходе технического проектирования.

## 3.3.3.5 Требования к методологическому обеспечению

Нормативно-техническая документация системы должна содержать техническое задание, технический проект и рабочий проект, оформленные в соответствии с ГОСТ 34.

## 3.4 Порядок контроля и приемки системы

Информационная система должна быть спроектирована до 17 июня 2013 года. При этом должны быть составлены техническое задание, технический проект, рабочий проект и окончательный вариант готовой Системы. В течение этого срока необходима периодическая сдача проектной документации и демонстрация прототипов программы.

Стадии и порядок испытаний указан в техническом проекте.

# 4 ТЕХНИЧЕСКИЙ ПРОЕКТ

## 4.1 Общесистемные решения

## 4.1.1 Назначение системы

Разрабатываемая система предназначена для сбора данных о перемещении объекта. Основным взаимодействующим лицом является пользователь. Он взаимодействует как с клиентской частью, так и с серверной частью. Вторым действующим лицом является администратор. Он обеспечивает развертывание системы и ее бесперебойное функционирование.

Для дальнейшего проектирования, стоит рассмотреть более подробно варианты использования данной системы этими действующими лицами. Так как в системе уже определены серверная и клиентская часть, то стоит сразу выделить их, обозначив клиентскую часть как «Клиент», серверную как «Сервер».

Для иллюстрации способов взаимодействия пользователей с Системой ниже приведена диаграмма вариантов использования (рисунок 1). На ней изображены действующие лица и две основные части системы, на каждой из которых отображены варианты использования данной части действующими лицами.

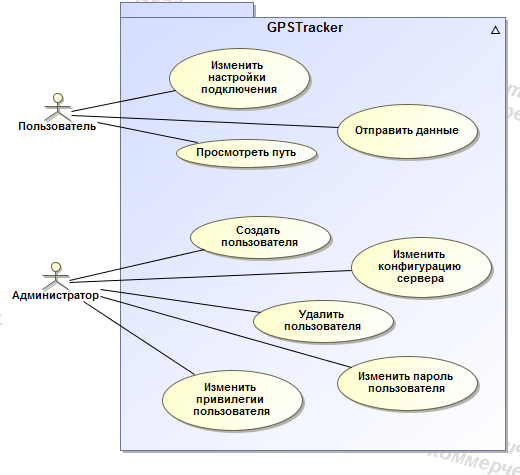


Рисунок 1 – Диаграмма вариантов использования

Каждый вариант использования представляет собой отдельную задачу (или подзадачу), которую может выполнять Система. Подробное описание каждого варианта использования будет приведено в пункте 4.1.3

## 4.1.2 Анализ требований и решения по компонентной структуре

Исходя из требований, предъявляемых к системе, а именно обеспечение web-интерфейса отображения пути и GUI редактирования настроек можно определить несколько компонентов и перераспределить функции системы, определенные в пункте 3.3.1.2, между ними:

Компонент «Клиент» - реализует функции следующих подсистем:

* Подсистема определения координат
* Подсистема обмена данными

Компонент «Сервер данных» реализует включает в себя следующие подсистемы:

* Подсистема обмена данными
* Подсистема ввода/вывода

Компонент «База данных» реализует подсистему хранения данных.

Компонент «web-интерфейс» реализует подсистему отображения пути

Компонент «Конфигуратор сервера» реализует функции подсистемы управления сервером и пользователями

Данное разбиение необходимо для обеспечения гибкости системы и увеличение надежности. Низкая связанность компонентов и узкая специализация решаемых компонентом задач позволят легко изменять и расширять систему, заменять компоненты по мере необходимости без изменения всего проекта целиком. Использование web-интерфейса позволяет решить сразу несколько задач: предоставление клиенту доступа к данным хранящимся на сервере при обеспечении их конфиденциальности, помимо этого клиент будет избавлен от необходимости каждый раз обновлять клиентскую часть. Выделение в отдельный компонент конфигуратор сервера функций по его настройке позволит оставить в компоненте «сервер данных» только необходимый минимум функций и избавит его от тяжеловесного графического интерфейса, и функции управления пользователями; так же из-за данного разделения в системе появляется еще один компонент – «Файл конфигурации». Данный файл обеспечивает сохранность настроек сервера при отключении электропитания.

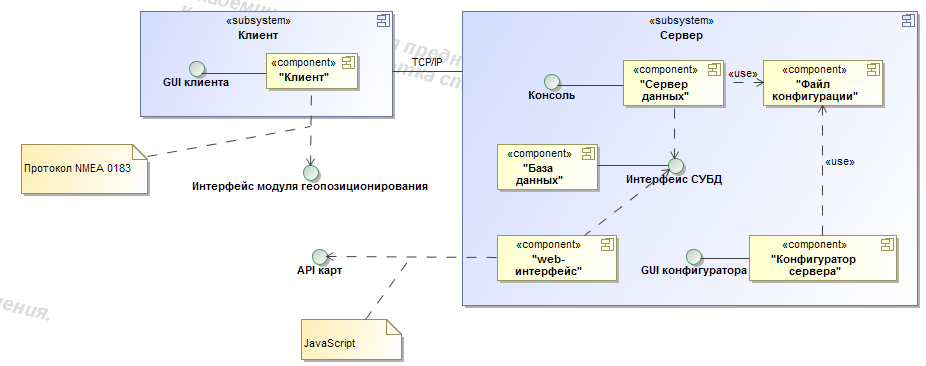


Рисунок 2 – Диаграмма компонентов

На рисунке 2 показаны две основные части системы – клиентская и серверная, которые обмениваются сообщениями между собой по протоколу TCP/IP. Также на данной диаграмме видно, к какой части относится компоненты, и как они взаимодействуют между собой.

## 4.1.3 Описание вариантов использования

## 4.1.3.1 Вариант использования Отправить данные

Пользователь запускает клиентское приложение, которое авторизуется на сервере и начитает получать данные с модуля геопозиционирования, интерпретировать эти данные, и отправлять серверной части. Эти процессы изображены на диаграмме последовательности рисунок 3.

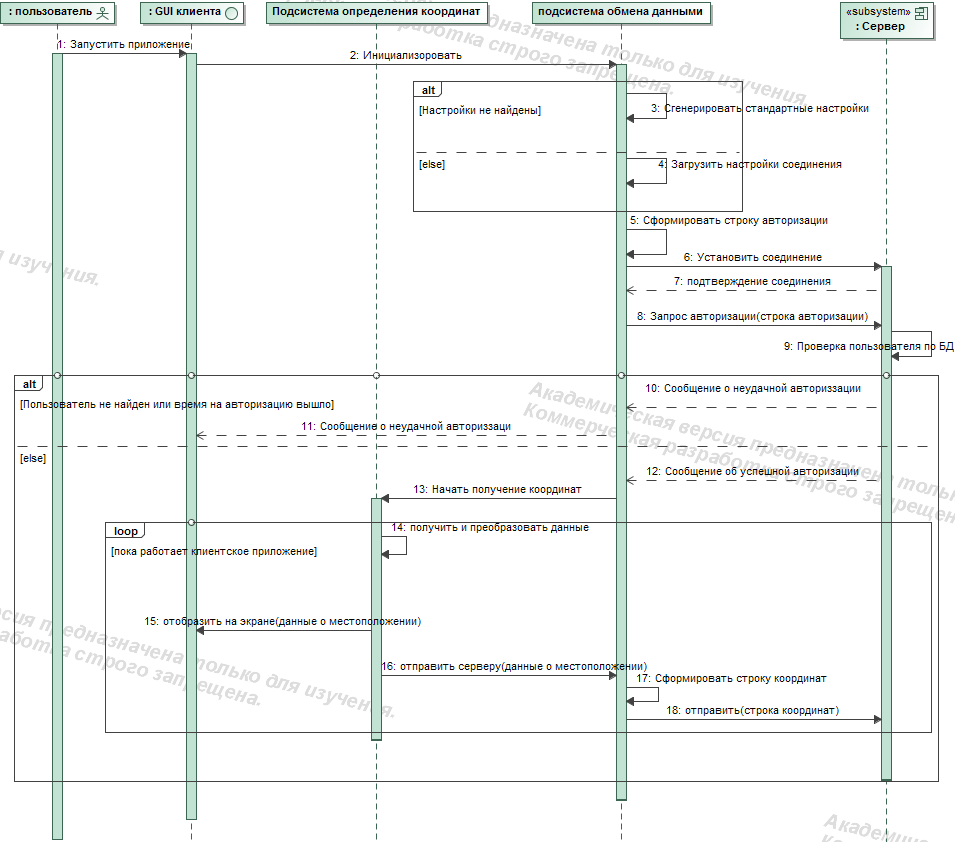


Рисунок 3 – Диаграмма последовательности для варианта использования отправить данные

## 

## 4.1.3.2 Вариант использования Изменить настройки подключения

Пользователь при помощи GUI редактирует настройки подключения, такие как логин, пароль, хост, порт.

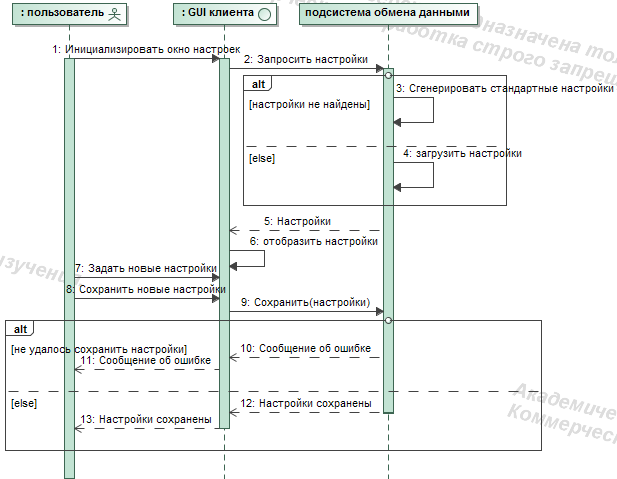


Рисунок 4 – Диаграмма последовательности для варианта использования изменить настройки подключения

## 4.1.3.3 Вариант использования Создать пользователя

Администратор, пользуясь окном управления пользователями в конфигураторе сервера, создает пользователя в системе. При этом система совершает следующие действия: создается таблица, которая будет содержать координаты перемещения объекта (пользователя) в БД. Затем создается запись в таблице пользователей. Администратор, после всех манипуляций, получает сообщение об итоге операции.

## 4.1.3.4 Вариант использования Редактировать пользователя

Администратор, пользуясь окном управления пользователями в конфигураторе сервера, изменяет данные о пользователе из системы. При этом система совершает следующие действия: Изменяется запись о пользователе. Администратор, после всех манипуляций, получает сообщение об итоге операции.

## 4.1.3.5 Вариант использования Изменить конфигурацию сервера

Администратор, при помощи конфигуратора сервера изменяет настройки и записывает их в файл, который потом будет использован сервером данных

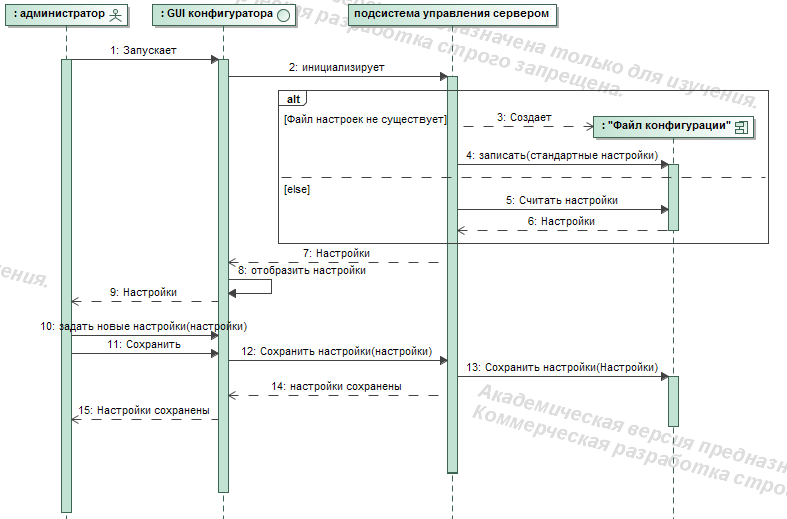


Рисунок 5 – Диаграмма последовательности для варианта использования изменить конфигурацию сервера

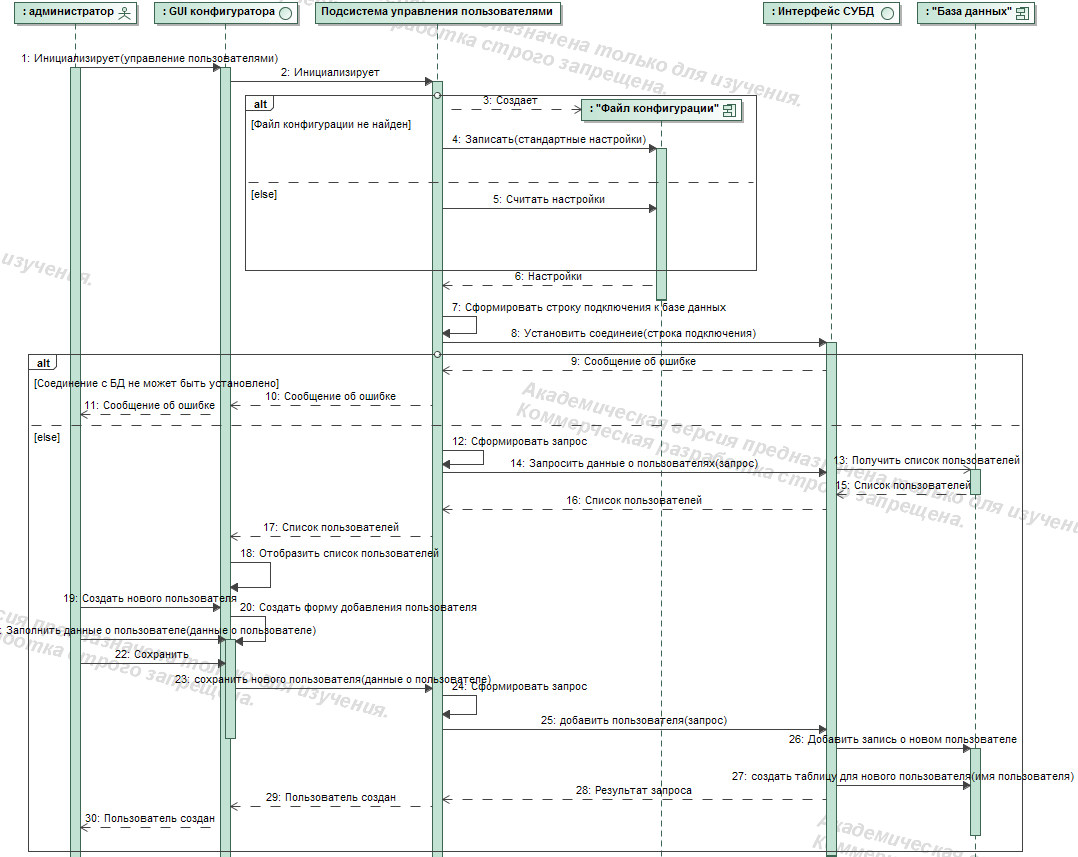


Рисунок 6 – Диаграмма последовательности для варианта создать пользователя

## 4.1.3.6 Вариант использования Изменить пароль пользователя

Администратор, пользуясь средствами конфигуратора сервера выбирает нужного пользователя и изменяет его пароль. Новый пароль незамедлительно шифруется и сохраняется в БД.

## 4.1.3.7 Вариант использования Просмотреть путь

Пользователь, используя браузер, переходит по адресу веб-интерфейса системы. Где он вводит логин и пароль, после чего веб-сайт получает координаты из базы данных, формирует из них путь и посредством API карт проецирует путь на карту.

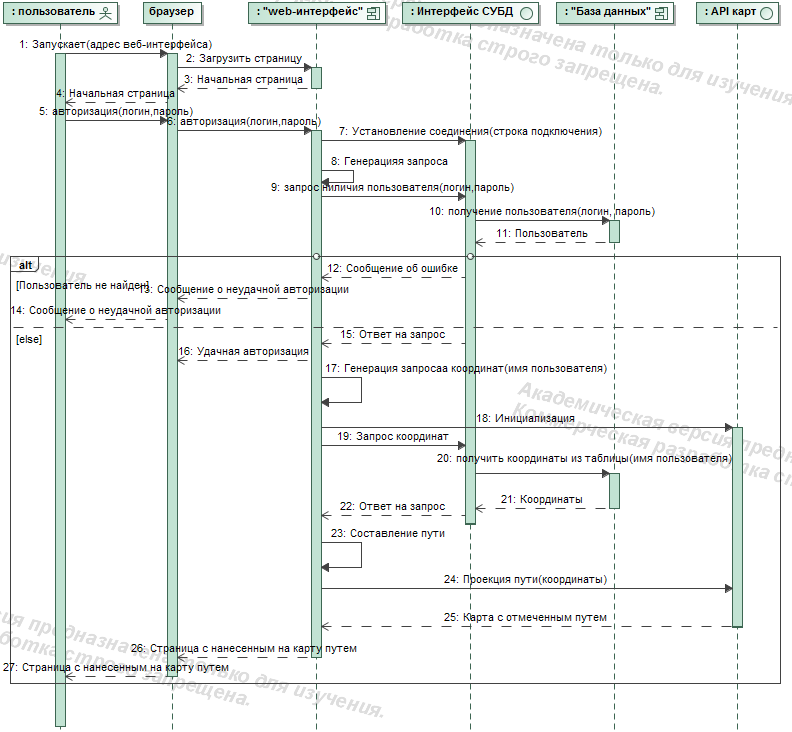


Рисунок 7 – Диаграмма последовательности для варианта использования просмотреть путь

## 4.2 Описание информационного обеспечения

## 4.2.1 Состав информационного обеспечения

В состав информационного обеспечения входят:

* Файл конфигурации сервера. Представляет из себя xml-файл и предназначен для хранения настроек сервера, содержит такие данные как время на авторизацию, количество одновременных подключений, порт сервера.
* База данных. Содержит информацию о пользователях и данные о координатах их перемещения

## 4.2.2 Структура базы данных

## 4.2.2.1 Организация сбора и передачи информации

Данные в систему заносятся и редактируются в процессе работы с системой.

## 4.2.2.2 Организация внутримашинной информационной базы.

Внутримашинная база данных организована в виде реляционной табличной структуры, обслуживаемой специализированным программным обеспечением – СУБД MySQL.

В качестве основного носителя данных в системе применяются встроенные серверные накопители на жестких магнитных дисках. Организация данных на дисках и доступ к хранимой информации обеспечиваются средствами используемых серверных операционных систем и СУБД, входящих в состав программного обеспечения комплекса технических средств.

Обновление и очистка базы данных производится в ходе нормального функционирования системы, в соответствии с заложенной в программные компоненты системы процедурной логикой. Улучшение этой процедуры является одним из путей развития системы.

Обеспечение связи и поддержание ссылочной целостности возлагается на СУБД.

База данных предназначена для структурированного хранения данных о пользователях и координатах их перемещения.

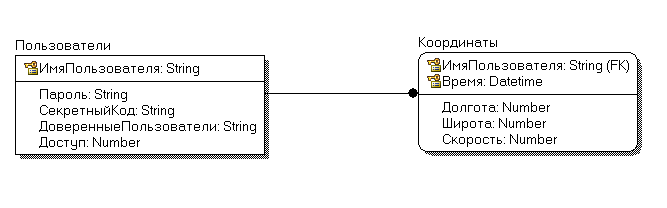


Рисунок 8 – Логическая модель базы данных

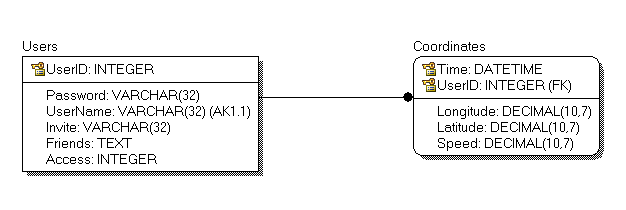
Рисунок 9 – Физическая модель базы данных

Таблица 2 – Структура таблицы пользователей (Users)

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип | Размерность | Описание |
| UserID | Integer | 32 | Идентификатор пользователя |
| UserName | Varchar | 32 | Имя пользователя, уникальный атрибут |
| Password | Varchar | 32 | Пароль |
| Invite | Varchar | 32 | Секретный код |
| Friends | Text |  | Список доверенных пользователей |
| Permission | Integer | 1 | Маркер доступа |

Список доверенных пользователей – содержит список пользователей, чьи маршруты можно просматривать, используя данную запись пользователя.

Маркер доступа используется для записи пользователя в целях предоставления ему дополнительных прав, таких как просмотр маршрутов клиентов, которые не входят в список доверенных пользователей.

Таблица 3 – Структура таблиц маршрутов пользователей (Coordinates)

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип | Размерность | Описание |
| UserID | Integer | 32 | Идентификатор пользователя |
| longitude | Decimal | (10,7) | Долгота |
| latitude | Decimal | (10,7) | Широта |
| Speed | Decimal | (10,7) | Скорость |
| Time | Datetime |  | Дата и время |

## 4.3 Выбор средств разработки

В ходе обследования предметной области были выделены технологии пригодные для создания данной системы. Данный пункт посвящен обоснованию выбора технологий которые будут использованы для создания проекта.

Первым делом стоит определится с выбором системы глобального позиционирования. В качестве вариантов рассматриваются две системы: GPS и ГЛОНАСС. В проекте будет использоваться система GPS так как на данный момент точность позиционирования ее немного выше, второе достоинство – дешевизна и доступность оборудования.

Следующий шаг – выбор ОС устройств, под которые будут создаваться приложения. Для серверной части выбрана ОС семейства Windows – Windows XP, компании Microsoft. Несмотря на дополнительные затраты связанные с приобретением лицензии на данный продукты они остаются наиболее популярным решением на рынке. Для клиентской части выбрана ОС Windows mobile 6.5, она позволит выбирать технологию передачи данных без изменений программного кода.

Еще один важный шаг –выбор технологии обмена данными между клиентской и серверной частями системы. Определяющим будет факторы надежности передачи данных, расстояние на которое эти данные можно передать, и поддержка технологий частями системы. В данном проекте для обмена сообщениями будет использована технология GPRS из-за большой площади покрытия.

Так как система уже разделена на компоненты будет разумным подбирать средства разработки для каждого из них в отдельности.

Сервер данных и клиентское приложение будут написаны на языке C# под платформу .NET. Данный выбор был сделан благодаря высокому быстродействию .NET приложений в среде Windows, а также благодаря реализации технологии сокетов обеспечивающих взаимодействие клиента с сервером. Не малую роль также сыграло наличие русифицированной документации и Специализированной среды разработки Visual Studio.

Web-интерфейс буден реализован на языке php, из –за простоты языка и небольшого объема скрипта. Для оформления также будут использованы каскадные таблицы стилей и гипертекстовый язык разметки. Для взаимодействия с картами – язык java script. Для визуальных эффектов – java script фрэймворк jQuery.

При выборе карт решено было оставить два варианта – Google maps и Яндекс карты, так как каждое из этих решений имеет свои плюсы и свои минусы.

В качестве СУБД была выбрана MySQL т.к. она прекрасно взаимодействует как с языком C#, так и с РНР.

Конфигуратор сервера – приложение Windows Forms, написанное на языке C#.

## 4.4 Формат строк обмена сообщениями

В данном пункте описывается формат строк которыми обмениваются клиент и сервер.

Всего определено три типа строк:

* Строка авторизации
* Строка данных
* Строка сообщений

Формат строки авторизации:

Клиент, после подключения к серверу, первым сообщение отправляет логин и md5-хэш пароля в формате:

!login@md5hash

Пример: !desu@202cb962ac59075b964b07152d234b70

(длина любого сообщения не должна превышать 255 символов)

Формат строки координат:

Latitude|Longitude|Speed|DateTime

Latitude, Longitude – широта и долгота в градусах. Представляют собой числа с плавающей запятой, с точностью до 7 знаков после запятой

Speed – скорость в узлах/час. Положительное число, точность 2 знака после запятой

DateTime – дата и время в формате: dd.MM.yy hh:mm:ss (в принципе должен подойти любой формат даты и времени день.месяц.год часы:минуты:секунды.

Пример строки координат

56,9642717|41,004865|0|27.03.13 12:31:32

## 4.5 Описание программного обеспечения

## 4.5.1 Структура программного обеспечения

Так как общая диаграмма классов очень объемная, то следует рассмотреть ее по частям, в зависимости от компонентов.

## 4.5.1.1 Структура программного обеспечения компонента «Клиент»

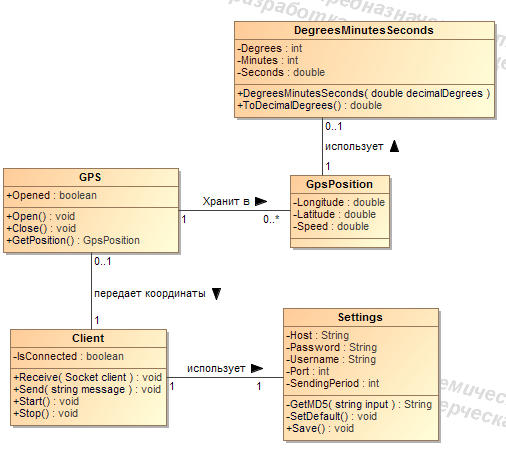


Рисунок 10 – Диаграмма классов компонента Клиент

Класс Client – программная реализация подсистемы обмена данными

Класс GPS – реализация подсистемы получения координат, GpsPosition – промежуточный класс для кранения координат и другой информации.

DegreeMinutesSecunds – класс для перевода в градусы данных со спутника

Класс Settings – сохраняет и загружает настройки, также занимается шифрованием пароля.

## 4.5.1.2 Структура программного обеспечения компонента «Сервер данных»

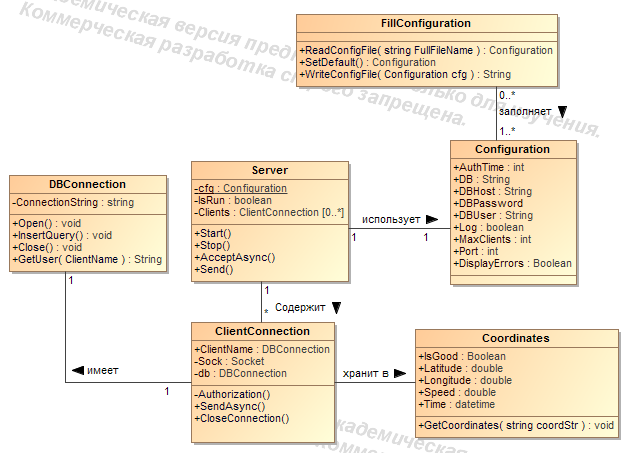


Рисунок 11 – Диаграмма классов компонента Клиент

Класс FillConfiguration – занимается загрузкой и сохранением настроек сервера.

Класс Configuration – хранит настройки сервера, полученные предыдущим классом.

Класс Server- основной класс данного компонента. Ожидает и принимает подключения, выполняет часть функций подсистемы обмена данными, ведет учет всех подключенных клиентов.

Класс ClientConnection – не менее важный класс. Создается в момент принятия нового запроса на подключения со стороны клиентской части. Экземпляры данного класса функционируют в отдельных потоках. Основные задачи:

* авторизация пользователя
* формирование строки сообщения клиенту
* занимается обменом данными с клиентом
* распознает строку присланную клиентом и реагирует на сообщения
* выделяет координаты из строки координат клиента
* записывает координаты в БД

Класс DBConnection осуществляет подключение и обмен данными с базой данных.

Класс Coordinates – класс-контейнер координат, полученных от клиента.

## 4.6 Программа и методика испытаний

## 4.6.1 Объект испытаний

Предварительные испытания проводятся для всей разработанной информационной системы согласно ГОСТ 34.603-92 и являются комплексными.

## 4.6.2 Цель испытаний

Целью проведения испытаний является проверка работоспособности системы в целом и ее отдельных задач.

## 4.6.3 Условия и порядок проведения испытаний

Для проведения испытаний создается контрольный пример. В качестве исходной информации для контроля будет использован фрагмент информации в объеме, достаточном для обеспечения необходимой достоверности информации.

При конечной сдаче системы необходимо продемонстрировать работу компонентов системы.

Сперва проводится предварительная подготовка базы данных (создание пользователя и генерация структуры БД).

После, демонстрируется работа конфигуратора сервера. Конфигуратор должен создать файл настроек сервера, которые были введены в него посредством графического интерфейса.

Следующий шаг – запуск сервера данных, на данном этапе он должен успешно соединиться с БД и ожидать подключения клиентов.

Далее демонстрируется работа клиента – при этом вводятся настройки (Если настройки жестко прошиты в клиенте, то эту часть можно пропустить и перейти сразу к соединению с серверу). После запуска клиента, он должен отправить строку аутентификации и авторизоваться на сервере. Клиент должен начать отправлять координаты серверу (В зависимости от типа и качества GPS-приемника процесс получения первых координат может занять от 5 до 45 минут).

Параллельно этому сервер данных должен записывать координаты в БД и отображать уведомления согласно настройкам, введенным в конфигуратор.

Для просмотра пути в браузер вводится URL адрес web-интерфейса, затем вводятся логин/пароль пользователя, затем выбираются карты на которых будет отображаться путь.

## 4.6.4 Контрольный пример

Оценить работоспособность разработанной информационной системы можно с помощью описанного ниже контрольного примера.

Настройка сервера данных:

В конфигураторе сервера введите необходимые настройки сервера, базы данных и выберите опции ведения логов и отображения ошибок (рисунок 12).

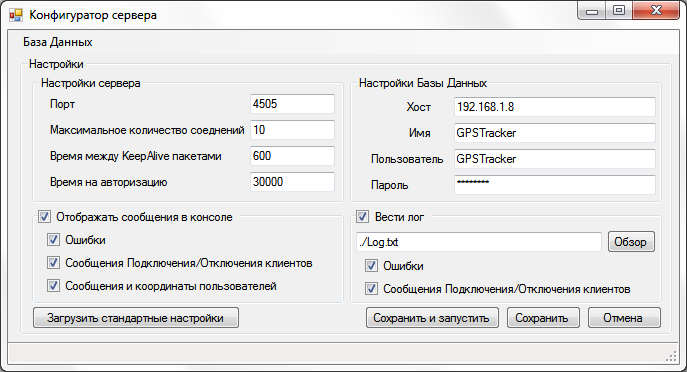


Рисунок 12 – Конфигуратор сервера

Запустите само приложение Сервер данных



Рисунок 13 – Запуск сервера данных

Настройка клиента:

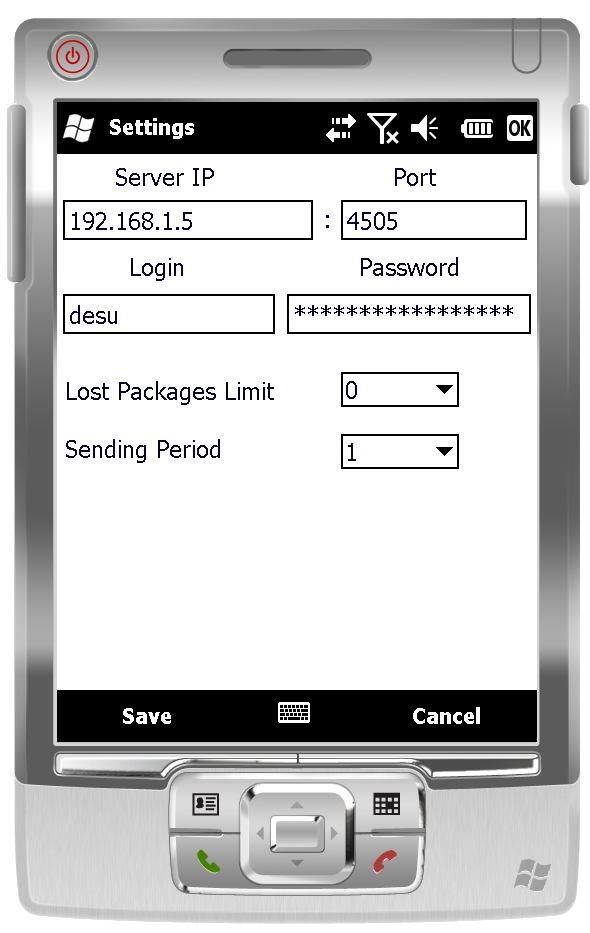


Рисунок 14 – Настройки клиентского приложения

Запустите приложение, введите логин/пароль, затем пункт меню старт.



Рисунок 15 – Клиент. Подключение к серверу

При попытке подключения клиента отображается соответствующее сообщение: Trying to connect – это означает что клиент установил соединение с сервером, и сервер ожидает строку авторизации пользователя.

После удачной авторизации отображается сообщение Auth Success, или Auth Fail в случае неудачи.



Рисунок 16 – Сервер. Подключение клиента

Если все настроено верно, то вскоре клиент начнет передавать координаты (рисунок 17).

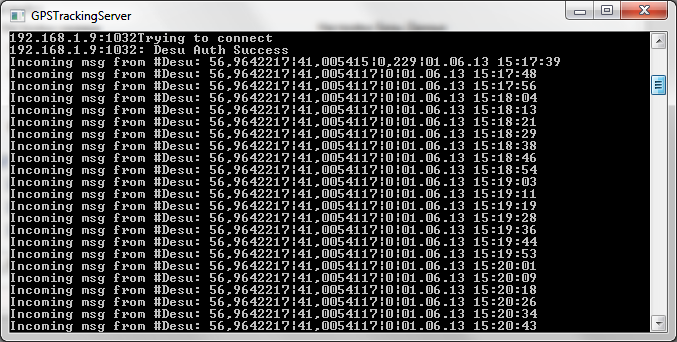


Рисунок 17 – Сервер. Подключение получение данных от клиента

Для просмотра пути надо зайти на сайт и ввести логин/пароль пользователя.

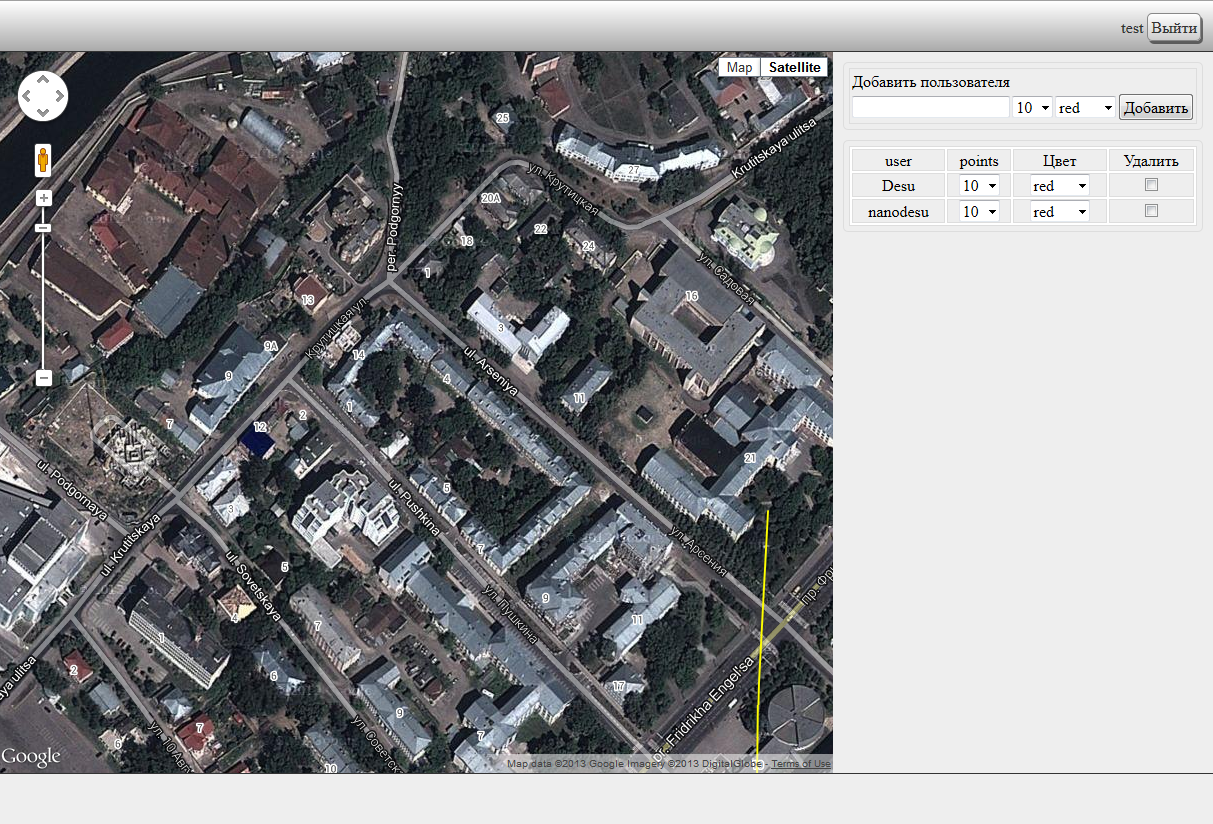


Рисунок 18 – Результат работы скрипта

# 5 РАБОЧАЯ ДОКУМЕНТАЦИЯ

## 5.1 Руководство пользователя

### 5.1.1 Введение

#### 5.1.1.1 Область применения

Система имеет широкую область применения начиная от слежения за личным транспортом или транспортом фирмы, заканчивая созданием онлайн сервиса слежения за устройствами и пользователями. Система может быть использована и для отслеживания пути перемещения туристов или любителей экстремальных видов отдыха для своевременного оказания помощи в случае возникновения непредвиденных обстоятельств.

#### 5.1.1.2 Краткое описание возможностей

* Отображение пути
* Хранение координат
* Разграничение доступа просмотра пути между пользователями.

#### 5.1.1.3 Уровень подготовки пользователя

Администратор системы должен:

* Иметь навыки администрирования баз данных
* уметь настраивать веб-сервер
* Иметь навыки программирования на языке PHP
* Разбираться в предметной области и представлять принцип работы системы

Пользователь системы должен:

* Уметь работать с мобильными приложениями
* уметь пользоваться сетью Интернет

#### 5.1.1.4 Перечень эксплуатационной документации

В состав эксплуатационной документации входит:

* Настоящая Рабочая Документация к системе;
* Рабочая документация к программному обеспечению, обеспечивающему функционирование системы.

### 5.1.2 Назначение и условия применения

#### 5.1.2.1 Программные и аппаратные требования к системе

В данном пункте приведены минимальные требования к аппаратному обеспечению, но с увеличением числа клиентов требования увеличиваются.

Требования к аппаратному обеспечению сервера системы:

* Процессор с тактовой частотой не менее 1,6 ГГц;
* Жесткий диск объемом не менее 20 Гб;
* Оперативная память объемом не менее 1 Гб.

Требования к программному обеспечению сервера системы:

* Операционная система Microsoft Windows Server;
* .Net Framework 4;

Требования к аппаратному обеспечению клиента:

* Процессор с тактовой частотой не менее 800 МГц;
* Свободное место на накопителе 5 Мб;
* Оперативная память объемом не менее 256 Мб.

Требования к программному обеспечению клиента системы:

* Операционная система Windows Mobile 6.5;
* Microsoft Compact Framework 3.5

Требования к аппаратному обеспечению клиента:

* Подключение к сети Интернет
* Высокая производительность при большом числе подключений

Требования к программному обеспечению сервера скрипта отображения пути:

* Веб-сервер с поддержкой PHP и MySQL

Требования сервера базы данных определяются требованиям СУБД MySQL

### 5.1.3 Описание операций

Все операции, производимые в конфигураторе сервера однотипны, типовая схема проведение операций описываются в разделе контрольный пример настоящего руководства. Рассмотрим подробно функцию добавления пользователя.

5.1.3.1 Компонент конфигуратор сервера

5.1.3.1.1 Добавление нового пользователя

5.1.3.1.1.1Условия выполнения операции

Для обеспечения успешного выполнения операции необходимо:

* наличие в базе данных всех необходимых для создания новой записи таблиц (в данном случае таблицы Users);
* заполнение всех обязательных полей;
* корректность всех введенных и добавляемых в базу данных значений.

5.1.3.1.1.2 Подготовительные действия

Подготовительные действия отсутствуют.

5.1.3.1.1.3 Основные действия

Для добавления нового пользователя необходимо запустить конфигуратор сервера(рисунок 12), выбрать в меню пункт управление пользователями, затем выбрать кнопку добавить на панели управления.

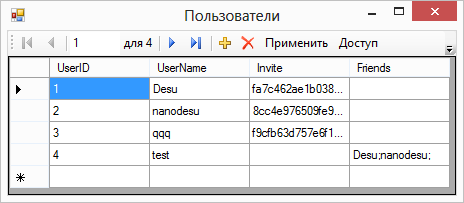


Рисунок 20 – Окно управления пользователями

В качестве основных действий при выполнении данной операции рассматривается ввод всех данных в специально предназначенные для этого элементы на графических формах (рисунок 12):

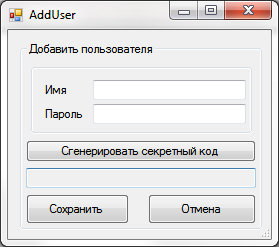


Рисунок 21 – Окно добавления клиента

Для изменения пароля или управления доступом следует выбрать соответствующие кнопки на панели меню.

5.1.3.1.1.4 Заключительные действия

После ввода всех данных при нажатии на кнопку «Сохранить» в случае правильных действий пользователя список клиентов обновится и в конце его появится добавленный клиент. В случае неправильных действий будет выведено окно, сообщающее об ошибке.

5.1.3.2 Компонент сервер

5.1.3.2.1 Получение списка подключенных пользователей

5.1.3.2.1.1Условия выполнения операции

Для обеспечения успешного выполнения операции необходимо наличие хотя бы одного подключенного пользователя.

5.1.3.2.1.2 Подготовительные действия

Подготовительные действия отсутствуют.

5.1.3.2.1.3 Основные действия

Для получения списка пользователей необходимо запустить сервер и ввести команду list.

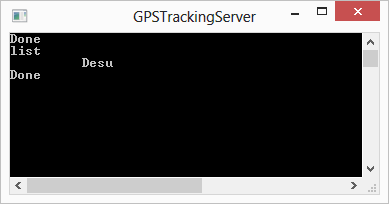


Рисунок 22 – Консоль сервера со списком подключенных пользователей

Список всех команд с описанием можно узнать введя help в консоль сервера.

5.1.3.2.1.4 Заключительные действия

Заключительные действия отсутствуют.

5.1.3.3 Компонент «Клиентское приложение»

5.1.3.3.1 Изменение настроек подключения

5.1.3.3.1.1Условия выполнения операции

Условия отсутствуют.

5.1.3.3.1.2 Подготовительные действия

Подготовительные действия отсутствуют.

5.1.3.3.1.3 Основные действия

Выбрать пункт Settings в основном окне приложения.

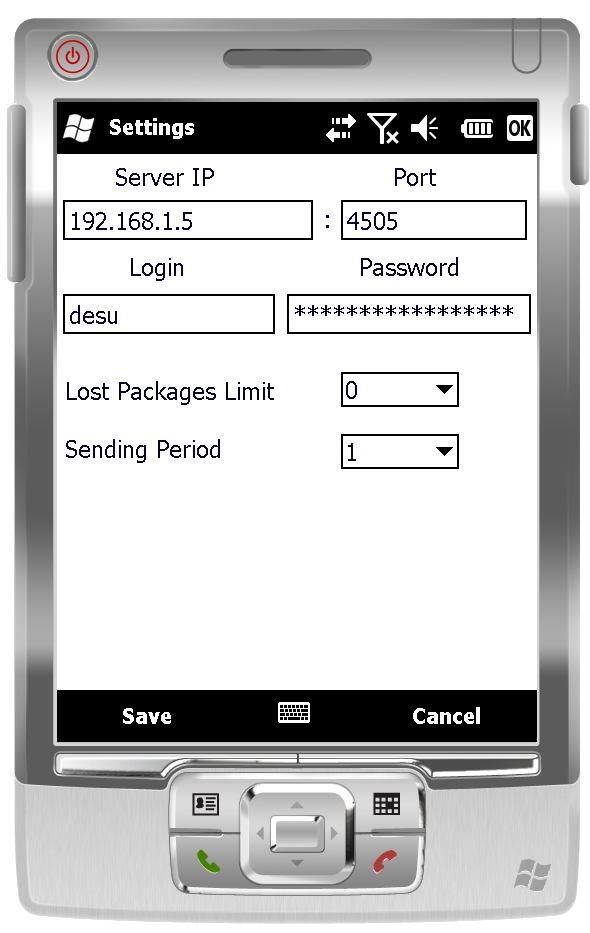


Рисунок 23 – Окно настройки клиентского приложения.

5.1.3.3.1.4 Заключительные действия

Нажать кнопку сохранения настроек (Save).

5.1.3.4 Компонент «web-интерфейс»

5.1.3.4.1 Просмотр пути пользователя

5.1.3.4.1.1 Условия выполнения операции

Должны быть выполнены следующие условия:

* Клиент должен быть зарегистрирован в системе и в системе должны быть координаты минимум двух точек пути
* Компьютер, с которого просматривается путь, должен иметь подключение к сети Интернет.

5.1.3.4.1.2 Подготовительные действия

Настройка скрипта:

Откройте \files\DB\DBConnection.php текстовым редактором и измените настройки подключения к базе данных.

Файлы скрипта должны быть размещены на хостинге с поддержкой PHP

Пользователь должен авторизоваться в системе т.е. ввести пару логин/пароль в соответствующие поля. (см. рисунок 24 правый верхний угол)

5.1.3.4.1.3 Основные действия

Выбрать пункт «карты» в меню web-интерфейса и выбрать карту, на которой требуется отобразить путь (рисунок 18).

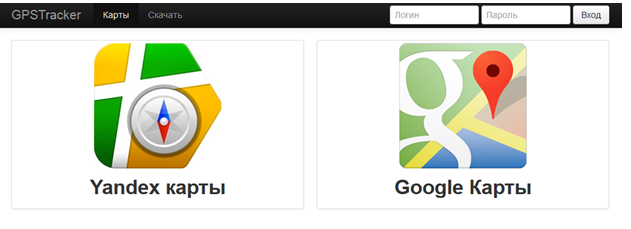


Рисунок 24 – Окно выбора карты

5.1.3.4.1.4 Заключительные действия

Заключительные действия отсутствуют.

# ЗАКЛЮЧЕНИЕ

Данная система не претендует на инновационность, но благодаря абстрактной постановке задачи может стать универсальным решением для многих задач. Благодаря гибкости системы ее легко масштабировать и модернизировать под потребности конкретного потребителя, что позволит быстро внедрить ее на малые предприятия, затратив минимум ресурсов для этого.

При наличии промышленных GPS-приемников и датчиков данный проект сможет стать конкурентом систем описанных в пункте 2.1.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.602-89 Информационная технология. Техническое задание на создание автоматизированной системы

2. ГОСТ 34.601-90 Информационная технология. Автоматизированные системы. Стадии создания.

3. Требования к оформлению квалификационных работ: метод.указания для студентов по направлению 230200 «Информационные системы» / Сост.: А.П.Власов, Н.А. Марчук: Иван. гос. хим.-технол. ун-т. – Иваново, 2010, 35 с.

1. Требования к содержанию квалификационных работ: метод.указания для студентов по направлению 230200 «Информационные системы» / сост.: А.П.Власов, С.П.Бобков, Н.И.Терехин: Иван. гос. хим.-технол. ун-т. – Иваново, 2010.- 40 с.
2. Леоненков А.В. Нотация и семантика языка UML–http://www.intuit.ru/department/pl/umlbasics (5.03.2013) Сайт «intuit.ru»
3. Проектирование информационных систем http://www.intuit.ru/department/itmngt/designis/class/free/status/ (4.03.2013) Сайт «intuit.ru»
4. Система контроля транспорта Сitypoint -http://www.citypoint.ru/functions.htm (15.06.2013) сайт «citypoint.ru»
5. Спутниковая система мониторинга транспорта «АВТОТРЕКЕР» http://www.autotracker.ru/system/ (15.06.2013) Сайт «autotracker.ru»
6. Спутниковые системы мониторинга GEOSTRON - http://www.geostron.ru/ (15.06.2013) Сайт «geostron.ru»
7. Библиотека MSDN http://msdn.microsoft.com/ru-ru/library/ (15.06.2013) Сайт «msdn.microsoft.com»
8. Комплексная система мониторинга ANITOR - http://www.antor.ru/products/kompleksnaya-sistema-monitoringa/ (15.06.2013) Сайт «antor.ru»
9. Интернет-энциклопедия “Википедия” –

http://ru.wikipedia.org (15.06.2013) Сайт «wikipedia.org»

1. Электроника.Микроконтроллеры. Статья GPS-модуль GMM-1 -http://radioelektr.ru/gps-moduly-gmm-u1-ot-globaltop/ (15.06.2013) Сайт «radioelektr.ru»

# ПРИЛОЖЕНИЕ А. ПРОТОКОЛ NMEA0183

NMEA («National Marine Electronics Association») — полное название «NMEA 0183» — текстовый [протокол](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) связи морского (как правило, навигационного) оборудования (или оборудования, используемого в поездах) между собой. Стал особенно популярен в связи с распространением [GPS приёмников](http://ru.wikipedia.org/wiki/GPS-%D0%BF%D1%80%D0%B8%D1%91%D0%BC%D0%BD%D0%B8%D0%BA), использующих этот стандарт. (В настоящее время внедряется стандарт [NMEA 2000](http://ru.wikipedia.org/wiki/NMEA_2000))

* символ «$».
* 5-буквенный идентификатор сообщения. Первые две буквы — идентификатор источника сообщения, следующие три буквы — идентификатор формата сообщения, согласно протоколу NMEA 0183 определённой версии.
* список данных (буквы, цифры и точки), разделённых запятыми. Если какие-либо данные отсутствуют внутри строки, запятые всё равно ставятся (например «,,»). Некоторые поля в конце строки могут отсутствовать вовсе.
* символ «\*».
* 2-значное 16-ричное число — контрольная [XOR](http://ru.wikipedia.org/wiki/XOR)-сумма всех байт в строке между «$» и «\*».
* <CR><LF> ([конец строки](http://ru.wikipedia.org/wiki/ASCII#.D0.A3.D0.BF.D1.80.D0.B0.D0.B2.D0.BB.D1.8F.D1.8E.D1.89.D0.B8.D0.B5_.D1.81.D0.B8.D0.BC.D0.B2.D0.BE.D0.BB.D1.8B)).

Полный набор NMEA-сообщений и команд достаточно велик. Как правило, NMEA-сообщения посылаются с интервалом в 1 [секунду](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%83%D0%BD%D0%B4%D0%B0) при скорости обмена данными 9600 бод.

Иногда производители приёмников спутниковой навигации и др. устройств с выходом NMEA добавляют собственную информацию, т.е. проприетарные предложения, которые несут дополнительную информацию о работе устройства.

Проверка контрольной суммы предложения почти всегда может быть настроена в приемнике информации.

Каждая строка начинается с символов $GP, далее идёт название строки, данные, а после всех данных стоит \*, после которой идёт контрольная сумма. Данные разделяются между собой запятыми. Всего таких строчек 7:

1. PGGA,064951.000,2307.1256,N,12016.4438,E,1,8,0.95,39.9,M,17.8,M,,\*65

* GGA – название строки. Строка GGA содержит в себе время по UTC, позицию и данные о достоверности координат.
* 064951.000 – время по UTC. 06:49:51.000
* 2307.1256 – широта.
* N – указывает, северная широта или южная. N – северная, S – южная.
* 12016.4438 – долгота.
* E – указывает, западная или восточная долгота. E – восточная, W – западная.
* 1 -  указатель, говорящий о том, зафиксирована ли позиция. 0 – позиция неизвестна, 1 – позиция зафиксирована, 2 – позиция зафиксирована и используются данные DGPS.
* 8 – количество используемых спутников. От 0 до 14
* 0.95 – горизонтальная точность в метрах(HDOP).
* 39.9 – высота над уровнем моря.
* M – единицы измерения высоты. Метры.
* 17.8 – геоидальное различие.
* M – метры.
* Время, прошедшее после последнего обновления данных DGPS. Отсутствует.
* 65 – Контрольная сумма.

2. $GPGSA,A,3,29,21,26,15,18,09,06,10,,,,,2.32,0.95,2.11\*00

* GSA – содержит информацию о точности определения координат и активных спутниках.
* A – режим переключения между 2D/3D фиксацией. A – автоматический, M – Ручной.
* 3 – режим фиксации. 1 – фиксация недоступна, 2 – 2D, 3 – 3D.
* 29 – номер спутника, сигнал которого модуль принимает по 1 каналу.
* то-же самое для второго канала
* …
* то-же самое для 12-го канала.
* 2.32 – точность определения местоположения в метрах(PDOP)
* 0.95 – горизонтальная точность в метрах (HDOP)
* 2.11 – вертикальная точность в метрах(VDOP)
* 00 – Контрольная сумма.

3.$GPGSV,3,1,09,29,36,029,42,21,46,314,43,26,44,020,43,15,21,321,39\*7D

* GSV – содержит информацию о видимых и используемых спутниках, их положении, качестве сигнала.
* 3 – количество сообщений. Вся инфа не влезает в одно сообщение, поэтому приходят три сообщения.
* 1 – номер сообщения. От 1 до 9.
* 09 – количество видимых спутников.
* 29 – номер спутника
* 36 – высота спутника в градусах.
* 029 – азимут истинный(спутника относительно антенны) в градусах.
* 42 – отношение сигнал/шум(0-99). Если 0 – спутник не используется.
* То-же самое для остальных спутников…
* 7D – Контрольная сумма.

4.$GPRMC,064951.000,A,2307.1256,N,12016.4438,E,0.03,165.48,260406,3.05,W,A\*2C

* RMC – самое важное сообщение. Содержит информацию о местоположении, времени, дате, скорости.
* 064951.000 – время по UTC. 06:49:51.000
* A – достоверность данных. A – данные достоверны, V- данные не достоверны.
* 2307.1256 – широта.
* N – указывает, северная широта или южная. N – северная, S – южная.
* 12016.4438 – долгота.
* E – указывает, западная или восточная долгота. E – восточная, W – западная.
* 0.03 – скорость относительно Земли(узлов в час).
* 165.48  - курс(путевой угол).
* 260406 – дата. 24.04.06.
* 3,05,W – магнитное склонение. Отсутствует.
* A – режим работы модуля. A – автономный, D – с использованием DGPS.
* 2C –   Контрольная сумма.

5.$GPVTG,165.48,T,,M,0.03,N,0.06,K,A\*37

* VTG – информация о курсе и скорости относительно Земли.
* 165.48 – курс(путевой угол).
* T – данные достоверны.
* M – точно не знаю что это за параметр, но в модуле GMM-U1 он недоступен.
* 0.03,N – скорость(узлов в час).
* 0,06,K – скорость(Км/Ч)
* A – режим работы модуля. A – автономный, D – с использованием DGPS.
* 37 -  Контрольная сумма.

6.$GPGLL, 2307.1256,N,12016.4438,E,182134.000,A,A\*57

* GLL – информация о местоположении, времени определения координат.
* 2307.1256 – широта.
* N – указывает, северная широта или южная. N – северная, S – южная.
* 12016.4438 – долгота.
* E – указывает, западная или восточная долгота. E – восточная, W – западная.
* 182134.000 – время по UTC на момент определения координат.    18:21:34.000.
* A – достоверные данные.
* 57 – Контрольная сумма.

7. $GPZDA,182855.000,08,09,2011,,\*56

* ZDA – информация о времени по UTC, дате и локальном часовом поясе.
* 182855.000 – время по UTC.   18:28:55.000.
* 08,09,2011 – дата.  08/09/2011.
* Локальный часовой пояс(смещение относительно UTC). В этом модуле почему-то отсутствует :(.
* 56 -  Контрольная сумма.

# ПРИЛОЖЕНИЕ Б. Фрагмент ИСХОДНого КОДа

Компонент Сервер

Листинг файла Server.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Timers;

using ConfigurationLibrary;

namespace GPSTrackerServer

{

class Server

{

public bool IsRun { get; set; }

public static Configuration cfg = null;

FillConfiguration fcfg = new FillConfiguration();

// Сокет для принятия подключений

private Socket Sock;

private SocketAsyncEventArgs AcceptAsyncArgs;

// Список клиентов

private List<ClientConnection> Clients = new List<ClientConnection>();

public Server()

{

Console.Title = "GPSTrackingServer";

cfg = fcfg.ReadConfigFile();

Sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

AcceptAsyncArgs = new SocketAsyncEventArgs();

AcceptAsyncArgs.Completed += AcceptCompleted;

System.Timers.Timer timer = new System.Timers.Timer(cfg.KeepAliveTime\*100);

timer.Elapsed += KeepAlive;

timer.Enabled = true;

IsRun = false;

}

/// <summary>

/// Запускает сервер

/// </summary>

public void Start()

{

IsRun = true;

Sock.Bind(new IPEndPoint(IPAddress.Any, cfg.Port));

Sock.Listen(cfg.MaxClients);

AcceptAsync(AcceptAsyncArgs);

Output.Write("Server started on port" + cfg.Port, 2);

//Console.WriteLine("Server started on port {0}", cfg.Port);

}

/// <summary>

/// Отправляет сообщение Keep Alive по таймеру

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void KeepAlive(object sender, ElapsedEventArgs e)

{

SendToAll("Still Alive =)");

}

private void AcceptCompleted(object sender, SocketAsyncEventArgs e)

{

if (e.SocketError == SocketError.Success)

{

ClientConnection Client = new ClientConnection(e.AcceptSocket);

Client.AuthorizationFaild += new ClientConnection.ConnectionEvent(Client\_AuthorizationFaild);

Client.AuthorizationSuccess += new ClientConnection.ConnectionEvent(Client\_AuthorizationSuccess);

Client.Disconnected += new ClientConnection.ConnectionEvent(Client\_disconnected);

Output.Write(e.AcceptSocket.RemoteEndPoint + "Trying to connect", 2);

//Console.WriteLine("{0} Trying to connect", e.AcceptSocket.RemoteEndPoint);

}

e.AcceptSocket = null;

if(Clients.Count < cfg.MaxClients) AcceptAsync(AcceptAsyncArgs);

}

/// <summary>

/// Обработчик события оключения клиента

/// </summary>

/// <param name="sender">Клиент</param>

/// <param name="message">Сообщение о причине отключения</param>

void Client\_disconnected(ClientConnection sender, string message)

{

Clients.Remove(sender);

Output.Write(message, 2);

}

/// <summary>

/// Обработчик успешной авторизации клиента

/// </summary>

/// <param name="sender">Клиент</param>

/// <param name="message">Сообщение об успешной авторизации</param>

void Client\_AuthorizationSuccess(ClientConnection sender, string message)

{

Clients.Add(sender);

sender.SendAsync("Auth Success");

Output.Write(message, 2);

}

/// <summary>

/// Обработчик неудачной авторизации клиента

/// </summary>

/// <param name="sender">Клиент</param>

/// <param name="message">Сообщение о неудачной авторизации</param>

void Client\_AuthorizationFaild(ClientConnection sender, string message)

{

sender.SendAsync("Auth Failed");

sender.CloseConnection();

Clients.Remove(sender);

Output.Write(message, 2);

}

/// <summary>

/// Принимает асинхронное подключение

/// </summary>

/// <param name="e"></param>

private void AcceptAsync(SocketAsyncEventArgs e)

{

if (IsRun)

{

bool WillRiseEvent = Sock.AcceptAsync(e);

if (!WillRiseEvent) { AcceptCompleted(Sock, e); }

}

}

/// <summary>

/// Отправляет сообщение всем клиентам

/// </summary>

/// <param name="data">Сообщение</param>

public void SendToAll(string data)

{

foreach (ClientConnection Cl in Clients)

{

try

{

Cl.SendAsync(data);

}

catch (Exception ex)

{

Output.Write(ex.Message, 1);

Clients.Remove(Cl);

}

}

}

/// <summary>

/// Возвращает ссылку на объект клиентского подключения по имени пользователя

/// </summary>

/// <param name="\_username">Имя пользователя</param>

/// <returns>Клиентское подключение</returns>

private ClientConnection GetDescriptorByUserName(string \_username)

{

lock (Clients)

{

foreach (ClientConnection cl in Clients)

if (cl.ClientName == \_username) return cl;

}

return null;

}

/// <summary>

/// Отправляет пользователю сообщение

/// </summary>

/// <param name="\_username">Имя поьзователя</param>

/// <param name="\_msg">Сообщение</param>

private void Send(string \_username, string \_msg )

{

ClientConnection cl = GetDescriptorByUserName(\_username);

if (cl == null) { Output.Write("Client " + \_username + " not found", 2);}

else { cl.SendAsync(\_msg); }

}

/// <summary>

/// Отключает от серверауказанного пользователя

/// </summary>

/// <param name="\_username">Имя пользователя</param>

/// <returns></returns>

private bool KickUser(string \_username)

{

ClientConnection cl = GetDescriptorByUserName(\_username);

if (cl == null) { Output.Write("Client " + \_username + " not found", 2); }

else { cl.CloseConnection(); Clients.Remove(cl); return true; }

return false;

}

/// <summary>

/// Выводит список пользователей

/// </summary>

private void UsersList()

{

lock (Clients)

{

foreach (ClientConnection cl in Clients)

{

Output.Write("\t " + cl.ClientName, 0);

//Console.WriteLine("\t " + cl.ClientName);

}

}

}

/// <summary>

/// разбивает строку в массив

/// </summary>

/// <param name="inputString">входная строка</param>

/// <param name="separator">разделитель</param>

/// <returns></returns>

private string[] Explode(string inputString, string separator)

{

string[] exploded = new string[2];

inputString = inputString.Trim();

int pos = inputString.IndexOf(separator);

if (pos != -1)

{

exploded[1] = inputString.Substring(pos).Trim();

exploded[0] = inputString.Substring(0, pos).Trim();

}

else

{

exploded[0] = inputString;

}

return exploded;

}

/// <summary>

/// Интерпретатор команд с консоли сервера

/// </summary>

/// <param name="input">команда</param>

/// <returns></returns>

public string Command(string input)

{

string[] result = Explode(input," ");

string command = result[0].ToLower();

string argument = result[1];

switch (command)

{

case "all": { SendToAll(argument); } break;

case "cls": { Console.Clear(); } break;

case "stop":

case "exit":

case "shutdown": { Stop(); } break;

case "kick": { KickUser(argument); } break;

case "list": { UsersList(); } break;

case "send":

{

if (argument != null)

{

result = Explode(argument," ");

string username = result[0];

string message = result[1];

if (message != null) { Send(username, message); }

else { Console.WriteLine("Wrong command format"); }

}

else { Console.WriteLine("Wrong command format"); }

} break;

case "help":

{

Console.WriteLine("List show list of connected users");

Console.WriteLine("Send <username> <message> - sending message to user");

Console.WriteLine("All <message> - sending message to all clients");

Console.WriteLine("Kick <username> - disconnect client");

Console.WriteLine("Cls - clear this screen");

Console.WriteLine("stop/exit/shutdown - shutdown this server");

} break;

default: return "Command not found";

}

return "Done";

}

/// <summary>

/// отключает всех клиентов и останавливает работу сервера

/// </summary>

public void Stop()

{

foreach (ClientConnection Cl in Clients)

{

Cl.SendAsync("Server shutdown");

Cl.CloseConnection();

}

IsRun = false;

Sock.Close();

}

}

}

Компонент клиент

Листинг файла Client.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Net.Sockets;

using System.Net;

using System.Threading;

namespace GPSTrackerClient

{

public class StateObject

{

public Socket workSocket = null;

public const int BufferSize = 32;

public byte[] buffer = new byte[BufferSize];

public StringBuilder sb = new StringBuilder();

}

public class Client

{

public delegate void ConnectionEventDelegate(string status);

public event ConnectionEventDelegate Connected;

Settings settings;

public bool IsConnected { get; private set; }

private ManualResetEvent connectDone;

private ManualResetEvent sendDone;

private ManualResetEvent AuthDone;

private Socket client;

public Client()

{

IsConnected = false;

}

public void Start()

{

settings = new Settings();

connectDone = new ManualResetEvent(false);

sendDone = new ManualResetEvent(false);

AuthDone = new ManualResetEvent(false);

try

{

IPEndPoint remoteEP = new IPEndPoint(IPAddress.Parse(settings.Host), Convert.ToInt16(settings.Port));

client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

client.BeginConnect(remoteEP, new AsyncCallback(ConnectCallback), client);

connectDone.WaitOne();

if (!IsConnected) { return; }

Receive(client);

Send(client,"!" + settings.UserName + "@" + settings.Password);

AuthDone.WaitOne();

}

catch (Exception)

{

Connected("Client inner Error");

}

}

private void ConnectCallback(IAsyncResult ar)

{

try

{

Socket client = (Socket)ar.AsyncState;

client.EndConnect(ar);

client.RemoteEndPoint.ToString());

IsConnected = true;

connectDone.Set();

}

catch (Exception)

{

connectDone.Set();

Connected("Server not found");

IsConnected = false;

}

}

public void Receive(Socket client)

{

try

{

StateObject state = new StateObject();

state.workSocket = client;

client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new AsyncCallback(ReceiveCallback), state);

}

catch (Exception)

{

Connected("Recieve Error");

return;

}

}

private void ReceiveCallback(IAsyncResult ar)

{

try

{

StateObject state = (StateObject)ar.AsyncState;

Socket client = state.workSocket;

int bytesRead = 0;

try

{

bytesRead = client.EndReceive(ar);

}

catch (ObjectDisposedException) { }

StringBuilder response = new StringBuilder();

if (bytesRead > 0)

{

state.sb.Append(Encoding.ASCII.GetString(state.buffer, 0, bytesRead));

if (state.sb.ToString() == "Auth Success") { AuthDone.Set(); }

if (state.sb.ToString() == "Auth Failed") { AuthDone.Set(); }

Connected(state.sb.ToString());

state.sb = new StringBuilder();

client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new AsyncCallback(ReceiveCallback), state);

}

}

catch (Exception)

{

Connected("Can't recieve. Server down");

return;

}

}

public void Send(string \_msg)

{

Send(client, \_msg);

sendDone.WaitOne();

}

private void Send(Socket client, String data)

{

byte[] byteData = Encoding.ASCII.GetBytes(data);

client.BeginSend(byteData, 0, byteData.Length, 0, new AsyncCallback(SendCallback), client);

}

private void SendCallback(IAsyncResult ar)

{

try

{

Socket client = (Socket)ar.AsyncState;

int bytesSent = client.EndSend(ar);

bytesSent);

sendDone.Set();

}

catch (Exception)

{

sendDone.Set();

Connected("Can't send. Server down");

return;

}

}

public void Stop()

{

client.Shutdown(SocketShutdown.Both);

client.Close();

IsConnected = false;

}

}

}