ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное автономное образовательное учреждение  
высшего профессионального образования**

**Национальный исследовательский университет  
«Высшая школа экономики»**

**Московский институт электроники и математики**

**Факультет Прикладной математики и кибернетики**

**Кафедра Кибернетики**

Дипломная работа

по специальности 230201.65 «Информационные системы и технологии»

**Разработка алгоритмов и программного обеспечения подсистемы сбора и обработки цифровой информации КАС СиДИМ**

Студент группы МС91 Резцов А.А.

Руководитель доцент, к.т.н.

Белов А.В.

Зав. кафедрой профессор, д.т.н.

Афанасьев В.Н.

Москва 2014

# Аннотация

Работа изложена на 66 страницах и состоит из введения, трех глав, содержащих 32 параграфов, заключения и библиографического списка, содержащего 3 наименования. В работе содержится 35 рисунков.

**Объектами исследования** являются процессы сбора, обработки и визуального представления навигационной информации.

**Целью работы** является проектирование и разработка подсистемы сбора, обработки и визуального представления навигационной информации.

**Результат** - разработано алгоритмическое, интерфейсное и программное обеспечение подсистемы сбора и визуального представления навигационной информации.

**Новизна полученных результатов** заключается в проектировании и программной реализации подсистемы сбора, обработки и представления навигационной информации.

**Рекомендации по внедрению решений:**

Результаты, полученные в результате решения поставленных в работе задач, будут использованы НППФ "Спектр" при дальнейшей разработке системы работы с навигационными данными.

Оглавление

[Аннотация 2](#_Toc123863104)

[Оглавление 2](#_Toc123863105)

[Введение. 2](#_Toc123863106)

[Глава 1. Анализ современной системы сбора, обработки и доведения до авиационных пользователей навигационной информации. 2](#_Toc123863107)

[1.1. Описание процесса сбора, обработки и доведения до авиационных пользователей навигационной информации. 2](#_Toc123863108)

[1.1.1. Источники навигационной информации. 2](#_Toc123863109)

[1.1.1.2. Общий состав GNSS. 2](#_Toc123863110)

[1.1.2. Потребители навигационной информации 2](#_Toc123863111)

[1.1.3. Принцип работы системы. 2](#_Toc123863112)

[1.2. Анализ существующей системы управления воздушным движением. 2](#_Toc123863113)

[1.2.1. Функциональность системы. 2](#_Toc123863114)

[1.2.2. Программное ядро существующей системы. 2](#_Toc123863115)

[1.2.3. Недостатки существующей системы. 2](#_Toc123863116)

[1.2.4. Требования к новому порталу. 2](#_Toc123863117)

[1.2.5. Постановка задачи. 2](#_Toc123863118)

[Глава 2. Построение информационного и алгоритмического обеспечения подсистемы КАС СиДИМ. 2](#_Toc123863119)

[2.1. Синтез информационно-логической модели подсистемы сбора и обработки информации Real Time Service. 2](#_Toc123863120)

[2.2. Разработка информационной схемы подсистемы работы с новостным контентом системы. 2](#_Toc123863121)

[2.2.1. Алгоритм обработки и представления новостных данных. 2](#_Toc123863122)

[2.3. Рассмотрение возможных архитектур. 2](#_Toc123863123)

[2.3.1. Обоснование выбора архитектуры 2](#_Toc123863124)

[3. Разработка программного обеспечения. 2](#_Toc123863125)

[3.1. Анализ платформ для веб-разработки. 2](#_Toc123863126)

[3.1.1. Краткое описание технологий PHP и ASP.NET. 2](#_Toc123863127)

[3.1.2. Общие принципы работы. 2](#_Toc123863128)

[3.1.3. Языки программирования. 2](#_Toc123863129)

[3.1.4. Среды разработки. 2](#_Toc123863130)

[3.1.5. Безопасность. 2](#_Toc123863131)

[3.1.6. Маленькие и большие проекты. 2](#_Toc123863132)

[3.1.7. Выбор платформы. 2](#_Toc123863133)

[3.2. Разработка визуальной части веб-приложения. 2](#_Toc123863134)

[3.2.1. Разработка интерфейса подсистемы обработки и редактирования новостного контента. 2](#_Toc123863135)

[3.2.2. Разработка пользовательского интерфейса функционального дополнения просмотра новостной ленты. 2](#_Toc123863136)

[3.2.3. Разработка интерфейса Real Time Service. 2](#_Toc123863137)

[3.3. Разработка программного обеспечения. 2](#_Toc123863138)

[3.3.1. Тестирование программного обеспечения. 2](#_Toc123863139)

[3.3.1.1. Тестирование новостного модуля. 2](#_Toc123863140)

[3.3.1.2. Тестирование Real Time Service. 2](#_Toc123863141)

[Заключение. 2](#_Toc123863142)

[Библиографический список. 2](#_Toc123863143)

# Введение.

Основополагающим фактором, влияющим на безопасность авиаперелетов является достоверность данных получаемые со Спутниковых Навигационных Систем. В связи с тем, что информация, передаваемая со спутников, по некоторым причинам(эфемериды, неточные часы спутника, ионосферные ошибки, тропосферные ошибки и т.д.) может быть некорректна, выявилась необходимость в разработке системы, позволяющей нивелировать влияние неблагоприятных факторов.

**Целью работы** является проектирование и разработка подсистемы сбора навигационной информации, ее преобразования и визуального отображения.

**Объектами исследования** являются процессы сбора, обработки и визуального представления навигационной информации, а так же выбор подходящих решений для программной разработки.

**Структура работы.** Работа состоит из введения, 3 глав, содержащих 32 параграфа, заключения и библиографического списка, содержащего 3 наименования.

**В первой главе** проводится анализ современной проблемы обработки навигационной информации, рассматриваются источники и потребители этой информации, проводится анализ существующей системы и разрабатываются основные требования к новому порталу.

**Во второй главе** происходит проектирование информационно-логической модели подсистем сбора, обработки и представления информации для подсистем сервиса реального времени и работы с новостным контентом. Так же проводится анализ возможных архитектур.

**В третьей главе** происходит анализ платформ для веб-разработки, выбор технологии, выбор языка программирования. Так же был разработан интерфейс подсистем работы с новостным контентом и функции реального времени. Было разработано и протестировано программное обеспечение.

**В заключении** были изложены выводы по результатам выполнения работы, определена практическая значимость и определена степень новизны.

# Глава 1. Анализ современной системы сбора, обработки и доведения до авиационных пользователей навигационной информации.

## 1.1. Описание процесса сбора, обработки и доведения до авиационных пользователей навигационной информации.

## 1.1.1. Источники навигационной информации.

Первичным источником навигационной информации является глобальные навигационные спутниковые системы(GNSS). На сегодняшний день можно говорить о следующих GNSS:

* GPS NAVSTAR - система глобального позиционирования, разработанная в США. На данный момент это наиболее востребованная навигационная система.
* ГЛОНАСС - система разработанная в России.
* GALILEO - система, разработанная европейским космическим агентством. Основная причина создания данной системы это возможность независимости от систем GPS и ГЛОНАСС.

## 1.1.1.2. Общий состав GNSS.

Навигационные системы NAVSTAR GPS и ГЛОНАСС состоят из трёх основных  подсистем:

* подсистема космических аппаратов
* подсистема контроля управления
* навигационной аппаратуры потребителя

**Подсистема космических аппаратов.**

Разбитые по группам спутники вращаются в своих средневысотных орбитах, на фиксированном удалении от поверхности Земли. На каждое полушария для получения сигнала в любой точке Земли и в любое время требуется 12 спутников. Таким образом чтобы охватить всю поверхность Земли потребуется 24 спутника. Спутники образуют "сетку" над Землей, таким образом чтобы в любой точке планеты всегда были доступны минимум четыре спутника, а как правило доступно не менее шести. Для обеспечения достаточной избыточности на случай выхода из строя основного спутника имеются так же резервные спутники, которые так же необходимы для увеличения точности позиционирования и увеличения степени покрытия отдельной области. Так же имеется возможность перегруппировки спутников, но осуществляются такие действия очень редко в связи с ограниченностью запаса топлива на борту. Одними из важнейших элементов на борту спутника являются высокоточные атомные часы. Для корректировки работы эталонна, а так же на случай его аварийной замены, в спутнике находится от трех до четырех высокоточных атомных часов.

**Подсистема контроля и управления.**

 Эта система состоит из:

* центра управления навигационной системой со своим мощным вычислительным центром
* развёрнутой сети станций измерения управления и контроля, связанных между собой
* центром управления каналами связи и наземного эталона времени и частоты “атомных часов”, для синхронизации бортовых “атомных часов” спутников (этот эталон более высокоточный,  чем те, что установлены на спутниках).

В задачи подсистемы контроля и управления входят:

* контроль за правильностью работы спутников
* внесение корректировок в параметры орбит
* передача спутникам программ управления и навигационных данных
* прием навигационного сигнала со спутника, его обработка и обмен информацией с центром мониторинга системой
* осуществляемая на главной станции математическая обработка и вычисление корректирующих данных, которые передаются в бортовой компьютер спутника

**Навигационная аппаратура потребителей.**

Включает в себя навигационные приемники и средства обработки, предназначенные для получения сигналов со спутника и вычисления собственных параметров, таких как координаты, скорость и время. [1].

## 1.1.2. Потребители навигационной информации

Навигационные спутниковые системы предназначены для определения координат потребителя, скорости движения, а так же точного времени всех видов потребителей. GPS NAVSTAR и ГЛОНАСС разрабатывались в первую очередь для нужд Министерств обороны. Таким образом существуют первичные и вторичные потребители GNSS: военные ведомства и гражданские пользователи. Естественно абсолютный приоритет использования отдается военным нуждам. Исходя из этого, все действующие в настоящее время спутники передают два вида сигналов: высокой точности для военных потребителей (сигнал закодирован и доступ к нему предоставляется только Министерством обороны) и стандартной точности для гражданских пользователей. Отличительной чертой GNSS является их независимость(полная автономность) и беззапросность(аппаратура спутника не посылает запрос на спутник, а только принимает сигнал).

Гражданские потребители GNSS:

* авиация
* морской и речной транспорт
* геодезия и картография
* строительство
* наземный транспорт
* системы безопасности
* спорт
* сельское хозяйство
* спасательные работы
* частное использование

## 1.1.3. Принцип работы системы.

Задача нахождения своего местоположения пользователем является достаточно сложной, так как для вычисления собственных координат в пространстве необходимо вычислить координаты нескольких спутников, т.е. знать их точное местоположение относительно приёмной аппаратуры. Спутники непрерывно движутся в своей орбитальной плоскости, соответственно их координаты постоянно меняются. Для минимизации вычислительной нагрузки на аппаратуру потребителей и снижения требований, предъявляемых к ней вычисление максимально возможного количества данных происходит в наземном комплексе управления, в котором по результатам данных полученных при наблюдении за спутниками просчитывается прогноз параметров орбиты в фиксированные(опорные) моменты времени и во время сеансов связи передаются на спутник. При наличии данных о прогнозируемых параметрах орбиты и точное местоположение спутника в фиксированной точке можно вычислить координаты спутника в любой произвольный момент времени. Спрогнозированные параметры орбиты и их производные называются –эфемеридами. Набор сведений, применяемых для поиска видимых спутников и выбора оптимального созвездия и, содержащих сведения о текущем состоянии навигационной системы в целом, включая “загрублённые” эфемериды, называются  альманахом. Передатчики, находящиеся на спутнике в беспрерывном режиме на высокой частоте передают навигационные сообщения, содержащие эфемериды с метками времени и альманахом. Пользовательская аппаратура, принимая такое навигационное сообщение и опираясь на заложенный в памяти предыдущий альманах, максимально быстро и точно определяет собственные координаты, при необходимости выводя их на средства отображения информации.

Вычислив координаты спутника, зная точное расстояние от спутника до земли и эталонное время распространения радиосигнала, приёмная аппаратура сможет вычислить расстояние от спутника до пользовательского приёмника, а вычислив расстояние до нескольких спутников, можно будет определить своё местоположение. Вот как это происходит в теории:



Рис. 1. Определение координат по спутникам.

Вычислив расстояние  от спутника № 1 до приёмника, представим сферу, где центром будет спутник № 1.

Вычислив расстояние от приёмника до спутника № 2, представим себе вторую сферу, где центром будет спутник № 2 область. Где эти две сферы пересекутся, и будет областью нашего предполагаемого местонахождения.

Для получения более точных данных нам понадобится  информация о расстоянии до спутника № 3 и одна из двух точек. Место пересечения трёх предполагаемых сфер и будет местом нашего позиционирования. Для устранения неверного решения и одновременного уточнения места позиционирования потребуется чётвертый спутник.

  Решенная нами задачи чисто теоретическая, в жизни все немного сложнее. Например, нельзя отменить такой фактор как влияние тропосферы и ионосферы, где скорость сигнала замедляется, естественные и искусственные препятствия для прохождения радиоволн. Сигнал так же имеет свойство отражаться от поверхности, в связи с чем появляется задача корректировки сигнала от спутников с помощью наземных станций и других технологий. Таки образом хоть для определения координат и необходимо всего 4 видимых спутника, в жизни "лишние" спутники точно не помешают и чем больше сигналов от разных спутников в одно и то же время сможет просчитать приемник, тем более точным будет позиционирование. Но даже при наличии самого современного и мощного приемника гражданского применения, мы можем рассчитывать только на точность в пределах от двух до пяти метров. Тогда как геодезическое оборудование обеспечивает точность до метра, а военная аппаратура до нескольких сантиметров. Все дело в том, что для разных потребителей передаётся разный сигнал и используется совершенно разная аппаратура. Для увеличения степени точности определения координат используются вспомогательные системы функциональных дополнений SBAS и GBAS[2].

**Спутниковые функциональные дополнения (SBAS).**Система включает в себя наземные контрольные станции, распределенные на достаточно обширной территории. Основной задачей ко­торых является выработка корректирующих данных из сигналов спутников и ин­формации о целостности.

Корректирующая информация со всех контрольных станций, а так же информация о состоянии спутников обрабатываются по специальным алгоритмам и в виде сообщения передается на геостационарный спутник. Стоит отметить что эти сообщения содержат в себе такие данные как дифференциальные поправки к сигналам навигационных спутников, информацию о целостности целостности и другую служебную информацию. Далее информация с геостационарный спутника передается на воздушные суда. Стоит отметить, что геостационарный спутник при этом может выполнять роль и навигационного, с помощью которого увеличивается количество видимых спутников и таким образом повышается точность определения местоположения. Одним из примеров реализации SBAS является WААS (Wide Area Augmentation System – Система функционального дополнения с широкой зоной действия) обслуживающая территорию США включая штат Аляска.

**Наземные функциональные дополнения (GBAS)** включает наземную подсистему и бортовую подсистему. Наземная подсистема GBAS передает данные и поправки к дальномерным сигналам GNSS посредством ОВЧ-радиовещательной передачи цифровых данных в бортовую подсистему. Наземная подсистема GRAS состоит из одной или нескольких наземных подсистем GBAS.

GBAS выполняет следующие функции:

* обеспечение локальных поправок к псевдодальности;
* обеспечение данных о системе GBAS;
* обеспечение данных для конечного участка точного захода на посадку;
* обеспечение прогнозирования данных об эксплуатационной готовности дальномерного источника;
* обеспечение контроля целостности источников дальномерных измерений СНС.

Система GRAS предназначена для обеспечения операций на маршруте, в районе аэродрома, заходов на посадку, вылетов и заходов на посадку с вертикальным наведением[3].

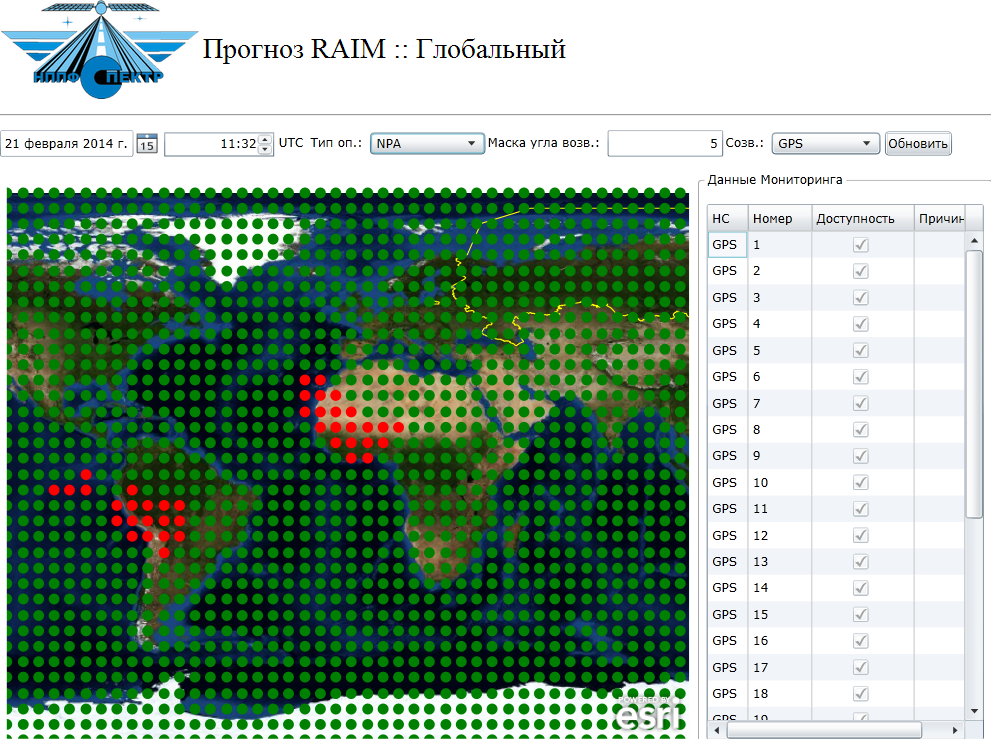
## 1.2. Анализ существующей системы управления воздушным движением.

## 1.2.1. Функциональность системы.

Существующий в настоящее время портал, находящийся по адресу **http://spectr.gkovd.su**, поддерживает следующий функционал:

1. прогноз доступности функции RAIM - позволяет получить информацию о возможности использования функции RAIM на территории всего земного шара;

Рис.2. Страница прогноза доступности функции RAIM для NPA.



1. предсказание доступности RAIM по маршруту - позволяет получить информацию о возможности использования функции RAIM на заданном маршруте движения воздушного судна;



Рис.3. Отображение маршрута на карте.

1. Прогноз RAIM Аэропорт - данная функция позволяет оценить доступность функции RAIM для аэродрома, времени и типа полета, а также используемую орбитальную спутниковую группировку;

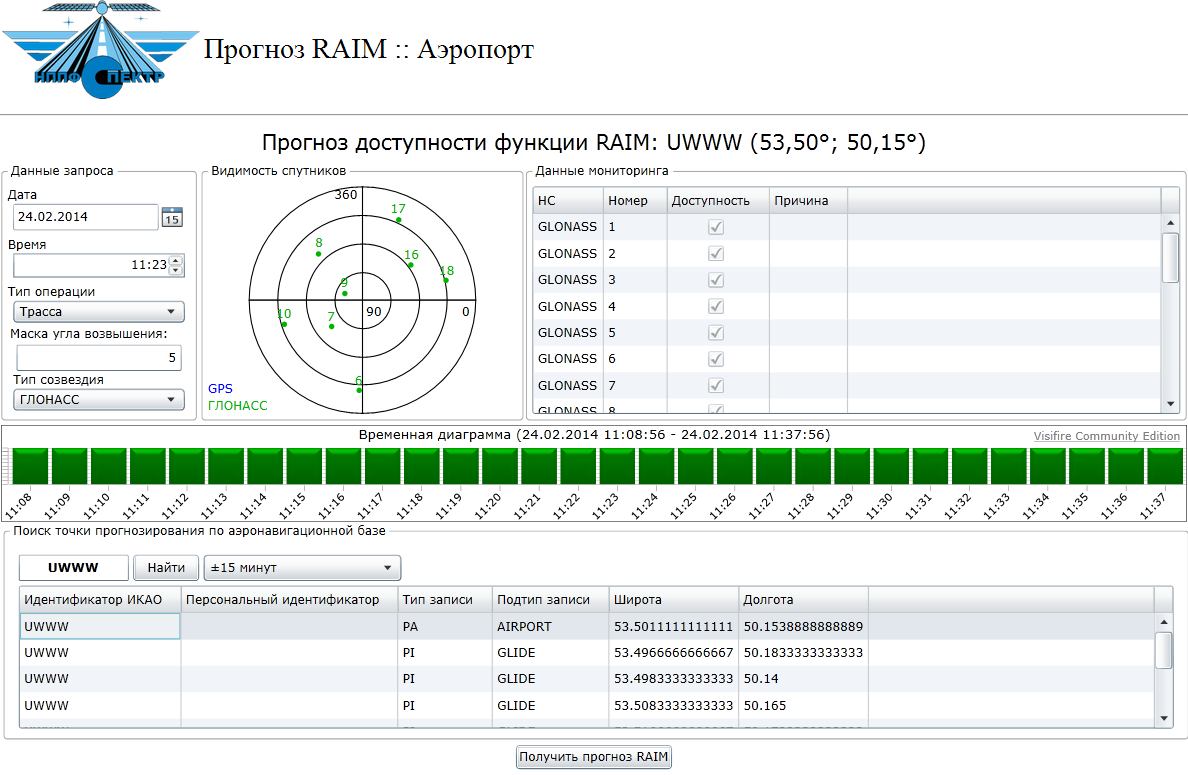


Рис.4. Интерфейс функции RAIM Аэропорт.

**Видимость спутников** – отображает видимые на время запроса навигационные спутники в полярной системе координат азимут-угол возвышения.

**Данные о здоровье спутников** – информация о здоровье спутников по данным альманахов и данным, предоставленным ИАЦ ГЛОНАСС (http://www.glonass-ianc.rsa.ru).

**Временная диаграмма** – отображает доступность функции RAIM для данной точки в интервале полчаса (+/- 15 минут от заданного времени прогнозирования) с дискретностью 1 минута.

1. Функция реального времени - обеспечивает отображения информации о текущем состоянии группировок ГЛОНАСС и GPS и данных спутниковых систем в зоне действия станций мониторинга как при автономном, так и при дифференциальном режиме работы бортового оборудования СНС. Дифференциальный режим отличается от автономного тем, что ВС принимает поправки с ЛККС к данным, получаемым со спутников. Тогда как при автономном режиме ВС использует нескорректированные данные;

Рис.5. Интерфейс функции реального времени.

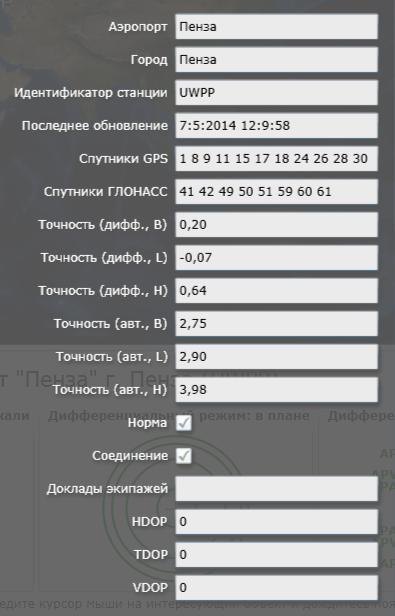
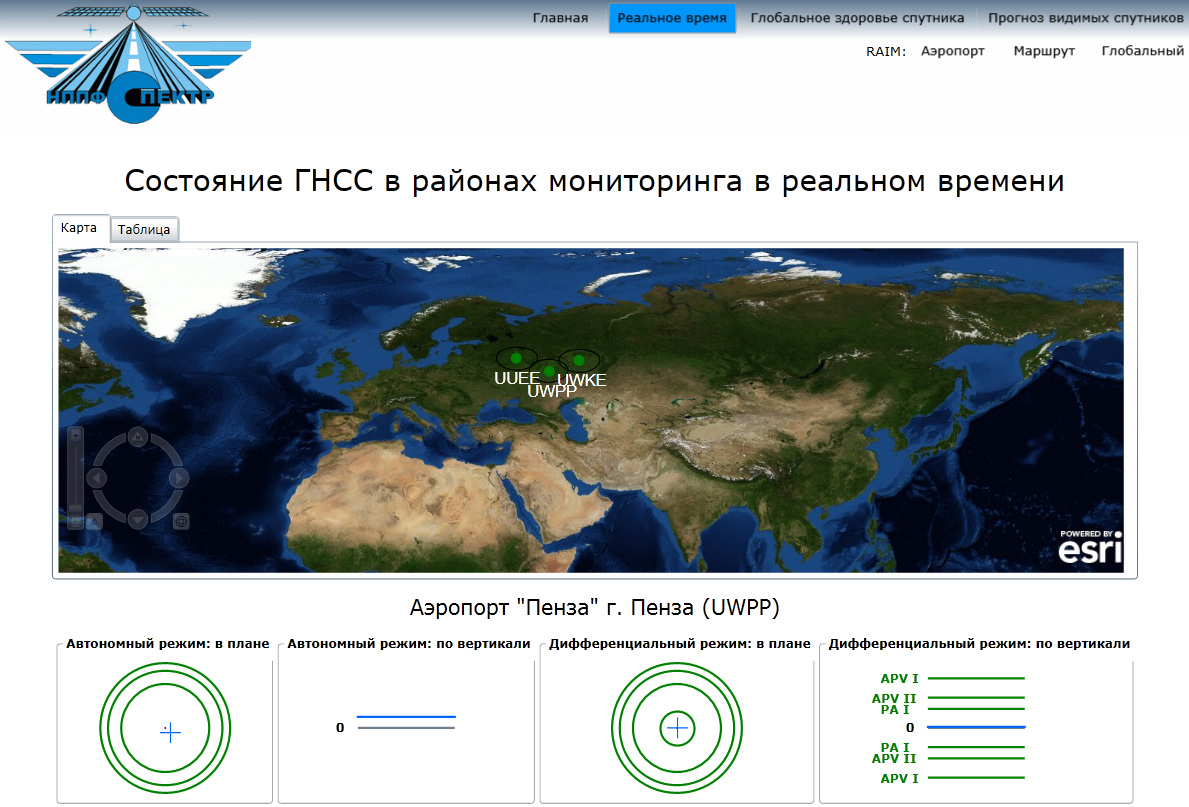


Рис.6. Данные спутниковых систем в зоне действия станций мониторинга.

1. Функция глобального здоровья спутника отображает информации о текущем состоянии выбранного спутника в зоне действия станций мониторинга;

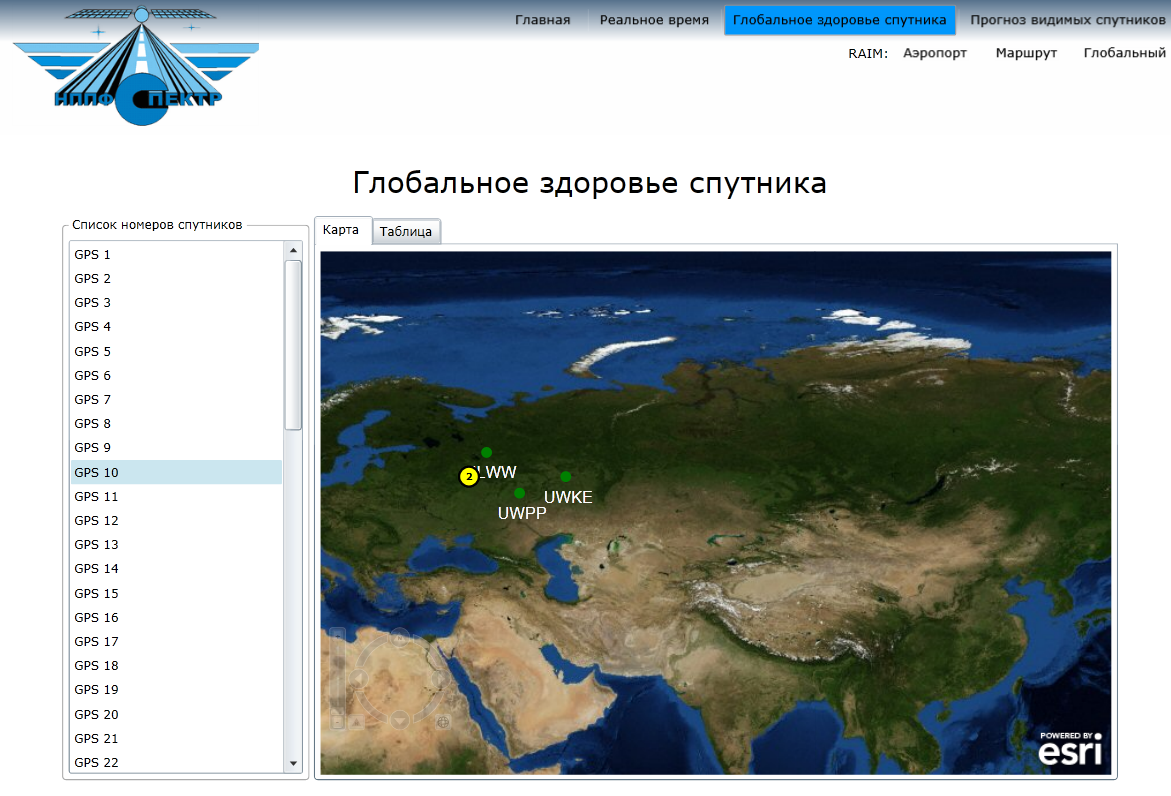
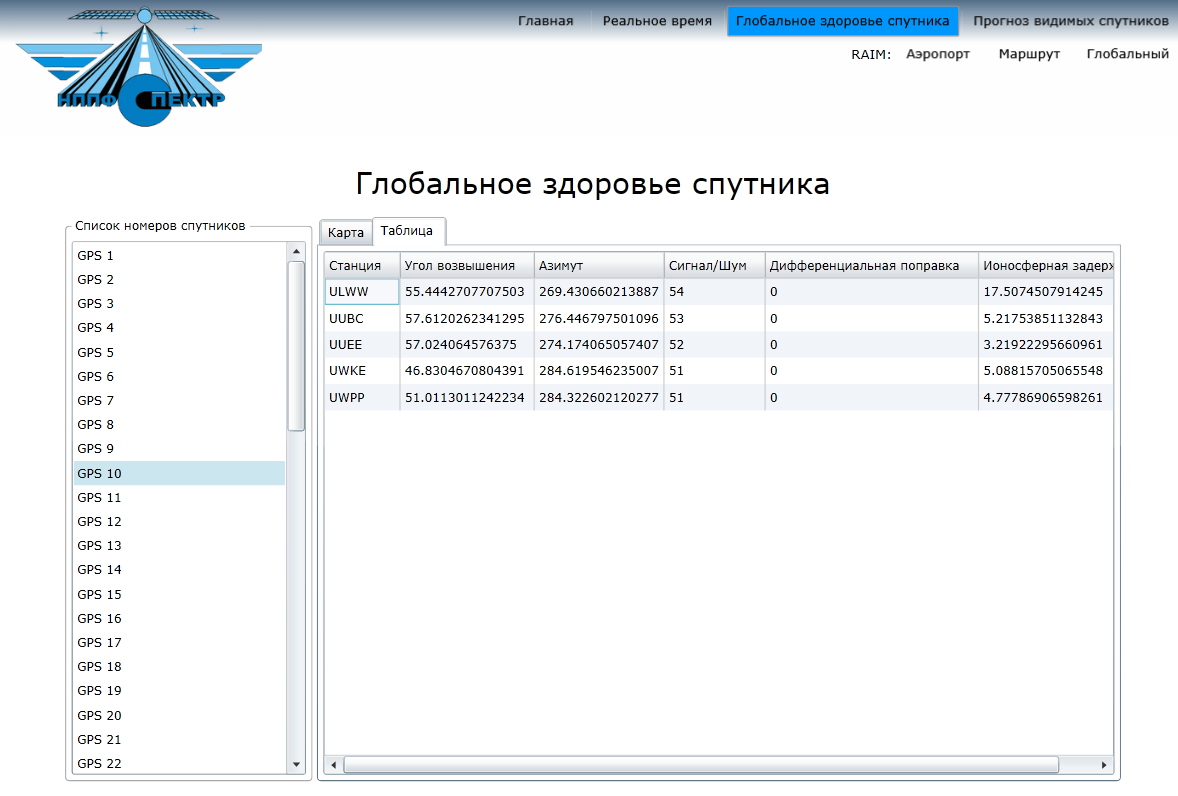


Рис.7. Глобальное здоровье спутника.

Рис.8. Глобальное здоровье спутника.



1. Прогноз видимых спутников предоставляет прогнозную информацию о состоянии готовности дальномерных источников (спутников) для конкретного времени и конкретной станции для орбитальной спутниковой группировки ГЛОНАСС и GPS;

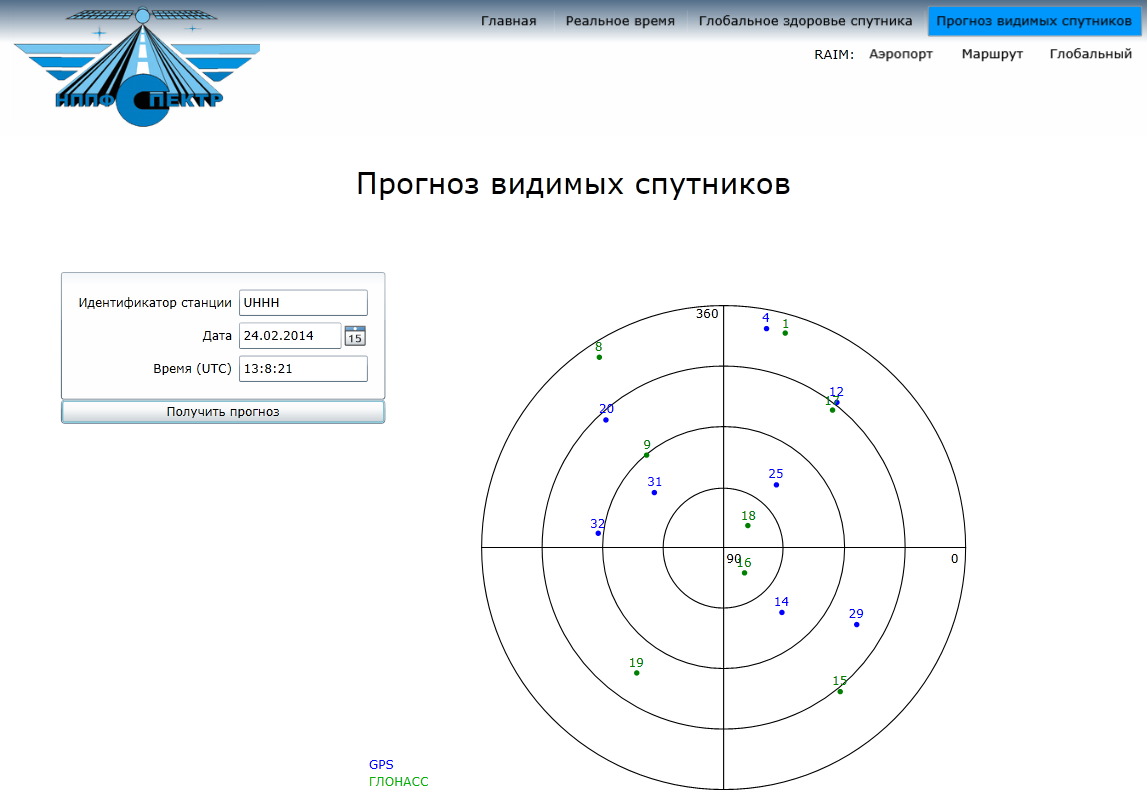


Рис.9. Страница веб-приложения «Прогноз видимых спутников».

## 1.2.2. Программное ядро существующей системы.

Существующий портал реализован на следующем программном обеспечении:

* Система управления базами данных Microsoft SQL Server 2008
* Веб-сервер Microsoft Internet Information Server (IIS)
* Технология Silverlight 3

## 1.2.3. Недостатки существующей системы.

Самым существенным недостатком существующей системы является ее платформа. В связи с тем что даже сама Microsoft отказалась от поддержки Silverlight, с стратегической точки зрения не представляется возможным дальнейшее использование этой технологии.

Так же к минусам технологии Silverlight относится абсолютная невозможность использования ее на мобильных платформах. Известно, что Silverlight не поддерживается на Android и iOS.

Еще одним недостатком существующей системы является ее дизайн. В условиях возможного высокого уровня конкуренции, такой аспект как внешний вид портала не может оставаться без внимания.

## 1.2.4. Требования к новому порталу.

Основное требование к новому порталу это наличие "мультиплатформенности" и соответственно адаптивной верстки для телефонов и планшетов под ОП iOS, android и windows. Необходимо предусмотреть автоматическое определение устройства и представление наиболее подходящей версии.

Функциональная часть сайта должна иметь возможность:

* предоставлять информацию о доступности функции RAIM
* предоставлять информацию органов обслуживания воздушного движения и обновляемой информации о ситуации
* формирование информации для выпуска соответствующих предупреждений и НОТАМ
* поиск информации по реквизитам и контексту
* размещение на сайте информации различными пользователями
* корректную обработку аварийных ситуаций, вызванных неверными действиями пользователя, неверным форматом или недопустимыми значениями входных данных
* выдачу пользователю сообщения об ошибке, после чего возвращаться в рабочее состояние, предшествовавшее неверным действиям пользователя.
* мультиязычность
* при долгих ответах сервера добавить анимацию загрузки

## 1.2.5. Постановка задачи.

В ходе подготовки к написанию моей дипломной работы была поставлена задача разработать программное обеспечение подсистемы сбора информации и последующей ее обработки для сервиса портала под названием Real Time Service. Так же одним из заданий было разработать программную возможность добавления новостного контента в портал.

# Глава 2. Построение информационного и алгоритмического обеспечения подсистемы КАС СиДИМ.

## 2.1. Синтез информационно-логической модели подсистемы сбора и обработки информации Real Time Service.

Мы получаем данные реального времени в виде текстового файла с установленной грамматикой. Каждой позиции в этом файле соответствует определенная переменная. Стоит отметить что такие файлы поступают с периодичностью в 5 секунд.

Так как это данные реального времени не имеет смысла их где либо сохранять и, следовательно, время жизни такого текстового документа 5 секунд.

Упрощенно такой документ имеет вид:

$LCCS0001,CCCI,140121.112425.000

FAIL,1,0,0,0,0,100

STDA,1.4036417710594833,1.9034831612370908,5.615929890424013

DIFF,0.0673663896080692,0.08796123404051824,0.06300383585511607

SAT,1,191.44084137511115,35.31967882499713,22321015.67649441,22321011.76

Расшифруем этот документ:

$LCCS0001, gbasId, dateTime

FAIL,1,0,0,0,0,100

STDA, elatS, elonS, ealtS

DIFF, elatD, elonD, ealtD

SAT, sat\_number, azimuth, angle

gbasId - идентификатор станции

dateTime - дата и время

elatS - отклонение координат по широте в автономном режиме

elonS - отклонение координат по долготе в автономном режиме

ealtS - отклонение координат по высоте в автономном режиме

elatD - отклонение координат по широте в дифференциальном режиме

elonD - отклонение координат по долготе в дифференциальном режиме

ealtD - отклонение координат по высоте в дифференциальном режиме

sat\_number - номер спутника

azimuth - азимут

angle - угол возвышения

Далее эти данные используются для визуального отображения видимых спутников и отклонения координат в автономном и дифференциальном режиме.

Координаты видимых спутников вычисляются по следующим формулам:



- угол возвышения



- азимут



## 2.2. Разработка информационной схемы подсистемы работы с новостным контентом системы.

В соответствии с функциональными требованиями, предъявляемыми к порталу, должна быть реализована возможность добавления новостного контента, его обработки и визуального представления.

Нам понадобится разработать модель базы данных, учитывающую всю специфику работы с новостными потоками.

Для работы с новостной лентой нам нужно две таблицы. В первой будут храниться непосредственно новости, доступ которым можно будет получить на портале, а во второй будут сохраняться все авторы этих новостей.

В новостной таблице, которую мы назовем News, нам понадобятся следующие поля:

* ID - эксклюзивный идентификатор новости
* Title - заголовок новости
* Body - тело новости
* UserCreate - имя пользователя, создавшего новость
* UserUpdate - имя пользователя, дополнившего новость
* DateCreate - дата создания новости
* DateUpdate - дата изменения новости
* AuthorID - эксклюзивный идентификатор автора новости

В таблице, хранящей информацию об авторах предусмотрены следующие поля:

* ID - эксклюзивный идентификатор новости
* Name - имя автора
* UserCreate - имя пользователя, создавшего автора
* UserUpdate - имя пользователя, изменившего автора
* DateCreate - дата создания автора
* DateUpdate - дата изменения автора

## 2.2.1. Алгоритм обработки и представления новостных данных.

Для начала стоит определиться какая информация необходима и целесообразна для представления конечному потребителю. Например такие поля, как ID или UserCreate совершенно необязательны для пользователя новостной составляющей портала. Соответственно, было решено визуально отображать только заголовок и соответственно саму новость.

Работа с новостным потоком сводится к следующему функционалу:

* отображение всех новостей
* отображение выбранной новости
* добавление новости
* удаление выбранной новости
* изменение новости

Опишем стандартный сценарий работы с новостной частью системы:

1. Администратор портала или пользователь с соответствующими правами через специально разработанный модуль добавляет в базу данных информацию о новости, т.е. ее заголовок, саму новость, а так же указывает автора.
2. Новость добавляется в базу данных
3. Автор новости сравнивается с уже существующими авторами. Если такого автора нет, то он добавляется в таблицу авторов и ему присваивается уникальный идентификатор.
4. Новость загружается из базы данных на портал и может быть просмотрена потребителем в новостной ленте.

## 2.3. Рассмотрение возможных архитектур.

При выборе архитектуры прорабатывалась возможность использования толстого и тонкого клиентов. Т.е. было два концептуально различных пути:

* система представляет собой веб-приложение(тонкий клиент)
* система имеет вид программного продукта, который необходимо устанавливать на конкретное устройство(толстый клиент)

Рассмотрим более конкретно каждый из этих подходов.

Тонкий клиент (thin client) представляет собой терминал сети не имеющий жестких дисков. При этом стоит отметить, что вычислительная мощность объем памяти определяются конкретными требованиями пользователей. После регистрации пользователь получает доступ ко всем программам и приложениям, хранящимся на сервере. Тонким клиентом называют также ПК (в том числе и мобильный) с минимизированной мощностью процессора, оперативной и внешней памятью, позволяющий пользователю осуществлять ввод и отображение данных за счет выполнения вычислений и хранения данных на более мощном ПК или сервере, с которыми он может осуществлять связь при помощи каналов средней пропускной способности. К тонкому клиенту могут подключаться внешние устройства ввода/вывода данных (сканеры, мониторы, принтеры и проекторы).

Клиент называется тонким, если он не содержит вовсе или содержит лишь малую часть бизнес-логики, т. е. представляет собой исключительно презентационный слой. Лучший пример тонкого клиента — Web-браузер, настолько универсальный, что способен подключаться к абсолютно разным прикладным программам, о которых "не знает" ничего, и, тем не менее обеспечивать приемлемый интерфейс пользователя. Вся концепция сетевого компьютера строится на идее создания дешевого небольшого устройства, на котором будет работать Web-браузер[4].

Тонкий клиент имеет следующие преимущества:

* Централизация администрирования настольных устройств - за счет централизации ресурсов приложения на сервере, его модификации и изменения в нем автоматически становятся доступными всем пользователям, не требуя от них никаких действий.
* Возможность контроля за действиями пользователя - благодаря отсутствию накопителей на рабочем месте, пользователь не может привносить в конфигурацию программного обеспечения что-то свое, устанавливая собственные программы.
* Мобильность пользователей - пользователь не привязан к конкретному рабочему месту и имеет возможность произвольно перемещаться в пределах локальной сети, применять устройства дистанционного доступа.
* Повышение производительности труда операторов - сведение всех сервисных операций на сервер заметно повышает производительность труда операторов.
* Снижение стоимости эксплуатации оборудования - при не слишком большом различии в стоимости оборудования тонкий клиент заметно дешевле в эксплуатации.

Технология «тонкий клиент-сервер» базируется на трех основных составляющих:

1) стопроцентное выполнение прикладных задач на терминальном сервере,

2) многопользовательская операционная система,

3) технология распределенного отображения пользовательского интерфейса приложений.

Пользователи могут одновременно пользоваться системой и выполнять приложения, хранимые на сервере. При этом у каждого пользователя будет своя защищенная сессия сервера.

Весь принцип использования тонкого клиента сводится к отслеживанию действий пользователя, таких как передвижения мыши или нажатия на ту или иную клавишу и т.д. В свою очередь на сервере производятся соответствующие вычисления и формируется вид экрана пользователя и передаются эти изменения тонкому клиенту. Тонкий клиент получает от сервера измененный вид экрана и отображает их на дисплее. стоит отметить что с современных системах может передаваться не весь вид экрана, а лишь части изображения, дополненные соответствующими командами, с помощью которых программное обеспечение тонкого клиента формирует вид нового экрана. В роли клиента может использоваться любой ПК, но так как на нем практически не выполняются операции бизнес-логики, имеется возможность применения недорогих "низкопроизводительных" терминалов, не содержащих такие компоненты как жесткие диски и вентиляторы. Так же такие терминалы могут иметь весьма ограниченный объем памяти.

Такой факт как хранение всех прикладных программ и данных на сервере дает следующие преимущества:

* отсутствие необходимости установки программного обеспечения на каждом терминале
* удобство проведения резервного копирования данных, т.е. для проведения данной операции необходимо скопировать только содержание сервера
* возможность восстановления прерванной сессий (все пользовательские сессии автоматически сохраняются на сервере).
* ориентация на сотрудников, пользующихся дистанционным доступом.

За счет использования вычислительных мощностей сервера, тонкий клиент имеет высокую производительность даже на рабочих местах не имеющих высоких вычислительных мощностей. Требование повысить производительность всей системы удовлетворяется заменой всего лишь одного устройства - терминального сервера. После такой операции все рабочие места автоматически получают более высокую степень производительности без необходимости замены каких либо устройств[5].

Толстый клиент (Rich client) - это приложение, которое в противоположность тонкому клиенту обеспечивает расширенную функциональность независимо от центрального сервера. Зачастую сервер в этом случае выступает только банком данных, а вся работа по обработке и визуальному представлению этих данных возлагается на машину клиента.

**Достоинства:**

* Толстый клиент обладает широким функционалом в отличие от тонкого.
* Режим многопользовательской работы.
* Предоставляет возможность работы даже при обрывах связи с сервером.
* Имеет возможность подключения к банкам без использования сети Интернет.
* Высокое быстродействие.

**Недостатки:**

* Большой размер дистрибутива.
* Многое в работе клиента зависит от того, для какой платформы он разрабатывался.
* При работе с ним возникают проблемы с удаленным доступом к данным.
* Довольно сложный процесс установки и настройки.
* Сложность обновления и связанная с ней неактуальность данных[6].

## 2.3.1. Обоснование выбора архитектуры

При выборе архитектуры основными мотивами, влияющими на концептуальную направленность системы, были:

* как можно сильнее облегчить пользование системой для авиационных потребителей, т.е. освободить их от возможности что-то устанавливать и настраивать
* обеспечить пользователю мобильность
* разработать возможность работы одним пользователем на различных устройствах

Очевидно, что для этих целей больше подходит архитектура тонкого клиента, когда в роли клиента выступает обычный веб-браузер.

# 3. Разработка программного обеспечения.

## 3.1. Анализ платформ для веб-разработки.

В качестве платформы для разработки веб-портала рассматривались две самые распространенные технологии PHP и ASP.NET.

## 3.1.1. Краткое описание технологий PHP и ASP.NET.

PHP (Hypertext Preprocessor) – язык для написания серверных сценариев (скриптов). Интерпретатор языка бесплатен, с открытым исходным кодом, созданы версии для различных веб-серверов – прежде всего, для Apache и Internet Information Services.

ASP.Net (Active Server Pages .Net) – средство для разработки веб-приложений от Microsoft. ASP.Net – это не язык программирования, это технология, включающая в себя множество компонент. Стоит заметить, что ASP.Net кардинально отличается от своего предшественника – ASP. Принцип работы ASP.Net-приложения совершенно иной и будет рассмотрен далее.

Сразу стоит ответить на вопрос, касающийся возможности сравнения этих совершенно разных технологий. Дело в том что они хоть и отличаются концептуально, но решают одну и ту же задачу - создание веб-сайтов, и неважно какими средствами это достигается.

## 3.1.2. Общие принципы работы.

PHP является серверным языком программирования и осуществляет связь сайта с сервером и его базой данных. Сам по себе язык PHP за создание сайта не отвечает, и если взглянуть на код страницы такого ресурса, мы увидим все тот же HTML, только с вкраплениями PHP кода[7].

В отличие от PHP, принцип работы ASP.NET совершенно иной – каждая страница состоит из двух файлов: файла с HTML-разметкой и ASP.Net-контролами (файл с расширением Aspx) и файла серверной логики (расширения vb или cs). Идея такова: в файле разметки – только разметка (фактически, описание внешнего вида страницы), в файле логики – только логика (программный код). И хотя можно и добавлять программный код в файл разметки, и в файле логики формировать HTML-разметку (фактически, использовать подход, аналогичный PHP), поступать так не рекомендуется – нарушается общая идея – отделение внешнего вида страницы от логики.

## 3.1.3. Языки программирования.

PHP сам по себе является языком программирования. Используется си-подобный синтаксис, отсутствует типизация переменных, в последнее время значительно улучшена поддержка объектно-ориентированного программирования.

При создании веб-приложений ASP.Net можно использовать любой язык из платформы .Net – C# или Visual Basic.Net. Доступна вся функциональность библиотеки .Net Framework.

На мой взгляд, здесь со значительным перевесом выигрывает ASP.Net.   
Во-первых, по возможностям PHP далеко позади от .Net-языков. Например, хотя в PHP5 (который вышел лишь летом 2004 года) и была значительно улучшена поддержка объектно-ориентированного программирования, но она по-прежнему слабее, чем в .Net, где языки (по крайней мере, C# и VB.Net) являются полностью объектными. В них нельзя создать просто функцию – только метод класса, нельзя создать просто переменную – только атрибут класса и т.п. В PHP нет перечислений (enum), нет событий («ручное создание» нельзя считать событиями, т.к. фактически встроенной возможности программирования с использованием модели событий нет), нет возможности перегружать методы (вследствие отсутствия типизации в языке) – это лишь некоторые неприятные особенности, с которыми приходится сталкиваться, используя PHP после серьезных объектно-ориентированных языков. В PHP нет поддержки многопоточности, есть проблемы с поддержкой Unicode (она есть, но реализована неудобно для разработчика, эту проблему обещают исправить в PHP6).  
Во-вторых, PHP гораздо менее строгий язык, чем .Net-языки. В первую очередь – из-за отсутствия типизации. Переменные объявлять необязательно. В результате – множество опасных конструкций, которые не скомпилируются даже в компиляторе Си, в PHP будут выполнены, с непредсказуемым результатом. Некоторые из ошибок (например, неинициализированные переменные) можно увидеть, если включить режим отображения всех ошибок (error\_reporting=all), но об этом нужно знать и помнить. Если человек начинает изучение программирования с PHP, то далеко не факт, что он будет информирован о необходимости «дополнительной настройки». К тому же, часто встречается такое, что начинающему программисту просто надоедают постоянные Notice об ошибках, и он их отключает. Часто в таких случаях сайт по-прежнему работает, и, возможно, даже правильно работает. Но рано или поздно возникнет такая ситуация, когда пропущенная ошибка приведет к серьезным последствиям (взлом, некорректность работы и т.п.). В .Net-языках такое сделать не получится: пока не будет исправлена ошибка – программа не скомпилируется.   
Кроме того, в PHP нет исключений, выкидываемых самим интерпретатором. Например, если в .Net вы попытаетесь обратиться к несозданному объекту (ссылка есть, а new вызвать забыли), то у вас произойдет исключение. В PHP в лучшем случае вы увидите предупреждение (если вы не забыли настроить их отображение). Кардинальное отличие заключается в том, что исключение вы не пропустите никак – вместо ожидаемой страницы вы получите информацию о том, что возникла ошибка. В PHP к исходной странице где-нибудь будет добавлено сообщение с предупреждением. Его можно легко не заметить – например, если сообщение появится в малозаметном месте, или внизу страницы, куда вы редко заглядываете и т.п.  
Даже если PHP -программист – опытный и знает обо всех тонкостях, описанных выше, по-прежнему существует человеческий фактор. Например, расстался разработчик вчера с девушкой, но работать надо. Однако во время работы программист не сконцентрирован на программе, мысли совсем о другом – повышается вероятность того, что он забудет где-то что-то проверить. Во многих случаях это может остаться незамеченным – интерпретация в PHP произойдет успешно. А в ASP.Net ошибку не пропустить, т.к. либо не пройдет этап компиляции, либо произойдет исключение.  
PHP дает разработчику столько вольностей (потенциально опасных), что я бы крайне не рекомендовал изучение программирования (именно программирования, а не веб-разработки) с PHP. Лучше всего начать с серьезного типизированного языка (например, тот же Си), а потом уже переходить на PHP, имея привитые полезные привычки (всегда объявлять и инициализировать переменную, не игнорировать предупреждения интерпретатора и т.п.).  
То, что о чем в PHP нужно всегда помнить, в .Net-языках за вас проверяет компилятор. Пока не будет исправлена ошибка – программа не скомпилируется.  
Коснусь вопроса производительности. В ASP.Net исходный код компилируется единожды в исполняемый байт-код, в PHP – интерпретируется каждый раз. Согласно этому, ASP.Net должен работать быстрее, однако реальные тесты показывают примерно равную производительность. В некоторых случаях какая-то из технологий вырывается вперед, однако в целом можно сказать, что в этом вопросе технологии примерно равнозначны. К тому же, нельзя забывать о том, что время генерации страницы – это еще и время выполнения запросов в базу данных, которое может быть на порядок больше времени интерпретации и не зависит от используемого языка программирования.

## 3.1.4. Среды разработки.

Под язык PHP существует множество сред разработки: родная Zend Studio, версия Eclipse для PHP, плагин под Visual Studio и т.п. Все они являются довольно развитыми средами с поддержкой множества современных возможностей сред разработки.  
 Основной продукт, который используется для разработки веб-приложений на ASP.Net – это Microsoft Visual Studio. Есть и сторонние средства разработки, но они получили гораздо меньшее распространение.

Вообще среда разработки – это дело вкуса и привычки. По большому счету, все продукты как для PHP, так и для ASP.Net имеют встроенную подсказку (Intellisense), возможность доступа к базе данных, возможность отладки и т.п.  
 Однако несколько различий все же имеется. В разработке любых приложений очень помогает Intellisense, если ей научиться правильно пользоваться. Но в PHP она сама по себе не может так быть широко использована, как в ASP.Net – опять же, потому, что PHP – не типизированный язык. Например, у вас есть переменная-объект, которую вы получили не вызовом New, а вызовом метода, который вернул этот объект. В случае с ASP.Net известно, какого типа объект возвращает каждый метод, и Intellisense далее сможет подсказать вам атрибуты и методы у созданного объекта. В PHP такое неосуществимо: нет типизации – значит неизвестно, данные какого типа возвращает метод. Конечно, это можно в комментариях указать принудительно. Но это требует лишней работы, лишнего времени – общая производительность программиста снижается.

При ближайшем рассмотрении Zend Studio выявляется ее недостатки, такие как:

* возможные трудности при установке
* медленная работа подсказки Intellisense

Таким образом использование плагина для работы с PHP под Visual Studio представляется более целесообразным.

Резюмируя, можно сказать что в целом ситуация со средами разработки под обе платформы схожа, за исключением некоторых описанных выше нюансов.

## 3.1.5. Безопасность.

Весьма распространен миф о том, что все продукты от Microsoft – «дырявые», а их аналоги на Unix-платформе – безопасные. На самом деле, к безопасности в Microsoft относятся очень серьезно (об этом свидетельствуют некоторые их интервью и репортажи), основная причина возникновения мифа – широкая распространенность их продуктов, что обуславливает гораздо больший интерес к этим продуктам для злоумышленников.  
Вопреки всему, по данным Securitylab.ru, веб-сервер Internet Information Services имеет гораздо меньше уязвимостей, чем его Unix-аналог – Apache. Так, например, в IIS 6.0 было найдено всего 3 уязвимости!  
Однако гораздо больше случаев взлома веб-сайтов происходит из-за ошибок разработчиков этих сайтов. И хотя вероятность успешного взлома определяется в основном квалификацией программиста, в этом аспекте имеются некоторые преимущества у ASP.Net – более жесткий контроль вводимых посетителем данных. Например, по умолчанию включен запрет на использование HTML-тегов в полях ввода – таким образом, повышается защищенность веб-сайта от XSS (Cross Site Scripting) атак. В PHP такие проверки разработчику необходимо реализовывать самостоятельно. Это особенно важно для начинающих веб-программистов, которые, как правило, в начале своего пути не знают о возможных уязвимостях – эти знания приходят с опытом. ASP.Net  в случае обнаружения некорректных данных сразу останавливает выполнение веб-приложения и предупреждает разработчика о потенциальной опасности. В PHP выполнение продолжается и никаких предупреждений программист не видит – такой уязвимостью могут воспользоваться злоумышленники.

## 3.1.6. Маленькие и большие проекты.

Традиционно считается, что PHP больше подходит для маленьких проектов, а ASP.Net – для больших. Я считаю, что это неправильно. Как известно, под PHP существует множество шаблонизаторов и фреймворков (Smarty, Zend Framework и другие), которые приближают его к ASP.Net с точки зрения архитектуры приложения (отделяется программный код от HTML-разметки, присутствует подобие ASP.Net-контролов и т.п). И примеров крупномасштабных проектов, написанных на PHP, достаточно: vkontakte.ru, photofile.ru и т.п. В то же время, и на ASP.Net можно легко и быстро (возможно, даже быстрее, чем на PHP) создать простой веб-сайт.  
К большому преимуществу ASP.Net я отнесу то, что он использует общую платформу .Net. Во-первых, это означает, что если вы – программист .Net, то вам не нужно учить новый язык и осваивать новые библиотеки. Во-вторых, этим открываются широкие возможности интеграции Windows-приложений и ASP.Net-приложений. Например, написанный ранее класс для настольного приложения можно без доработок и изменений использовать и в веб-проекте на ASP.Net. PHP же используется только для создания веб-страниц. С его помощью еще можно писать скрипты для командной строки, но такая потребность возникает довольно редко[8].

## 3.1.7. Выбор платформы.

Резюмируя все вышесказанное выбор был сделан в пользу ASP.NET. К основным плюсам ASP.NET. можно отнести:

* имеется встроенная защита от различных видов нападений(SQL Injection, XSS, переполнение буфера, изменение скрытых полей и прочие)
* строение ASP.NET как технологии позволяет компилировать программный код и все страницы сайта. Код интерпретируется в PHP значительно медленнее и не дает нужного эффекта производительности.
* платформа Microsoft.NET имеет множество встроенных технологий для интеграции информационных систем и приложений, таких как службы web, WCF, JSON, remoting, XML и пр., при этом ASP.NET существует как ее часть. Наличие таких многочисленных решений дает возможность выбора оптимальной технологии для каждого отдельного случая. Это обеспечит отменную производительность, масштабируемость и, самое главное, - безопасность.
* удобство разработки проекта благодаря Visual Studio

Кроме выше перечисленных плюсов можно отметить то, что Asp.net хорошо подойдет нам, т.к. библиотека для распознавания текстов альманахов с последующей выгрузкой их на сайт, написана на C# и легко интегрируема в среду ASP.NET.

## 3.2. Разработка визуальной части веб-приложения.

## 3.2.1. Разработка интерфейса подсистемы обработки и редактирования новостного контента.

Опишем в виде блок-схемы навигационную основу подсистемы обработки и редактирования новостного контента.



Рис. 10. Схема модуля управления новостным контентом.

Такая навигационная карта с одной стороны не перегружает администратора ненужными функциями, а с другой дает инструменты для эффективного управления новостным и документным потоком.

Теперь стоит пошагово разработать интерфейс для каждого подраздела модуля работы с новостным контентом.

Исходя из схемы модуля управления новостным контентом, разумно предположить, что домашняя страница будет иметь ссылки на страницы новостей, документов и пользователей.

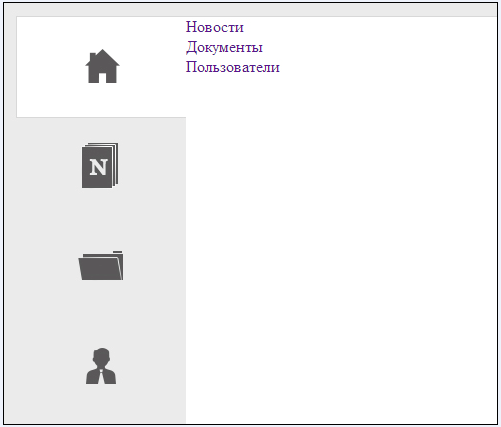


Рис. 11. Интерфейс домашней страницы.

Теперь стоит разработать интерфейс работы с новостным потоком. Исходя из нашей схемы, страница новостей будет содержать следующий функционал: добавление новости, редактирование новости и удаление выбранной новости или нескольких новостей.

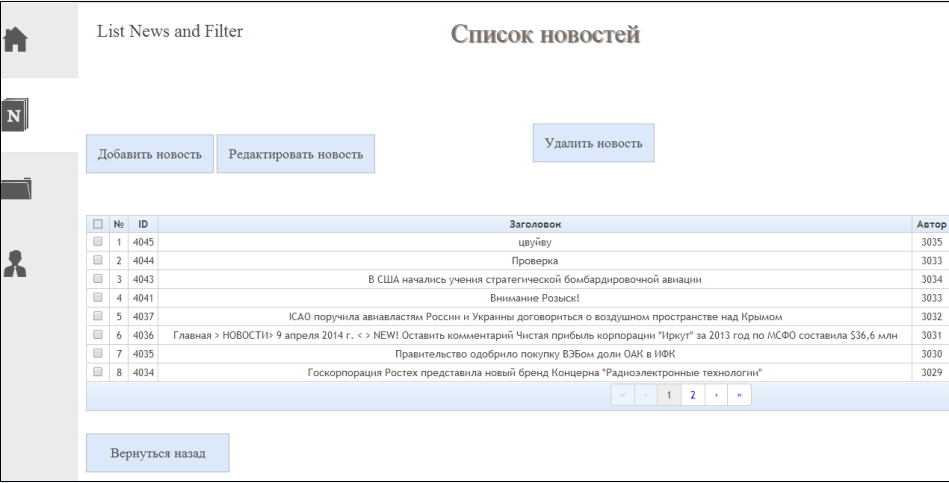


Рис. 12. Интерфейс страницы новостей.

При нажатии на кнопку «Добавить новость» должна появляться страница с соответствующими данной функции полями, т.е. «Заголовок новости» и «Текст новости».

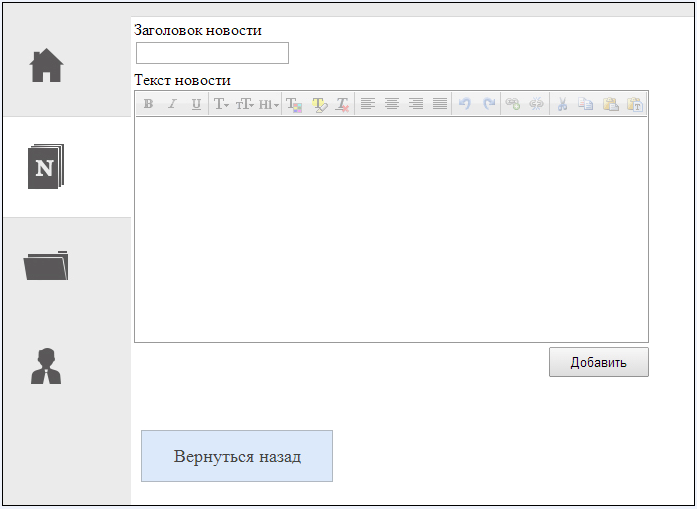


Рис.13. Страница добавления новости.

Рассмотрим функцию редактирования новости. Для начала стоит поставить «флажок» радом с выбранной новостью и нажать на кнопку «Редактировать новость». После этого должна появиться страница с уже заполненными существующими данными полями.

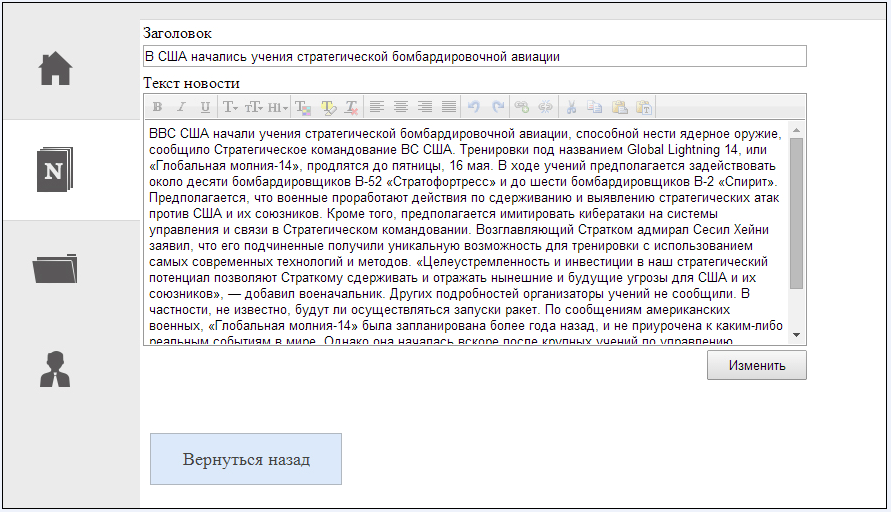


Рис.14. Страница редактирования новости.

Удаление выбранной новости осуществляется установкой «флажка» напротив новости и нажатия кнопки «Удалить новость».

Функционал подраздела работы с документами соответствует функциональной части подсистемы работы с новостным контентом и соответственно включает в себя функции добавления, редактирования и удаления документа.



Рис. 15. Интерфейс страницы документации.

При нажатии на кнопку «Добавить документ» будет появляться страница со следующими полями:

* формат документа
* тип документа
* название документа
* текст документа

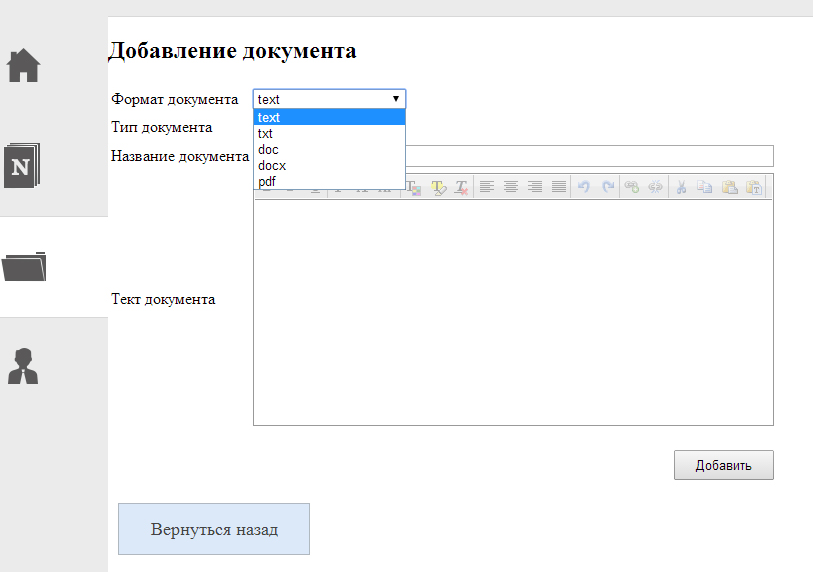


Рис. 16. Страница добавления документа.

Так же предусмотрено добавление уже существующего документа. Для этого необходимо просто выбрать его, используя стандартный поиск.

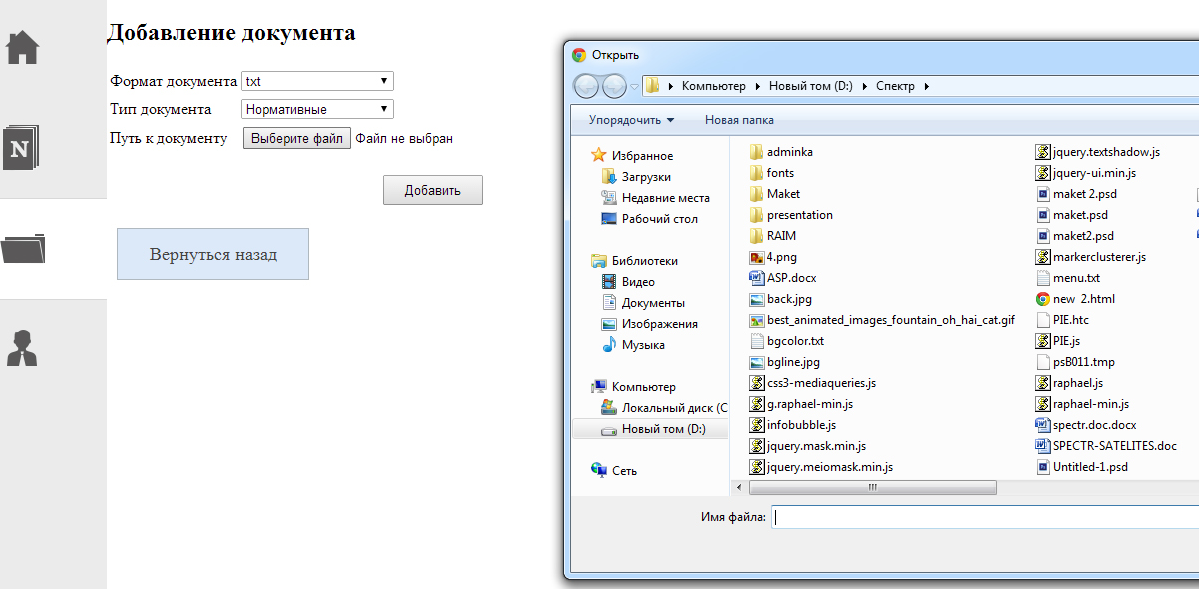


Рис. 17. Страница добавления документа.

Редактирование документа производится аналогично редактированию новости.

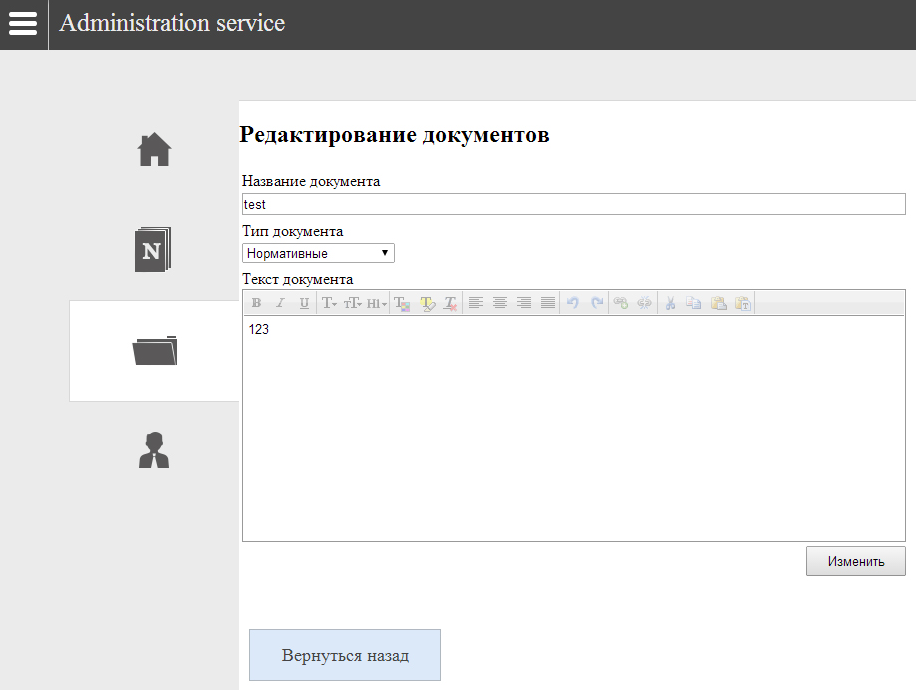


Рис. 18. Страница редактирования документа.

Домашняя страница функционального дополнения работы с данными пользователей будет иметь таблицу со списком всех пользователей, а так же кнопки, позволяющие вносить коррективы в их данные.

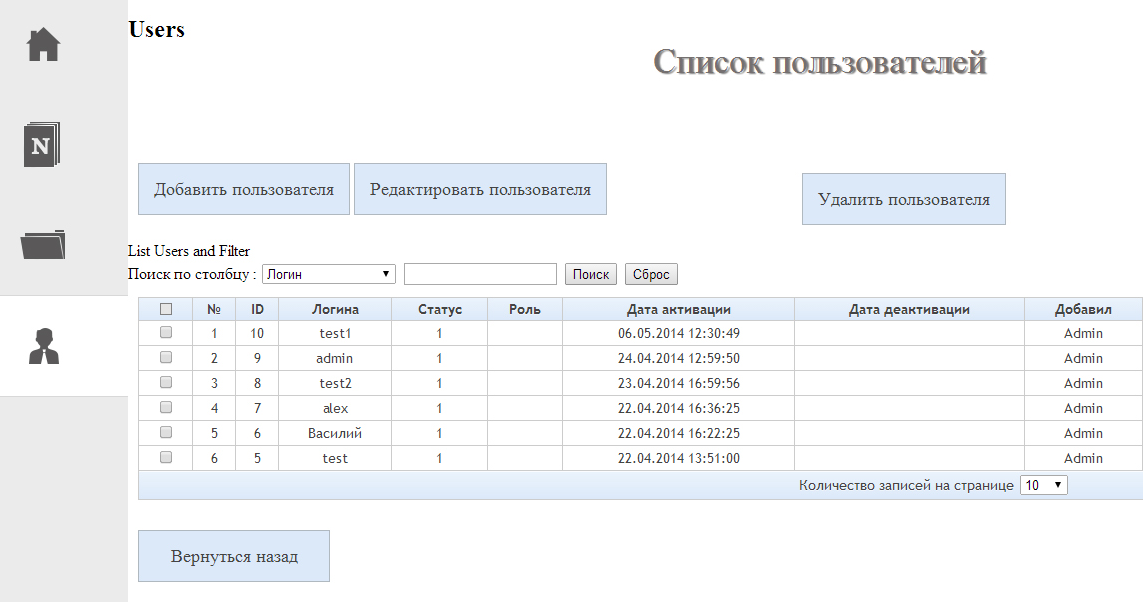
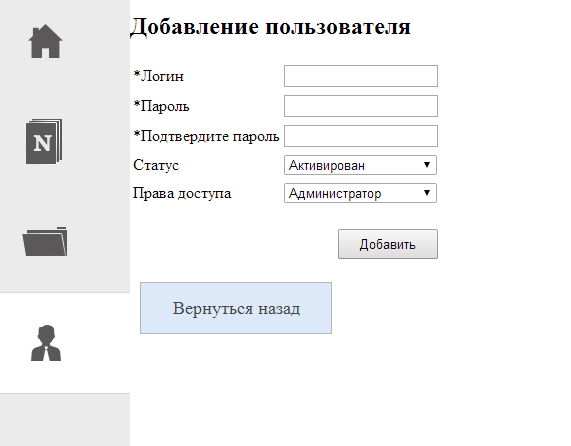


Рис.19. Страница списка пользователей.

Так же как и в подсистеме работы с новостями и документами предусмотрено добавление, редактирование и удаление пользователей.



Стр.20. Страница добавления пользователя.

## 3.2.2. Разработка пользовательского интерфейса функционального дополнения просмотра новостной ленты.

Как мы уже ранее определили для пользователя будет достаточно просмотра заголовка и самого текста новости. Список заголовков имеет смысл вынести в отдельную ленту.



Рис.21. Интерфейс новостной ленты.

## 3.2.3. Разработка интерфейса Real Time Service.

Интерфейс функции реального времени должен включать следующие компоненты:

* список аэропортов, для которых доступна данная функция
* табло состояния функции реального времени(такие данные как идентификатор ИКАО, число спутников, точность и т.д.)
* визуальное отображение данных получаемых в автономном и дифференциальном режиме
* визуальное отображение созвездий спутников

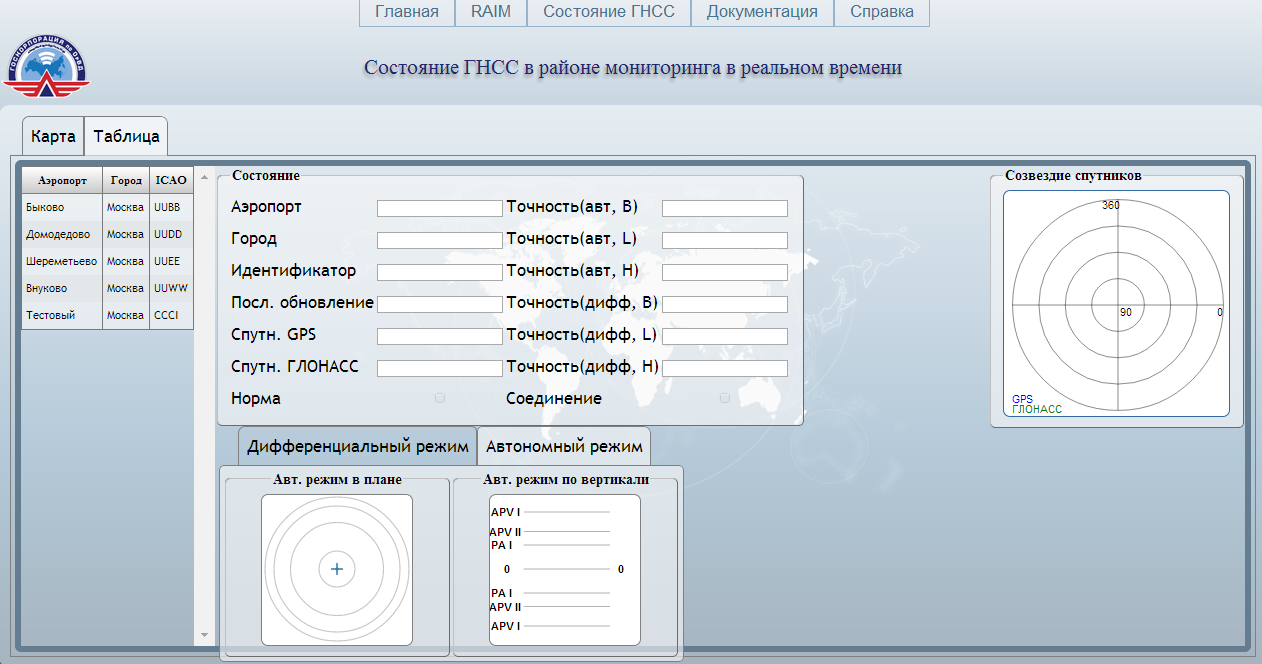


Рис. 22. Интерфейс функции реального времени.

## 3.3. Разработка программного обеспечения.

Контроллер пользовательской работы с новостями:

public class NewsController : Controller

{

//

// GET: /News/

int newscount = 1;

int nc = 5;

public ActionResult Index(int? i)

{

int pageNumber = 0;

if (i.HasValue)

{

pageNumber = i.Value - 1;

}

var NewsRepository = new GenericRepository<SpectrDbContext, News>();

var allNews = NewsRepository.All();

var orderedNews = from news in allNews

orderby news.DateCreate descending

select news;

var lastNews = from news in orderedNews.Skip(pageNumber \* nc).Take(nc)

select news;

foreach (var news in lastNews)

{

//int iLastWord = news.Body.LastIndexOf(" ", 200);

//news.Body = news.Body.Remove(iLastWord).TrimEnd(new char[] { ',', '.', '-' }) + "...";

}

var countOfNews = (from news in allNews

select news).Count();

int countOfPages = (countOfNews / nc) + 1;

ViewBag.News = lastNews;

ViewBag.CountPage = countOfPages;

return View();

}

[HttpGet]

public ActionResult ShowSelectedNews(int? id)

{

var NewsRepository = new GenericRepository<SpectrDbContext, News>();

var allNews = NewsRepository.All();

var selectedTitle = (from news in allNews

select news).Take(nc);

if (id != null)

{

int b = id.Value;

var selectedNews = (from news in allNews

where news.ID == b

select news).SingleOrDefault();

ViewBag.News = selectedNews;

ViewBag.Titles = selectedTitle;

return View();

}

else

{

ViewBag.Titles = selectedTitle;

return View();

}

}

public ActionResult ShowSelectedNews()

{

var NewsRepository = new GenericRepository<SpectrDbContext, News>();

var allNews = NewsRepository.All();

var selectedTitle = (from news in allNews

select news).Take(nc);

//var selectedNews = (from news in allNews

// orderby news.DateCreate

// select news).First();

//ViewBag.News = selectedNews;

ViewBag.Titles = selectedTitle;

return View();

}

public async Task<JsonResult> NextNews(int? n)

{

var NewsRepository = new GenericRepository<SpectrDbContext, News>();

int pageNumber = 0;

if (n.HasValue) {

pageNumber = n.Value;

}

var allNews = NewsRepository.All();

var selectedNews = (from news in allNews

orderby news.DateCreate

select news).Skip(pageNumber).Take(nc);

return Json(new { success = selectedNews != null, data = selectedNews }, JsonRequestBehavior.AllowGet);

}

}

Контроллер административной работы с новостями.

public class NewsController : Controller

{

private GenericRepository<SpectrDbContext, News> Repository { get; set; }

public NewsController()

{

Repository = new GenericRepository<SpectrDbContext, News>();

}

[Authorize]

public ActionResult Index(int? page, string filter, string val, string dstart, string dend)

{

var orderedNews = from news in Repository.All()

orderby news.DateCreate descending

select news;

ViewBag.NewsList = orderedNews;

NewsFilter nf = new NewsFilter();

ViewBag.Page = nf.Page(page);

ViewBag.Filter = nf.Filter(filter);

ViewBag.FilterVal = nf.FilterValue(val);

ViewBag.DateStart = nf.DateStart(dstart);

ViewBag.DateEnd = nf.DateEnd(dend);

ViewBag.NewsList = nf.FiltredList(filter, val, dstart, dend);

return View();

}

[Authorize]

public ActionResult Add()

{

return View();

}

/// <summary>

/// Добавление новости

/// </summary>

/// <param name="news"></param>

/// <returns></returns>

[Authorize]

[ValidateInput(false)]

[HttpPost]

public ActionResult Add(News news)

{

if (String.IsNullOrWhiteSpace(news.Body) || String.IsNullOrWhiteSpace(news.Title))

{

ViewBag.News = news;

ViewBag.Errors = "Заголовок и текст новости не должны быть пустыми!";

return View();

}

news.AuthorID = 3033;

news.UserCreate = "Admin";

news.UserUpdate = "Admin";

Repository.Add(news);

Repository.Save();

return RedirectToAction("Index");

}

[Authorize]

[HttpGet]

public ActionResult Edit(int id)

{

var selectedNews = Repository.FindBy(id);

ViewBag.News = selectedNews;

return View();

}

/// <summary>

/// Редактирование новости

/// </summary>

/// <param name="news"></param>

/// <returns></returns>

[Authorize]

[ValidateInput(false)]

[HttpPost]

public ActionResult Edit(News news)

{

News updateNews = Repository.FindBy(news.ID);

if (String.IsNullOrWhiteSpace(news.Body) || String.IsNullOrWhiteSpace(news.Title))

{

ViewBag.News = news;

ViewBag.Errors = "Заголовок и текст новости не должны быть пустыми!";

return View();

}

updateNews.Body = news.Body;

updateNews.Title = news.Title;

updateNews.UserUpdate = "Admin";

Repository.Update(updateNews);

Repository.Save();

return RedirectToAction("Index");

}

/// <summary>

/// Удаление новости

/// </summary>

/// <param name="news"></param>

/// <returns></returns>

[Authorize]

[HttpPost]

public ActionResult Del(News news)

{

News deleteNews = Repository.FindBy(news.ID);

Repository.Delete(news.ID);

Repository.Save();

ViewBag.News = news;

return RedirectToAction("Index");

}

}

Контроллер работы с функцией реального времени:

public class RealTimeController : Controller

{

AsyncGenericRepository<Airport> \_airportRepository = new AsyncGenericRepository<Airport>(new SpectrDbContext());

AsyncGenericRepository<City> \_airportCity = new AsyncGenericRepository<City>(new SpectrDbContext());

RedisRepository RTrepository = new RedisRepository("192.168.3.61");

protected override void Dispose(bool disposing)

{

base.Dispose(disposing);

}

public ActionResult Index()

{

return View();

}

public async Task<ActionResult> getAjaxTab(int id)

{

if (id == 2)

{

var data = \_airportRepository.SearchFor(a => a.CityID == 5360).Select(a => new CitiestableView()

{

CityName = a.City.NameRus,

ICAOCode = a.ICAOCode,

AirName = a.NameRus

}).ToList();

return PartialView("\_Table", data);

}

return PartialView("\_Map");

}

public async Task<JsonResult> GetRealTimeData(string StationID)

{

RealTimeViewModel rtData = null;

Packet data = Packet.Deserialize(await RTrepository.AsyncGet(StationID));

var airport = \_airportRepository.SearchFor(x => x.ICAOCode == StationID).First();

var sity = \_airportCity.SearchFor(x => x.ID == airport.CityID).First();

// /\*

if (data != null && data.Signature.GbasId == StationID)

{

rtData = new RealTimeViewModel()

{

Station = new Station()

{

Airport = airport.NameRus == null ? airport.NameEng : airport.NameRus,

CityName = sity.NameRus == null ? sity.NameEng : sity.NameRus,

IDStation = data.Signature.GbasId,

LastUpdateDateTime = data.Signature.DateTime.ToString("dd.MM.yyyy HH:mm:ss"),

GPSCount = data.Sats.Select(x => x.sat\_number < 37).Count(),

GLONASSCount = data.Sats.Select(x => x.sat\_number > 37).Count(),

Hdop = data.DOP.hdop,

Tdop = data.DOP.tdop,

Vdop = data.DOP.vdop,

NetRcv = data.Failover.net\_rcv == 0 ? 1 : 0,

NetSend = data.Failover.net\_send == 0 ? 1 : 0,

LatStd = Math.Round(data.Standalone.elat, 2, MidpointRounding.AwayFromZero),

LonStd = Math.Round(data.Standalone.elon, 2, MidpointRounding.AwayFromZero),

AltStd = Math.Round(data.Standalone.ealt, 2, MidpointRounding.AwayFromZero),

LatDiff = Math.Round(data.Diff.elat, 2, MidpointRounding.AwayFromZero),

LonDiff = Math.Round(data.Diff.elon, 2, MidpointRounding.AwayFromZero),

AltDiff = Math.Round(data.Diff.ealt, 2, MidpointRounding.AwayFromZero)

},

Sats = data.Sats.Select(x => new SPECTR.WEB.SITE.Models.ViewModel.Sat()

{

SatID = x.sat\_number,

SatNumber = x.sat\_number > 37 ? x.sat\_number - 37 : x.sat\_number,

X = (90 - x.angle) \* Math.Cos((x.azimuth + 270) \* -1 \* Math.PI / 180),

Y = (90 - x.angle) \* Math.Sin((x.azimuth + 270) \* -1 \* Math.PI / 180)

}).ToList()

};

return Json(new { data = rtData, success = true }, JsonRequestBehavior.AllowGet);

}

//\*/

Airport n = new Airport();

return Json(new { data = rtData, success = false }, JsonRequestBehavior.AllowGet);

}

public async Task<JsonResult> GetAirports()

{

var data = \_airportRepository.SearchFor(a => a.CountryID == 625).Select(a => new

{

CityName = a.City.NameRus,

ICAOCode = a.ICAOCode,

Latitude = a.Latitude,

Longitude = a.Longitude,

AirName = a.NameRus,

Status = a.ID < 5700 ? "red" : a.ID > 5740 ? "green" : "gray"

});

return Json(data, JsonRequestBehavior.AllowGet);

}

public async Task<JsonResult> GetCords()

{

var data = \_airportRepository.SearchFor(a => a.CountryID == 625).ToList().Select(a => new []

{

a.Longitude,

a.Latitude

});

//var x = data;

return Json(data, JsonRequestBehavior.AllowGet);

}

public async Task<JsonResult> GetStatusForStation(string StationID)

{

Packet data = Packet.Deserialize(await RTrepository.AsyncGet(StationID));

if (data == null)

{

return Json(new { success = false, data = 0 }, JsonRequestBehavior.AllowGet);

}

var airport = \_airportRepository.SearchFor(x => x.ICAOCode == StationID).First();

var sity = \_airportCity.SearchFor(x => x.ID == airport.CityID).First();

var rezult = new {

Airport = airport.NameRus == null ? airport.NameEng : airport.NameRus,

CityName = sity.NameRus == null ? sity.NameEng : sity.NameRus,

IDStation = data.Signature.GbasId,

LastUpdateDateTime = data.Signature.DateTime.ToString("dd.MM.yyyy HH:mm:ss"),

GPS = data.Sats.Where(x => x.sat\_number < 37).Select(s => s.sat\_number).ToArray(),

GLONASS = data.Sats.Where(x => x.sat\_number > 37).Select(s => s.sat\_number).ToArray(),

Hdop = data.DOP.hdop,

Tdop = data.DOP.tdop,

Vdop = data.DOP.vdop,

NetRcv = data.Failover.net\_rcv == 0 ? "да" : "нет",

NetSend = data.Failover.net\_send == 0 ? "да" : "нет",

LatStd = Math.Round(data.Standalone.elat, 2, MidpointRounding.AwayFromZero),

LonStd = Math.Round(data.Standalone.elon, 2, MidpointRounding.AwayFromZero),

AltStd = Math.Round(data.Standalone.ealt, 2, MidpointRounding.AwayFromZero),

LatDiff = Math.Round(data.Diff.elat, 2, MidpointRounding.AwayFromZero),

LonDiff = Math.Round(data.Diff.elon, 2, MidpointRounding.AwayFromZero),

AltDiff = Math.Round(data.Diff.ealt, 2, MidpointRounding.AwayFromZero)

};

return Json(new { success = rezult != null, data = rezult }, JsonRequestBehavior.AllowGet);

}

}

## 3.3.1. Тестирование программного обеспечения.

## 3.3.1.1. Тестирование новостного модуля.

Для начала проверим возможность добавления новости. Для того чтобы не заполнять базу данных посторонними новостями, будем черпать последние с сайта посвященного авиации www.aex.ru.

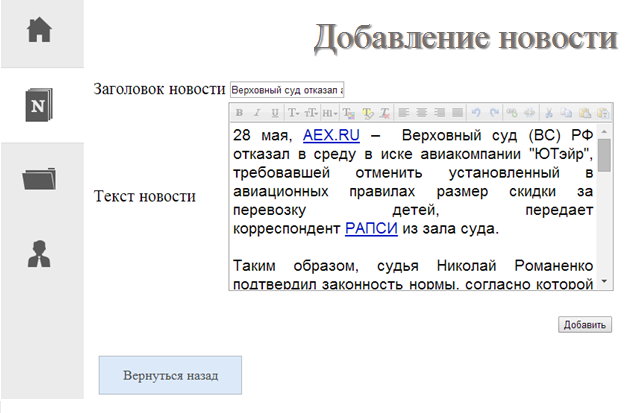


Рис.23. Добавление новости.

Как мы видим новость успешно добавлена.

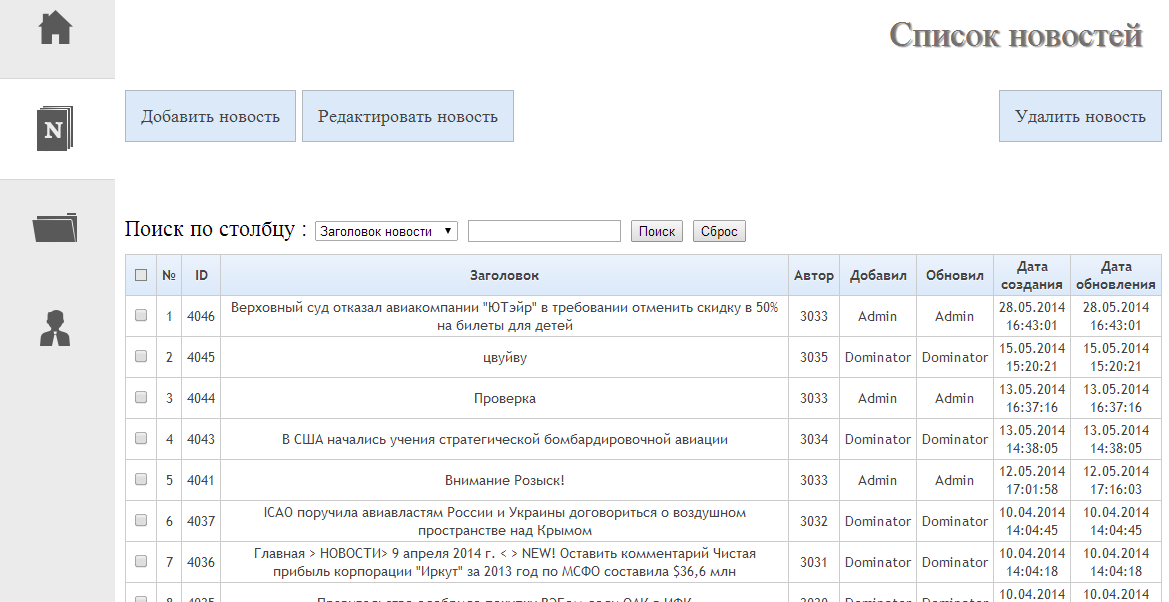


Рис. 24. Страница с добавленной новостью.

Теперь проверим возможность редактирования новости. Для этого установим "флажок" напротив выбранной новости и нажмем кнопку "Редактировать новость".

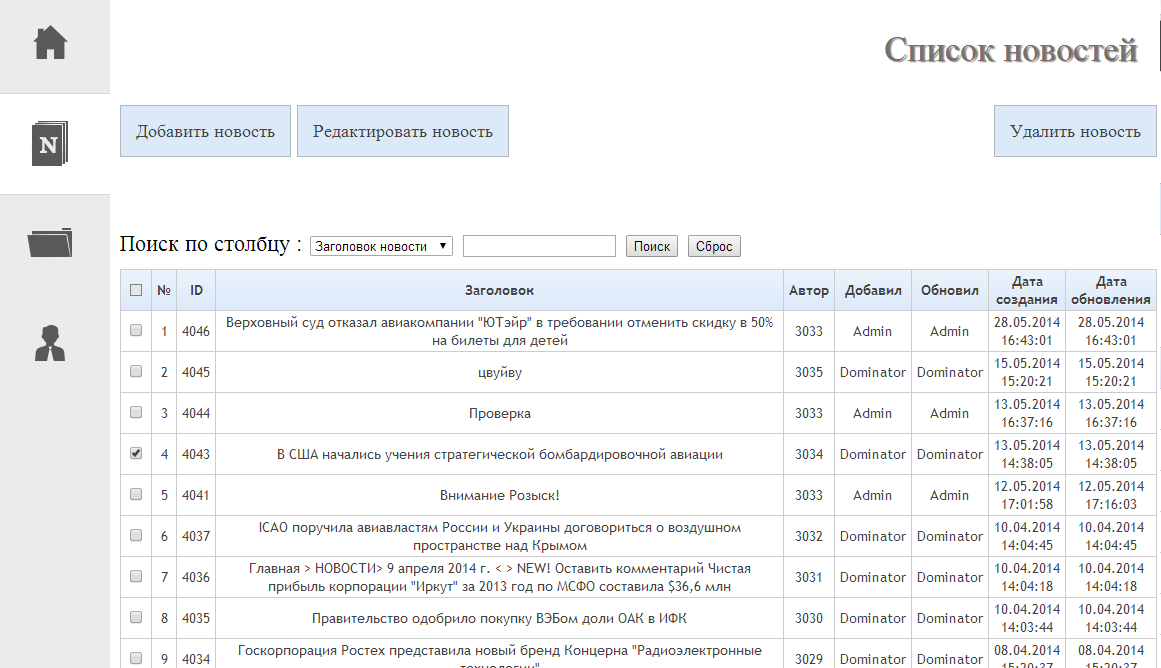


Рис.25. Выбор редактируемой новости.

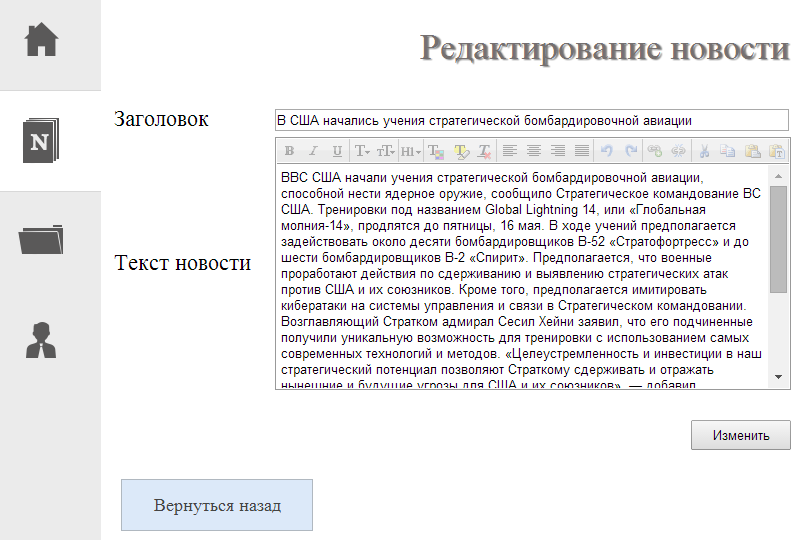


Рис.26. Новость до редактирования.

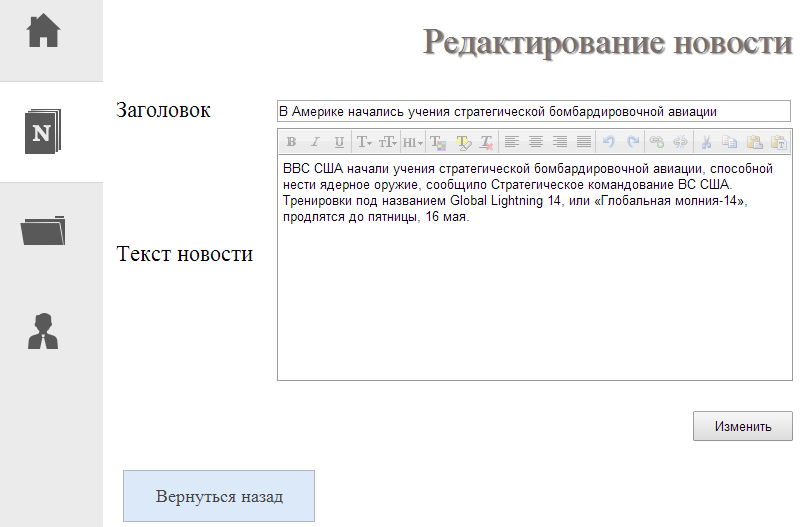


Рис.27. Новость после редактирования.

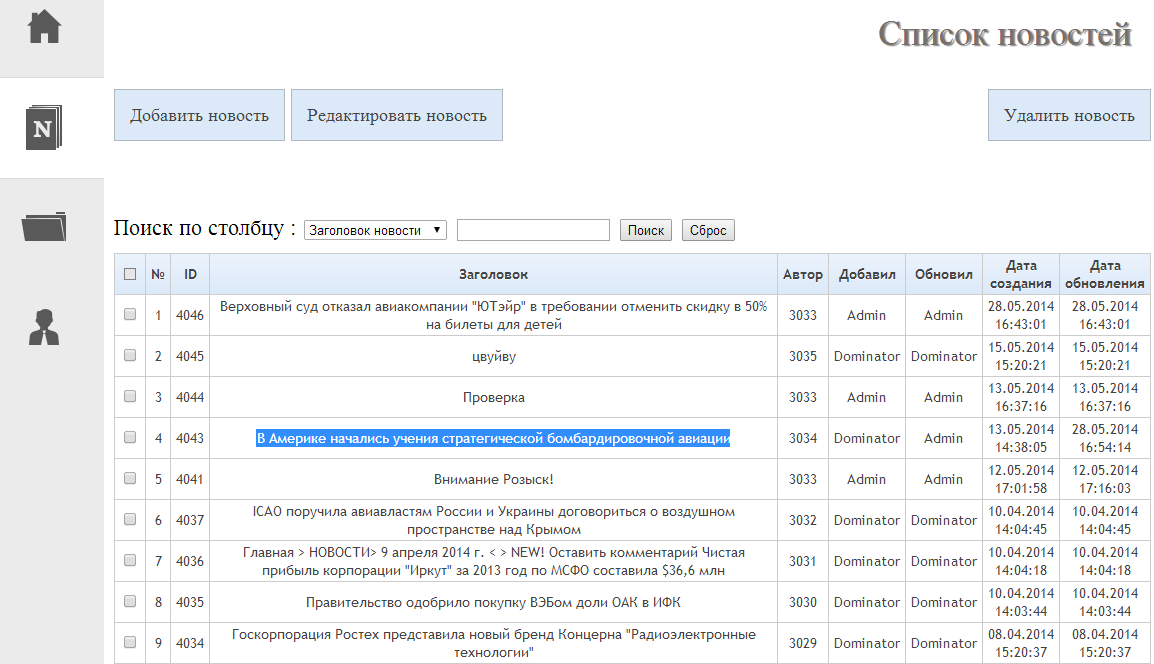


Рис.28. Отредактированная новость отображается в новостной ленте.

Проверим возможность удаления новости.

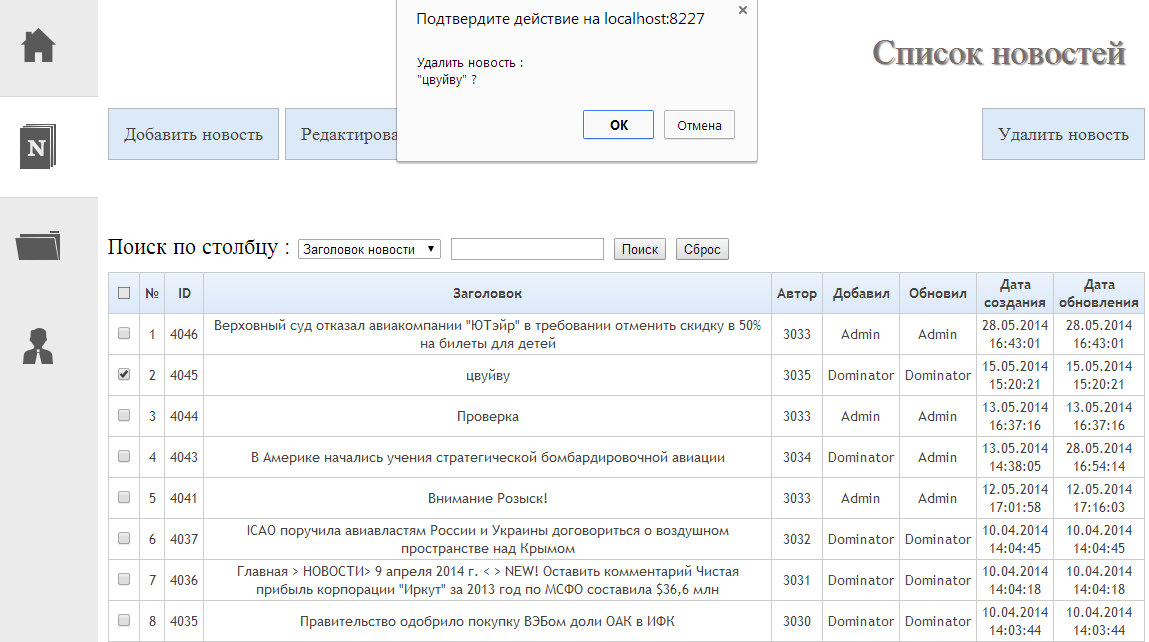


Рис.29. Выбор новости.

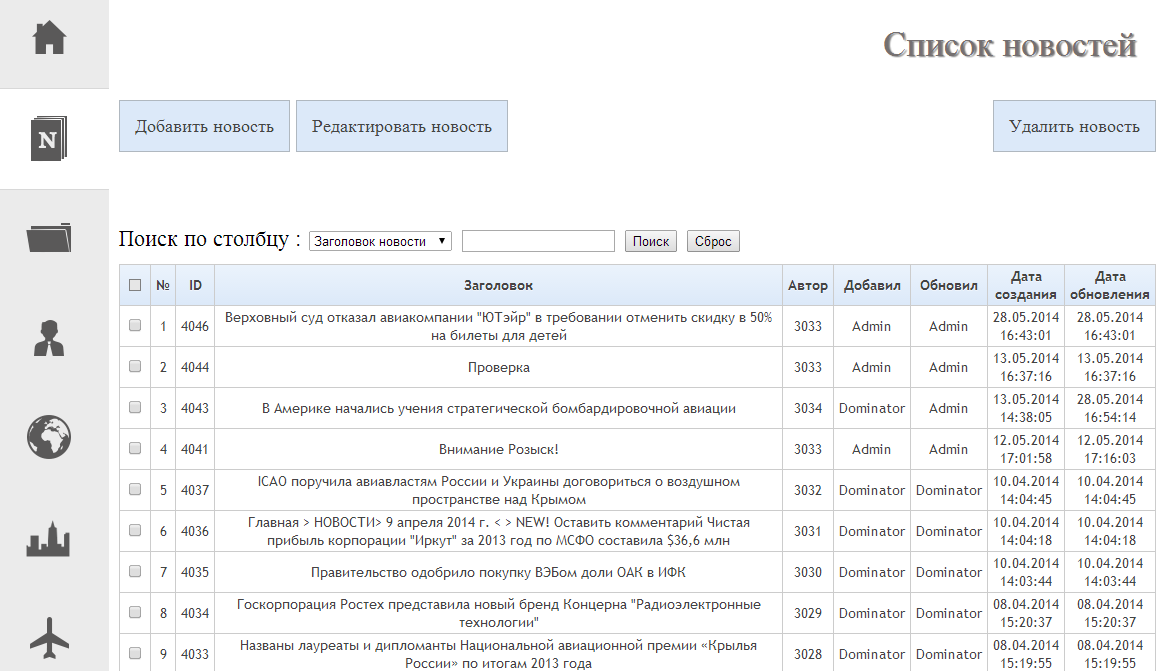


Рис. 30. Обновленный список новостей.

Так же необходимо рассмотреть сценарий, когда пользователь жмет на кнопку удаления или редактирования, но при этом не установил "флажок" напротив выбранной новости.



Рис. 31. Страница новостей.

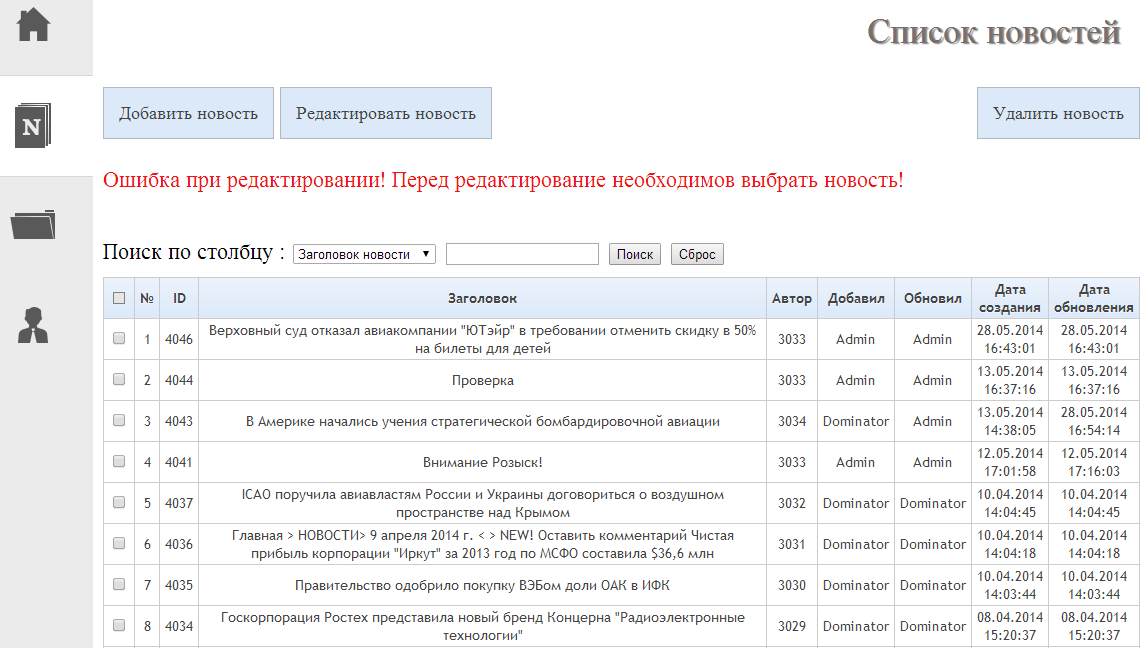


Рис. 32. Страница новостей.

Рассмотрим сценарий добавления и редактирования новости, в случае, когда не все поля заполнены данными.

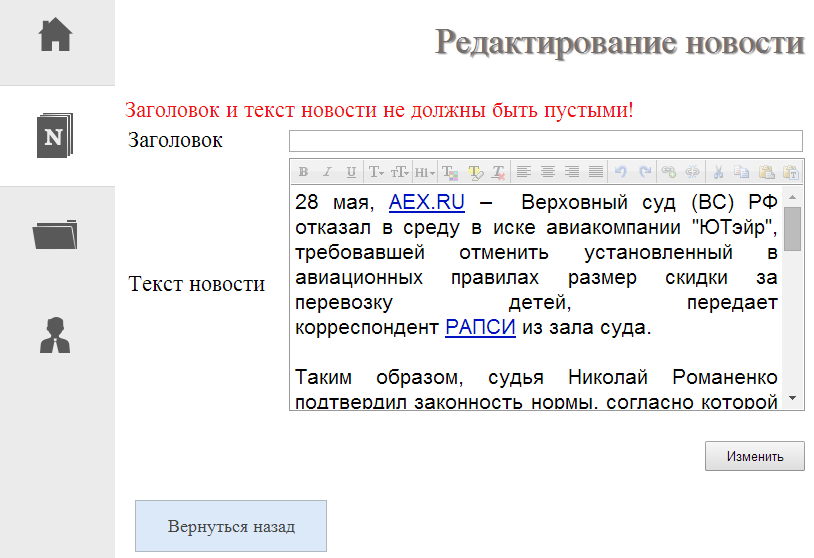


Рис.33. Страница редактирования новости.

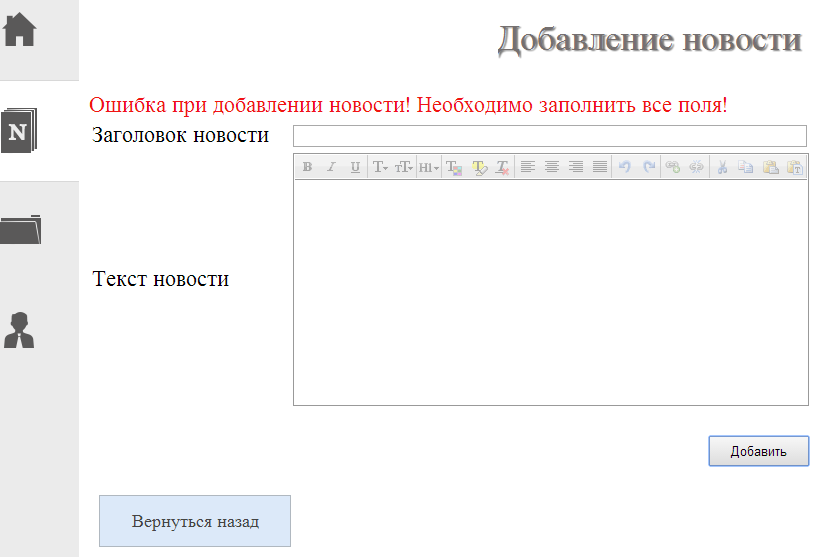


Рис. 34. Страница добавления новости.

## 3.3.1.2. Тестирование Real Time Service.

Для тестирования сервиса реального времени нужно выбрать вкладку "Таблица" и в списке аэропортов выбрать "Тестовый аэропорт". Таким образом, мы можем видеть, как обрабатываются данные, полученные при тестировании Real Time Service.

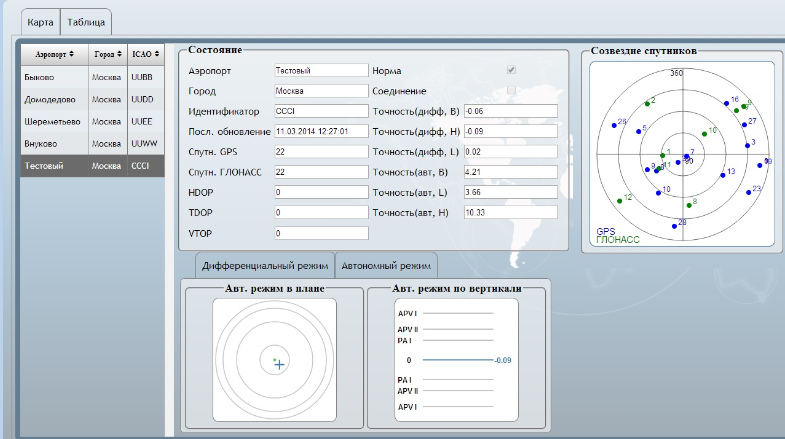


Рис. 35. Работа сервиса реального времени в тестовом режиме.

# Заключение.

В результате выполнения выпускной квалификационной работы, были получены следующие результаты:

* Были описаны базовые процессы работы с навигационной информацией
* Произведен анализ и выявлены основные недостатки существующей системы управления воздушным движением
* Разработана информационно-логическая модель подсистемы сбора и обработки информации сервиса реального времени
* Разработана информационно-логическая модель подсистемы работы с новостным контентом системы
* Произведен анализ и выбрана архитектура реализации проекта
* Проанализирована и выбрана платформа для веб-разработки
* Выбран язык программирования
* Разработана визуальная часть подсистем работы с новостным контентом и функции реального времени
* Разработано и протестировано программное обеспечение для подсистем работы с новостным контентом и функции реального времени

# Библиографический список.

1. Александр Коломенский: "И все таки оно вертится".

<http://old.computerra.ru/terralab/mobilis/366413/?print=Y>

2. Александр Петров, Зорин Виктор: Глобальные Навигационные Спутниковые Системы(GNSS). Как это работает?

<http://gps-club.ru/gps_think/detail.php?ID=20187>

3. Спутниковые навигационные системы.

<http://novdocs.ru/docs/index-26945.html>

4. Логические слои программного обеспечения распределенных информационных систем.

<http://knowledge.allbest.ru/programming/3c0b65635b3ac78a4c43a89521216d37_0.html>

5. Смирнов Александр: Курсовой проект на тему "Клиент-серверная архитектура и ее развитие".

<http://niios.ru/docs/Kurilenko/clientserverarch.pdf>

6. Типы клиентов в системе клиент-сервер

<http://bourabai.kz/dbt/richclient.htm>

7. Языки программирования Web-сайтов

<http://hinex.ru/info/yazyki-programmirovania-web-saitov.php>

8. Неволин Александр: PHP против ASP.Net - что лучше?

<http://nevlabs.ru/articles/web/php-vs-aspnet/comment_error/>/