СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc420900103)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ОБЗОР ТЕХНИЧЕСКИХ СРЕДСТВ И СУЩЕСТВУЮЩИХ ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ ПРОВЕДЕНИЯ МЕРОПРИЯТИЙ. 7](#_Toc420900104)

[1.1. Описание предметной области 7](#_Toc420900105)

[1.2. Функциональные особенности объекта управления. 7](#_Toc420900106)

[1.3. Исследования автоматизации процессов 8](#_Toc420900107)

[1.4. Обзор технических средств 9](#_Toc420900108)

[1.4.1. Мобильные устройства 9](#_Toc420900109)

[1.4.2. Мобильные операционные систем 10](#_Toc420900110)

[1.5. Описание существующих информационных систем 12](#_Toc420900111)

[1.5.1. Программно-аппаратный комплекс поддержки проведения параолимпийских игр 12](#_Toc420900112)

[1.5.2. Сравнительный анализ 15](#_Toc420900113)

[2. ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ ПРОВЕДЕНИЯ «ФЕСТИВАЛЯ НАУКИ» 17](#_Toc420900114)

[2.1. Функциональные возможности программного обеспечения поддержки проведения «Фестиваля науки» 17](#_Toc420900115)

[2.2. Входные и выходные данные программном обеспечении поддержки проведения «Фестиваля науки» 19](#_Toc420900116)

[2.3. Информационные потоки 21](#_Toc420900117)

[2.4. Структура классов 24](#_Toc420900118)

[2.5. Варианты использования системы. 27](#_Toc420900119)

[3 ОПИСАНИЕ РЕАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ ПРОВЕДЕНИЯ «ФЕСТИВАЛЯ НАУКИ» 30](#_Toc420900120)

[3.1. Выбор программного обеспечения. 30](#_Toc420900121)

[3.1.1. Используемые платформы. 30](#_Toc420900122)

[3.1.1.1. .NET Framework 30](#_Toc420900123)

[3.1.1.2. Java EE 32](#_Toc420900124)

[3.1.2. Используемая технологии 33](#_Toc420900125)

[3.1.3 Используемые языки программирования 39](#_Toc420900126)

[3.1.4 Средства разработки приложения клиента 40](#_Toc420900127)

[3.1.4.1 Unity3D 40](#_Toc420900128)

[3.1.4.2. Microsoft Visual Studio Professional 2012 41](#_Toc420900129)

[3.1.5 Средства разработки приложения сервера 42](#_Toc420900130)

[3.1.6. Используемая СУБД 43](#_Toc420900131)

[3.1.6.1. Проектирование базы данных системы 44](#_Toc420900132)

[3.1.7. Метод взаимодействия в сети 49](#_Toc420900133)

[3.1.8. Тип отправляемых пакетов 51](#_Toc420900134)

[3.2. Описание интерфейса и примеры работы приложения. 53](#_Toc420900135)

[3.2.1. Описание интерфейса редактора 53](#_Toc420900136)

[3.2.1. Описание интерфейса основной программы 60](#_Toc420900137)

[ЗАКЛЮЧЕНИЕ 61](#_Toc420900138)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63](#_Toc420900139)

# ВВЕДЕНИЕ

Задача управления и учета перемещения организованных групп людей по ограниченной территории относится к одной из наиболее востребованных в современном обществе. Такая задача имеет множество прикладных областей, начиная от ведения боевых действий и заканчивая управлением экскурсионными группами. Эти задачи трудны с организационной точки зрения и удовлетворительные их результаты сегодня могут быть достигнуты только при длительной специальной подготовке всех участников таких мероприятий, что не всегда возможно и не всегда дает нужный результат. Чаще всего, они порождают проблемы технического рода, препятствующие эффективному проведению мероприятия.

Проведение «Фестиваля науки», как и различных мероприятий выявляет множество технических проблем, препятствующие их эффективному проведению.

Цель выпускной квалификационной работы заключается в разработке специализированной программного обеспечения для «Фестиваля науки» СГТУ имени Ю.А. Гагарина, которая позволит автоматизировать процесс проведения мероприятия, облегчить сбор данных и их обработке.

Для достижения поставленной цели необходимо реализовать программное обеспечения и снабдить следующим функционалом:

* разграничение прав доступа;
* ввод данных о сотрудниках, аудиториях, мероприятиях, группах людей;
* редактирование данных о сотрудниках, аудиториях, мероприятиях, группах людей;
* удаление данных о сотрудниках, аудиториях, мероприятиях, группах людей;
* генерация отчетов;
* предоставление интерактивной карты местности;
* чат;
* новостная рассылка.

Информатизация мероприятия позволит снизить трудозатраты на его проведение, а также автоматизировать большинство процессов.

Объектом автоматизации является «Фестиваль науки» федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Саратовский государственный технический университет имени Гагарина Ю.А.»

# ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ОБЗОР ТЕХНИЧЕСКИХ СРЕДСТВ И СУЩЕСТВУЮЩИХ ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ ПРОВЕДЕНИЯ МЕРОПРИЯТИЙ.

## 1.1. Описание предметной области

Работа посвящена разработке специализированного программного обеспечения, содержащим информацию о мероприятии «Фестиваль науки»

«Фестиваль науки» это открытое мероприятие, направленное на привлечение внимания потенциальных абитуриентов к научной карьере, к обучению и выбору потенциальной профессии. Организационно данное мероприятие выглядит следующим образом: для ВУЗа назначается день и выбирается время (скорее всего первая половина дня) в которую проводится мероприятие, ВУЗ организует демонстрационные площадки, на ознакомление с которыми приезжают школьники.

Цели проведения «Фестиваля науки»:

* продвижение имиджа высшего образования на российском и региональном уровне;
* укрепление связей со школами и предприятиями Саратова и области;
* популяризация науки и инженерного образования среди учащейся молодежи, привлечение абитуриентов в вуз;
* активизация научно-технического творчества молодежи;
* демонстрация новейших научно-технических достижений, возрастающей роли науки и образования в экономике и культуре.

## 1.2. Функциональные особенности объекта управления.

Одной из главных задач проведения «Фестиваля науки» является привлечение потенциальных абитуриентов к научной карьере в ВУЗе. Это достигается путём демонстрации новейших научных достижений на стендах, проведением мастер-классов, разнообразными акциями и событиями.

Данная задача трудоёмка, ввиду незнания группами местности и отсутствием быстро доступной информации о мероприятии. Поэтому обязанности курировать положение групп ложится на персонал:

* встречают новоприбывшие группы;
* записывают данные о них;
* проводят ознакомление с материалами мероприятия;
* ищут свободные аудитории для проведения мастер классов

Ведение статистики и отчётности мероприятия ложится на ответственных за его проведение, путём непосредственного общения с персоналом.

## 1.3. Исследования автоматизации процессов

Определим основные проблемы предметной области и выделим ту часть, которую можно частично или полностью автоматизировать.

Выше были описаны процессы, выполняемые при проведении мероприятия. Выявим ключевые проблемы, возникающие при проведения «Фестиваля науки»:

* сбор и анализ информации о экскурсироваемых группах производится ручным способом на месте;
* в ключевых точках должны постоянно дежурить сотрудники ответственные за те или иные задачи, при этом у них нет никакой информации о возможной необходимой помощи в других местах;
* отсутствие полноценного понимания текущей ситуации о кого бы от ни было из участников, что зачастую приводит к демотивации, повышенным стрессовым нагрузкам;
* общая несогласованность действий разных подразделений;
* сложность учета занятости или доступности различных участков и разделения групп по зонам специализации;
* сложность ведения статистики мероприятия;
* Посетители, не входящие в организованные группы не смогут в полной мере использовать имеющиеся возможности.

Для решения рассмотренных проблем необходимо реализовать следующие процессы:

* предлагается разработать программное обеспечение для мобильных устройств, предоставляющее доступ к информации о мероприятии и согласующее персонал;
* внесение информации о пришедших группах;
* информационная рассылка;
* общение персонала;
* получение информации в реальном времени;
* программное обеспечение позволит автоматически генерировать необходимые отчеты.

## 1.4. Обзор технических средств

Рассмотрим мобильные технические средства.

### 1.4.1. Мобильные устройства

На данный момент активно развиваются такие устройства, как смартфоны, планшетные компьютеры. Они занимают огромную роль в жизни человека. Моментальная связь с любым другим устройством, легкодоступный выход в интернет, а следственно доступ в социальные сети, и связь с облачными хранилищами. Полезные программное обеспечение, функции, опции которыми оснащены современные устройства, делают его многофункциональным. Рассмотрим некоторые определения, позволяющие понять разницу между мобильными устройствами.

Смартфон – это мобильный телефон, дополненный функциональностью карманного персонального компьютера. Одно из главных отличий от мобильных телефонов наличием достаточно развитой операционной системы [5], открытой для разработки программного обеспечения сторонними разработчиками (операционная система обычных мобильных телефонов закрыта для сторонних разработчиков). Смартфоны имеют возможность полноценной обработки данных, которые ранее были присущи только персональным компьютерам. Установка дополнительных приложений позволяет значительно улучшить функциональность смартфонов по сравнению с обычными мобильными телефонами. Пользователи смартфонов могут устанавливать и запускать приложения, управлять данными в сети Интернет.

Планшетный компьютер – это относительно новый вид мобильных компьютеров. Главное отличие планшетного компьютера от персонального – это сенсорный экран, который является единственным устройством ввода данного устройства. Так как все планшетные компьютеры не оснащаются мышкой и аппаратной клавиатурой, то управление программами, набор текста, управление приложениями выполняется только на сенсорной панели компьютера.

На сегодняшний день для разработчиков приложений для мобильных платформ наиболее важной задачей является создание таких приложений, которые могли бы одинаково корректно функционировать и отображаться на различных мобильных устройствах. При этом с появлением различных платформ у разработчика появилась возможность при создании приложений выбирать наиболее подходящую и привлекательную с точки зрения реализации операционную систему.

### 1.4.2. Мобильные операционные систем

Наиболее распространенной платформой на данный момент является операционная система (ОС) Android, разработанная компанией Google. Android - это операционная система [4] для смартфонов, планшетных компьютеров и других устройств. Данная ОС основана на ядре Linux и собственной реализации Java от Google. Для данной ОС возможно разрабатывать Java-приложения, которые управляют устройством через разработанные Google библиотеки. Набор инструментов Android Native Development Kit позволяет портировать библиотеки и компоненты приложений, написанные на С++ и других языках [6].

Не менее распространена выпущенная в 2007 году система iOS от компании Apple. IOS - это мобильная операционная система, которая изначально создавалась для iPhone и iPod touch, но через некоторое время была портирована на такие устройства, как Apple TV и iPad. Данная операционная система выпускается только для устройств, производимых фирмой Apple, в отличие от Windows Phone и Google Android. iOS является первой из созданных операционных систем, которые основаны на концепции управления интерфейсом с использованием жестов мультитач. Она совместима со стандартом POSIX и разработана на основе OS X[6]. Последней версией операционной системы на данный момент является iOS 7.1.1 с улучшением стабильности и исправлением некоторых ошибок.

Созданная компанией Microsoft новая операционная система Windows Phone 8 занимает третье место среди мобильных платформ. Windows Phone 8 - это мобильная операционная система, анонсированная 20 июня 2012 года, главное преимущество которой - возможность объединения компьютеров, смартфонов и планшетов в одну "экосистему». Разработка программного обеспечения между разными платформами становится значительно проще и тем самым, данная ОС, является удобным средством для разработчиков[8]. Так же к достоинствам данной ОС относится прежде всего стабильная работа смартфонов, к недостаткам малое по сравнению с другими системами, число пользовательских приложений. Эти причины и послужили основанием для разработки клиентской части приложения для устройств под управлением операционной системы Windows Phone.

## 1.5. Описание существующих информационных систем

Рассмотрим существующие мобильные приложения по поддержки проведения мероприятий.

### 1.5.1. Программно-аппаратный комплекс поддержки проведения параолимпийских игр

Программно-аппаратный комплекс поддержки проведения параолимпийских игр [1], использовался для автоматизации задач по направлениям:

1. Предоставление оперативной сводки мероприятия;
2. Информационная рассылка;
3. Карта проведения события;
4. Общение персонала;
5. Поиск сотрудников.

Задачи были такие: обеспечивать организацию коммуникаций между сотрудниками Паралимпийского комитета, отвечающими за взаимодействие со СМИ, аналитику, координацию Игр и пр. Так, выделенные группы тренеров сборных команд России, сервисменов (отвечающих за подготовку спортинвентаря), аналитиков, координаторов и медицинских работников должны были пользоваться планшетами со специальным софтом на борту, находясь непосредственно в местах проведения соревнований. Сотрудники Штаба использовали десктопы или ноутбуки и браузерный клиент. Координация включала постоянный контроль информационного потока со всех мест Игр, постановку задач, поиск людей на карте и организацию связи с ними.

На рисунке 1.1 показан пример работы программного обеспечения поддержки параолимпийских игр.

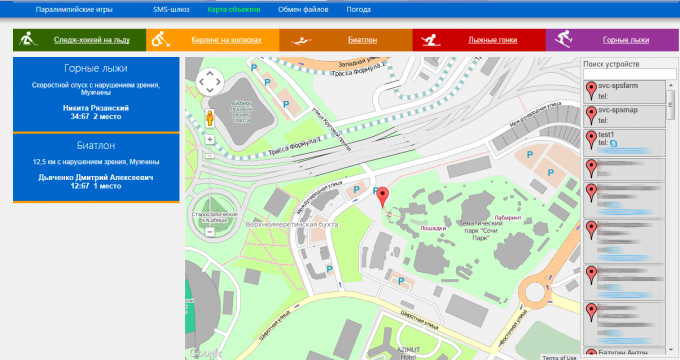


Рисунок 1.1. Пример работы ПО поддержки параолимпийских игр

На рисунке 1.2 показан пример структуры программного обеспечения поддержки параолимпийских игр.

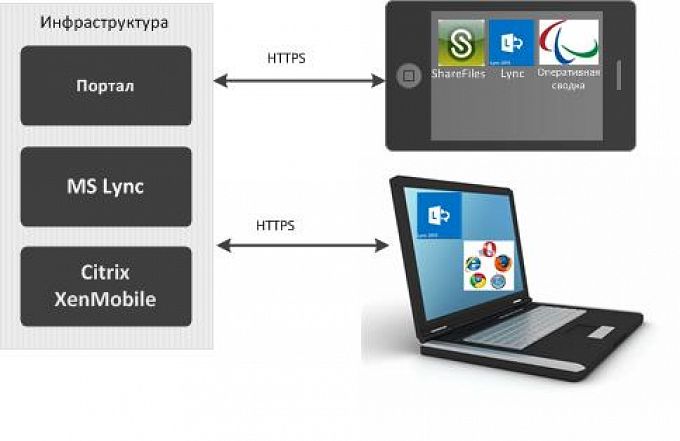


Рисунок 1.2. Пример структуры ПО поддержки параолимпийских игр

Программное обеспечение, используемое в программно-аппаратном комплексе:

* MS Lync — организация видеосвязи между оперативными работниками и работниками штаба.
* просмотр информации о погодных условиях.
* оперативная сводка результатов Игр, составляемая на лету: публикация и просмотр информации о результатах выступления сборной России, публикация и просмотр информации о результатах выступления соперников, просмотр информации по общему медальному зачёту.
* SMS-рассылка для различных групп пользователей, просмотр истории, управление группами рассылки.
* система обмена файлами (в основном — фото и видео из отчётов о выполнении задач).
* просмотр информации о местонахождении оперативных работников.
* управление (администрирование) и учёт планшетных компьютеров в рамках мероприятия.

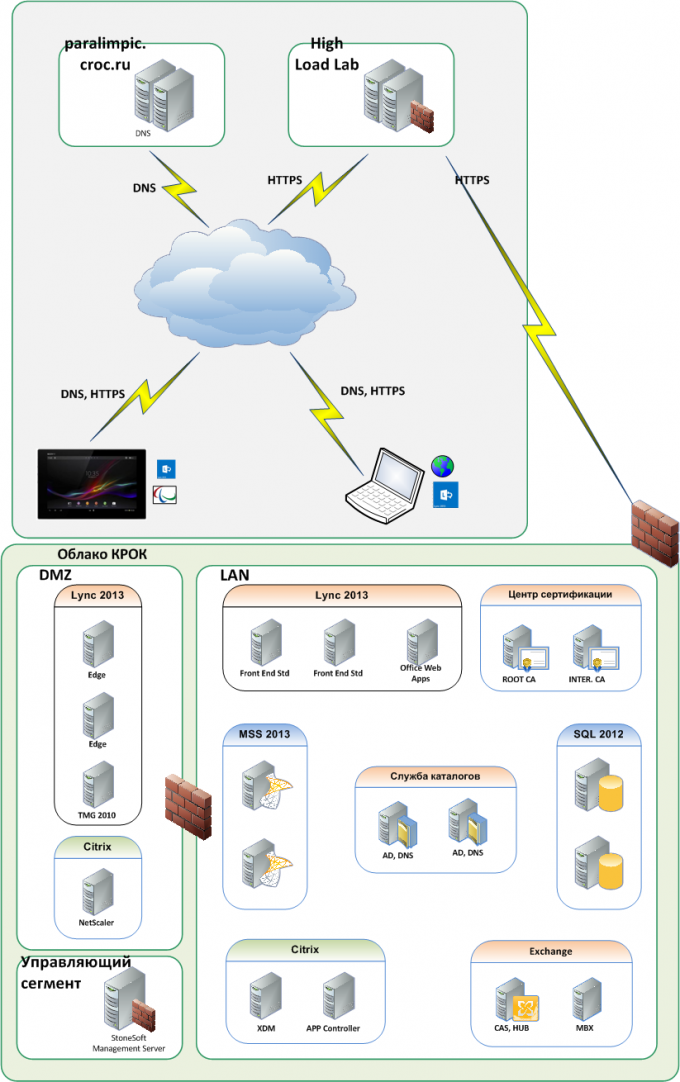


Рисунок 1.2. Архитектура ПО поддержки параолимпийских игр

Платформа реализации: android устройства, веб-интерфейс.

### 1.5.2. Сравнительный анализ

Для сравнительного анализа было выбрано вышеописанное программное обеспечения, разработанные для поддержки проведения параолимпийских игр, которое имеет некоторые сходства с разрабатываемой в рамках выпускной квалификационной работы программным обеспечением поддержки проведения «Фестиваля науки»: «Программно-аппаратный комплекс поддержки проведения параолимпийских игр».

Данная система достаточно полезна, включающая процессы, общения персонала. К плюсам данной системы относится

Сравнительная характеристика существующих информационных систем, разработанных для поддержки проведения мероприятий, приведена в таблице 1.1.

**Таблица 1.1**

**Сравнительная характеристика мобильного программного обеспечения по распознаванию городских объектов.**

|  |  |  |
| --- | --- | --- |
| Название системы  Функции | Программно-аппаратный комплекс поддержки проведения параолимпийских игр | Программный комплекс поддержки проведения «Фестиваля науки» |
| Предоставление оперативной сводки мероприятия | Есть | Есть |
| Информационная рассылка | Есть | Есть |
| Карта места проведения события | Есть | Есть |
| Связь персонала сообщениями | Есть | Есть |
|  | **Таблица 1.1** | **Продолжение** |
| Поддержка ОС Android | Есть | Есть |
| Поддержка ОС iOS | Нет | Есть |
| Поддержка ОС windows phone 8 | Нет | Есть |
| Планы зданий | Нет | Есть |
| Формирование отчётов | Нет | Есть |
| Создание и редактирование точек событий | Нет | Есть |
| Не требуется специализированные техника | Нет | Есть |

По результатам сравнительно анализа было выявлено, что недостатком системы поддержки проведения параолимпийских игр является ограниченный программный аппарат для использования, невозможность использовать приложение внутри зданий. Так же как и отсутствие необходимых отчётных данных.

Очевидно, что разработанная ранее система не способна полностью обеспечить процесс автоматизации основных задач, с которыми сталкиваются сотрудники мероприятия «Фестиваля науки», что позволяет сделать вывод о необходимости разработки полнофункционального программного обеспечения поддержки проведения «Фестиваля науки».

# ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ ПРОВЕДЕНИЯ «ФЕСТИВАЛЯ НАУКИ»

## 2.1. Функциональные возможности программного обеспечения поддержки проведения «Фестиваля науки»

В состав программного обеспечения поддержки проведения «Фестиваля науки» входят следующие подсистемы:

* подсистема хранения данных;
* веб-сервер;
* приложение клиента.

В программном комплексе имеется разграничение прав пользователей, которые при входе в систему могут выполнять определенные функции.

Рассмотрим, какими функциональными возможностями обладает каждая из подсистем отдельно.

**1. Функции подсистемы хранения данных**

Базы данных позволяют хранить большой объем информации некоторым упорядоченным образом. Подсистема хранения данных предназначена для:

* хранения большого объема информации системы;
* добавления, удаления или изменения хранимой информации;
* извлечения информации, удобном для человека виде;
* поиск необходимой информации по запросу пользователя системы.

**2. Функции веб-сервера**

Веб-сервер предоставляет возможность добавления, удаления и редактирования данных, а также для просмотра информации в зависимости от запроса пользователя:

* добавление, удаление, редактирование и просмотр информации в базу данных в зависимости от прав пользователя;
* генерация отчетности;
* отправка, принятие, обработка данных полученных от клиентов;
* поиск информации по запросу пользователя;
* ввод необходимой информации.

**3. Функции приложение клиента**

Приложение клиента имеет функционал:

* авторизация пользователя;
* генерация отчетности;
* предоставление оперативной информации о мероприятии;
* создание и удаление пользователей, карт, точек, событий в графическом редакторе.

Программное обеспечение поддержки проведения «Фестиваля науки» может обеспечивать публичный доступ гражданам Российской Федерации к открытому серверу через Интернет. Также, должна обеспечивать доступ пользователей системы к операционным данным БД ИС (путем предоставления сервисов, позволяющих формировать запросы на получение информации ограниченного доступа, в соответствии с уровнем компетентности пользователя).

Информационная система МБДОУ «Детский сад с.Яковлевка» должна обеспечивать публичный доступ гражданам Российской Федерации к открытой части информации через глобальную сеть Интернет.

## 2.2. Входные и выходные данные программном обеспечении поддержки проведения «Фестиваля науки»

Определим входные и выходные данные разработанной информационной системы.

**Входные данные.**

*1. Аутентификация пользователя*

Данные, вводимые пользователем при входе в систему:

* логин;
* пароль.

*2. Создание события*

Для создания события пользователь вводит следующие данные:

* название;
* описание;
* дата;
* активный.

*2. Создание карты*

Для создания карты пользователь вводит следующие данные:

* название;
* описание;
* событие;
* изображение карты.

*2. Создание точки*

Для создания точки пользователь вводит следующие данные:

* название;
* описание;
* карта;
* позиция по х;
* позиция по у;
* длина;
* ширина:
* тип.

*3. Создание пользователя*

Для создания пользователяпользователь вводит следующие данные:

* номер студенческого;
* ФИО;
* описание;
* роль;
* логин;
* пароль.

*4. Запись информации о группе*

Для внесения информации о группепользователь вводит следующие данные:

* событие;
* кол-во детей;
* кол-во ответственных;
* ФИО ответственных;
* учебное заведение;
* город;
* дату.

**Выходные данные:**

*1. Авторизация пользователя*

При авторизации пользователя происходит вход в рабочую область системы, где располагается карта события. Функциональные возможности зависят от прав вошедшего в систему пользователя.

*2. Сформированное событие*

В результате создается событие с картами и позициями на них;

*3. Пользователь*

В результате создается учётная запись для работы приложения;

*4. Отчёт*

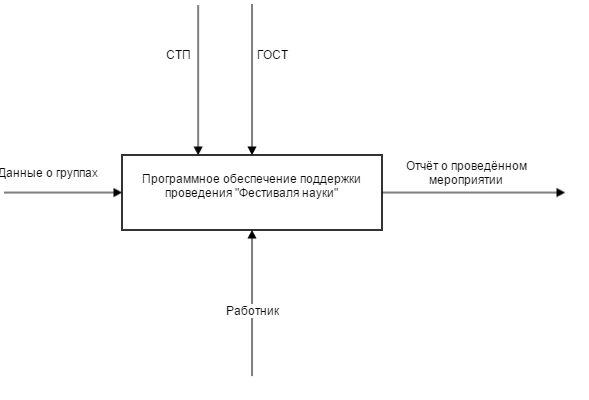
В результате формируется отчёт о посещаемости мероприятия;

## 2.3. Информационные потоки

На начальных этапах создания программного обеспечения требуется понять, как утроено мероприятие. Наиболее удобным языком моделирования бизнес-процессов является IDEF0 [32], где система представляется как совокупность взаимодействующих работ и функций. Для детального представления работы программного обеспечения отобразим на контекстной диаграмме функциональный блок, который соответствует одной из основных функций данной системы. При построении контекстной диаграммы IDEF0 выбирается точка зрения пользователя, ответственного за выполнение определенного процесса. (рис. 2.1.)

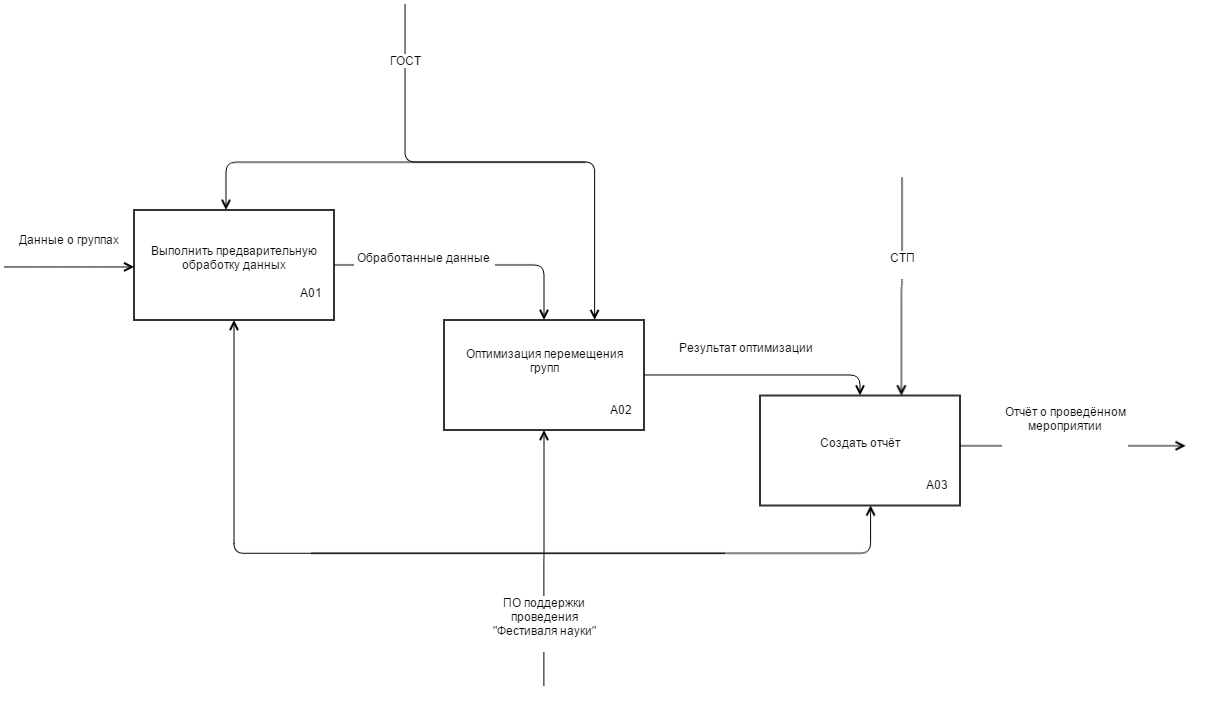
Точка зрения: пользователь

Процесс: добавление данных о группе



**Рис. 2.1. Контекстная диаграмма IDEF0 информационной системы.**

Функцию, приведенную на контекстной диаграмме можно разбить на более мелкие функции и описать их взаимодействие на диаграмме декомпозиции (рис 2.2).



**Рис. 2.2. Декомпозиция контекстной диаграммы информационной системы.**

Для описания достаточно крупных задач применяются контекстная диаграмма IDEF0. Однако с их помощью трудно понять, как выполняются задачи: синхронно или асинхронно, для начала выполнения новой задачи обязательно ли завершение всех остальных и т.д. Поэтому, для подробного описания работы той или иной функции используется диаграмма стандарта IDEF3 (рис. 2.3).



**Рис. 2.3. IDEF3-диаграмма работы информационной системы.**

Контекстная диаграмма IDEF0 и диаграмма стандарта IDEF3 в основном применяются для описания функционирования системы и обходят стороной существующие в ней информационные потоки. Для описания потоков данных используются диаграммы потоков данных (Data Flow Diagrams – DFD), которые представляют собой иерархию функциональных процессов, связанных с информационными потоками.



**Рис. 2.4. DFD-диаграмма потоков данных.**

Диаграмма потоков данных описывает процесс переработки информации от поступления в систему до выдачи пользователю. Информация поступает в систему от внешних сущностей и обрабатывается процессами, которые, в свою очередь, могут порождать новые информационные потоки. DFD представляет собой моделируемую систему как сеть связанных работ.

При разработке информационной системы необходимо провести объектно-ориентированный анализ, целью которого является создание объектной модели предметной области. UML (Unified Modeling Language) – унифицированный язык моделирования, являющийся языком широкого профиля, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью [28].

## 2.4. Структура классов

Центральное место в объектно-ориентированном программировании занимает разработка логической модели системы в виде диаграммы классов (class diagram), которая входит в состав языка моделирования UML. Она применяется для описания структуры системы, показывая ее классы, их атрибуты, операторы, а также взаимосвязи между классами. На рис. 2.4 представлена диаграмма классов сущностей. На рис. 2.5 диаграмма классов сервера программного обеспечения поддержки проведения «Фестиваля науки».



**Рис. 2.5. Диаграмма классов, сущности.**



**Рис. 2.6. Диаграмма классов, сервер.**

## 2.5. Варианты использования системы.

Для наглядного представления функциональных возможностей групп пользователей системы часто используются диаграммы вариантов использования. На рис. 2.7 представлена диаграмма вариантов использования ПО поддержки проведения «Фестиваля науки».



**Рис. 2.7. Диаграмма вариантов использования.**

В результате построения диаграммы вариантов использования было выявлено, что в системе существует пять основных ролей:

* гость;
* администратор;
* встречающий
* гид;
* управляющий;

Разграничение прав доступа пользователей системы реализовано в соответствии с разделением пользователей на группы. У каждой группы имеется свой функциональный набор. Рассмотрим варианты использования для каждой из этих ролей. «Гость» имеет право на просмотр общих сведений о мероприятии и просмотр информационной рассылки. Пользователи, принадлежащие роли «Гид» расширяют роль «Гость» и имеют следующие функциональные возможности:

* вход в систему по своим учетным данным (ввод логина и пароля);
* получение информации о мероприятии;
* принимать встречу;
* активировать точки экскурсий

Роли «Встречающий» расширяют роль «Гость» и имеют следующий набор функций:

* вход в систему по своим учетным данным (ввод логина и пароля);
* получение информации о мероприятии;
* назначить встречу;
* Ввести данные группы.

Роли «Управляющий» расширяют роль «Гость» и имеют следующий набор функций:

* вход в систему по своим учетным данным (ввод логина и пароля);
* получение информации о мероприятии;
* ввод новостей
* Создать позицию для «Встречающего»

Роли «Администратор» подготавливают программу к работе: вводят данные по событиям и картам, добавляют активные точки.

# ГЛАВА 3 ОПИСАНИЕ РЕАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ ПРОВЕДЕНИЯ «ФЕСТИВАЛЯ НАУКИ»

## 3**.1. Выбор программного обеспечения.**

Для реализации программного обеспечения использовались платформы: Microsoft .NET Framework, Java EE, с применением технологии Spring MVC, языки программирования: C#, Java, средства разработки: приложения клиента: Unity3D, Microsoft Visual Studio Professional 2012, web-сервера: IntelliJ IDEA 14.0, СУБД MS SQLServer 2008 Management Studio, метод взаимодействия в сети Интернет REST, тип отправляемых пакетов JSON.

### 3**.1.1. Используемые платформы.**

#### 3.1.1.1. .NET Framework

.NET Framework — это фреймворк для создания приложений под ОС Windows, Windows Phone, Windows Server и Windows Azure. Она состоит из среды CLR и библиотеки классов .NET Framework, которая содержит классы, интерфейсы и типы значений, поддерживающие широкий диапазон технологий. Платформа .NET Framework предоставляет управляемую среду выполнения, которая упрощает разработку и развертывание приложений, а также возможности интеграции с различными языками программирования, включая Visual Basic и Visual C#. При разработке платформы .NET Framework учитывались следующие цели.

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Обеспечение среды выполнения кода, минимизирующей конфликты при развертывании программного обеспечения и управлении версиями.
* Обеспечение среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем.
* Обеспечение среды выполнения кода, исключающей проблемы с производительностью сред выполнения сценариев или интерпретируемого кода.
* Обеспечение единых принципов работы разработчиков для разных типов приложений, таких как приложения Windows и веб-приложения.
* Разработка взаимодействия на основе промышленных стандартов, которое обеспечит интеграцию кода платформы .NET Framework с любым другим кодом.

[**Возможности среды CLR**](javascript:void(0))

Среда CLR управляет памятью, выполнением потоков, выполнением кода, проверкой безопасности кода, компиляцией и другими системными службами. Эти средства являются внутренними для управляемого кода, который выполняется в среде CLR. Она обеспечивает:

* управление доступом для кода;
* надежность кода, реализуя инфраструктуру строгой типизации и проверки кода;
* исключает многие часто возникающие проблемы с программным обеспечением.
* поддерживает сегодняшнее и вчерашнее программное обеспечение.
* может размещаться в высокопроизводительных серверных приложениях, таких как Microsoft SQL Server и службы IIS (Internet Information Services).

[**Библиотека классов .NET Framework**](javascript:void(0))

Библиотека классов платформы .NET Framework представляет собой коллекцию типов, которые тесно интегрируются со средой CLR. Библиотека классов является объектно-ориентированной; предоставляя типы, из которых управляемый код пользователя может наследовать функции. Это не только упрощает работу с типами .NET Framework, но также уменьшает время, затрачиваемое на изучение новых средств платформы .NET Framework. Кроме того, компоненты независимых производителей можно легко объединять с классами платформы .NET Framework.

#### **3.1.1.2. Java EE**

Java Platform, Enterprise Edition, сокращенно Java EE — набор спецификаций и соответствующей документации для языка [Java](https://ru.wikipedia.org/wiki/Java), описывающей архитектуру серверной платформы для задач средних и крупных предприятий.

Спецификации детализированы настолько, чтобы обеспечить переносимость программ с одной реализации платформы на другую. Основная цель спецификаций — обеспечить масштабируемость приложений и целостность данных во время работы системы. JEE во многом ориентирована на использование её через веб как в интернете, так и в локальных сетях. Вся спецификация создаётся и утверждается через JCP (Java Community Process) в рамках инициативы Sun Microsystems Inc.

JEE является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость, гибкость.

Сервер приложений J2EE (часто называемый J2EE-контейнер) — это реализация системы в соответствии со спецификацией J2EE, обеспечивающая работу модулей с логикой конкретного приложения. Включает в себя как минимум следующие сервисы:

* EJB-контейнер, который поддерживает автоматическую синхронизацию Java-объектов с базой данных (CMP — container managed persistence, BMP — bean managed persistence);
* JMS — сервис доставки сообщений между компонентами и серверами;
* управление ресурсами (доступ к СУБД, файловой системе, почтовому серверу и т. д.);
* безопасность и защита данных;
* поддержка транзакций (в том числе и распределённых, двухфазных). См. Java Transaction API.
* веб-сервер и сервлет-сервер;
* поддержка веб-сервисов.
* JSF

Разработчики J2EE-приложений также пишут свои программы в соответствии с спецификациями J2EE, что обеспечивает их работу внутри таких серверов.

Технологию J2EE рационально использовать для реализации крупных проектов, например, для организации сложных веб-порталов и предоставления услуг онлайн, особенно если необходимо обеспечить бесперебойную работу многих тысяч пользователей.

### 3.1.2. Используемая технологии

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы.

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первый стабильный релиз 1.0 был выпущен в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 4.0.

Между тем, особенности ядра Spring Framework применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе.

Spring Framework может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании. Эти фреймворки делятся на структурные элементы типовых комплексных приложений:

* Inversion of Control-контейнер: конфигурирование компонентов приложений и управление жизненным циклом Java-объектов;
* фреймворк аспектно-ориентированного программирования: работает с функциональностью, которая не может быть реализована возможностями объектно-ориентированного программирования на Java без потерь;
* фреймворк доступа к данным: работает с системами управления реляционными базами данных на Java-платформе, используя JDBC- и ORM-средства и обеспечивая решения задач, которые повторяются в большом числе Java-based environments;
* фреймворк управления транзакциями: координация различных API управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java;
* фреймворк MVC: каркас, основанный на HTTP и сервлетах, предоставляющий множество возможностей для расширения и настройки;
* фреймворк удалённого доступа: конфигурируемая передача Java-объектов через сеть в стиле RPC, поддерживающая RMI, CORBA, HTTP-based протоколы, включая web-сервисы;
* фреймворк аутентификации и авторизации: конфигурируемый инструментарий процессов аутентификации и авторизации, поддерживающий много популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект Spring Security;
* фреймворк удалённого управления: конфигурируемое представление и управление Java-объектами для локальной или удалённой конфигурации с помощью JMX;
* фреймворк работы с сообщениями: конфигурируемая регистрация объектов-слушателей сообщений для прозрачной обработки сообщений из очереди сообщений с помощью JMS, улучшенная отправка сообщений по стандарту JMS API;
* тестирование: каркас, поддерживающий классы для написания модульных и интеграционных тестов.

**Spring MVC**

Spring имеет собственную MVC-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать ее как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного Apache Struts, а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.



**Рис. 3.1. Шаблон проектирования Model-View-Controller.**

Класс DispatcherServlet является основным контроллером фрэймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.

Как и Struts, Spring MVC является фреймворком, ориентированным на запросы. В нем опредены стратегические интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса — быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают. MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с Servlet API. Эта связь рассматривается некоторыми как неспособность разработчиков Spring предложить для веб-приложений абстракцию более высокого уровня. Однако эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая все же работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:

* HandlerMapping: выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.
* HandlerAdapter: вызов и выполнение выбранного метода обработки входящего запроса.
* Controller: включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.
* View: ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.
* ViewResolver: выбор, какое именно View должно быть показано клиенту.
* HandlerInterceptor: перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется DispatcherServlet-ом).
* LocaleResolver: получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.
* MultipartResolver: обеспечивает Upload — загрузку на сервер локальных файлов клиента.

Spring MVC предоставляет разработчику следующие возможности:

* Ясное и прозрачное разделение между слоями в MVC и запросах.
* Стратегия интерфейсов — каждый интерфейс делает только свою часть работы.
* Интерфейс всегда может быть заменен альтернативной реализацией.
* Интерфейсы тесно связаны с Servlet API.
* Высокий уровень абстракции для веб-приложений.

В веб-приложениях можно использовать различные части Spring Framework, а не только Spring MVC.

Model-view-controller (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Основная цель применения этой концепции состоит в разделении бизнес-логики от её визуализации. За счет такого разделения повышается возможность повторного использования. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.
2. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

### 3.1.3 Используемые языки программирования

**C#**

**C#** — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR.

**Java**

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Дата официального выпуска — 23 мая 1995 года.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

### 3.1.4 Средства разработки приложения клиента

#### 3.1.4.1 Unity3D

Unity — это инструмент для разработки двух- и трёхмерных приложений и игр. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL.

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку приложения прямо в редакторе. Движок поддерживает три сценарных языка: C#, JavaScript, Boo. Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены — отдельные файлы, со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты, так и пустые игровые объекты — объекты, которые не имеют модели. Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название, может быть тег и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя — будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели. Редактор Unity поддерживает написание и редактирование шейдеров. Редактор Unity имеет компонент для создания анимации, но также анимацию можно создать предварительно в 3D-редакторе и импортировать вместе с моделью, а затем разбить на файлы.

Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат .unityassets и передавать другим разработчикам, или выкладывать в свободный доступ.

#### 3.1.4.2. Microsoft Visual Studio Professional 2012

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения.

### 3.1.5 Средства разработки приложения сервера

IntelliJ IDEA — коммерческая интегрированная среда разработки программного обеспечения на многих языках программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия IntelliJ IDEA появилась в январе 2001 года и быстро приобрела популярность, как первая Java IDE с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя им сконцентрироваться на разработке функциональности, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Среди прочих возможностей, IntelliJ IDEA хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit.

Начиная с версии 9.0, IntelliJ IDEA доступна в двух версиях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0. В ней реализована полная поддержка Java SE, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В версии Ultimate Edition реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

#### Используемая СУБД

Microsoft SQL Server — система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Transact-SQL (T-SQL) — процедурное расширение языка SQL.

SQL был расширен такими дополнительными возможностями как:

* управляющие операторы,
* локальные и глобальные переменные,
* различные дополнительные функции для обработки строк, дат, математики и т. п.,
* поддержка аутентификации Microsoft Windows

Язык Transact-SQL является ключом к использованию MS SQL Server. Все приложения, взаимодействующие с экземпляром MS SQL Server, независимо от их реализации и пользовательского интерфейса, отправляют серверу инструкции Transact-SQL.

Как и в языках программирования, в SQL существуют различные типы данных для хранения переменных:

* числа — для хранения числовых переменных (bit, int, tinyint, smallint, bigint, numeric, decimal, money, smallmoney, float, real).
* даты — для хранения даты и времени (datetime, smalldatetime).
* символы — для хранения символьных данных (char, nchar, varchar, nvarchar).
* двоичные — для хранения бинарных данных (binary, varbinary).
* большеобъемные — типы данных для хранения больших бинарных данных (text, ntext, image).
* специальные — указатели (cursor), 16-байтовое шестнадцатеричное число, которое используется для GUID (uniqueidentifier), штамп изменения строки (timestamp), версия строки (rowversion), таблицы (table).

SQL Server Management Studio — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

#### Проектирование базы данных системы

Одним из основополагающих этапов построения систем является проектирование базы данных.

База данных программного обеспечения поддержки проведения «Фестиваля науки» состоит из 9 таблиц. Описание таблиц приведены ниже:

**Таблица 3.1**

**Описание таблицы event – таблица событий**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| name | Текстовый | Название события |
| description | Текстовый | Описание события |
| date | Дата | Дата проведения события |
| isActiv | Целочисленный | Активно ли событие |

**Таблица 3.2**

**Описание таблицы maps – таблица карт**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| id\_event | Целочисленный | Код события |
| name | Текстовый | Название карты |
| description | Текстовый | Описание карты |

**Таблица 3.3**

**Описание таблицы point – таблица активных точек**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| name | Текстовый | Название события |
| description | Текстовый | Описание события |
| type | Текстовый | Тип точки |
| Х | Дробное число | Позиция точки по Х |
| У | Дробное число | Позиция точки по У |
| size\_w | Дробное число | Длина точки |
| size\_h | Дробное число | Ширина точки |
| isBusy | Целочисленный | Занята ли точка |
| id\_user\_Busy | Целочисленный | Код пользователя |

**Таблица 3.4**

**Описание таблицы users – таблица пользователей**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| Gradebook | Целочисленный | Номер студенческого |
| name | Текстовый | ФИО пользователя |
| description | Текстовый | Описание пользователя |
| role | Текстовый | Роль учётной записи |
| login | Текстовый | Логин учётной записи |
| password | Текстовый | Пароль учётной записи |

**Таблица 3.5**

**Описание таблицы news – таблица новостей**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| id\_event | Целочисленный | Код события |
| description | Текстовый | Описание новости |
| date\_write | Дата | Дата написания |

**Таблица 3.6**

**Описание таблицы message – таблица сообщений**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| idEvent | Целочисленный | Код события |
| message | Текстовый | Сообщение |
| Date | Дата | Дата написания |
| idUser | Целочисленный | Код пользователя, отправитель |
| idUserTo | Целочисленный | Код пользователя, кому отправленно |

**Таблица 3.7**

**Описание таблицы busy – таблица занятых точек**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| idUser | Целочисленный | Код пользователя |
| idPoint | Целочисленный | Код точки |
| idEvent | Целочисленный | Код события |

**Таблица 3.8**

**Описание таблицы log – таблица логов**

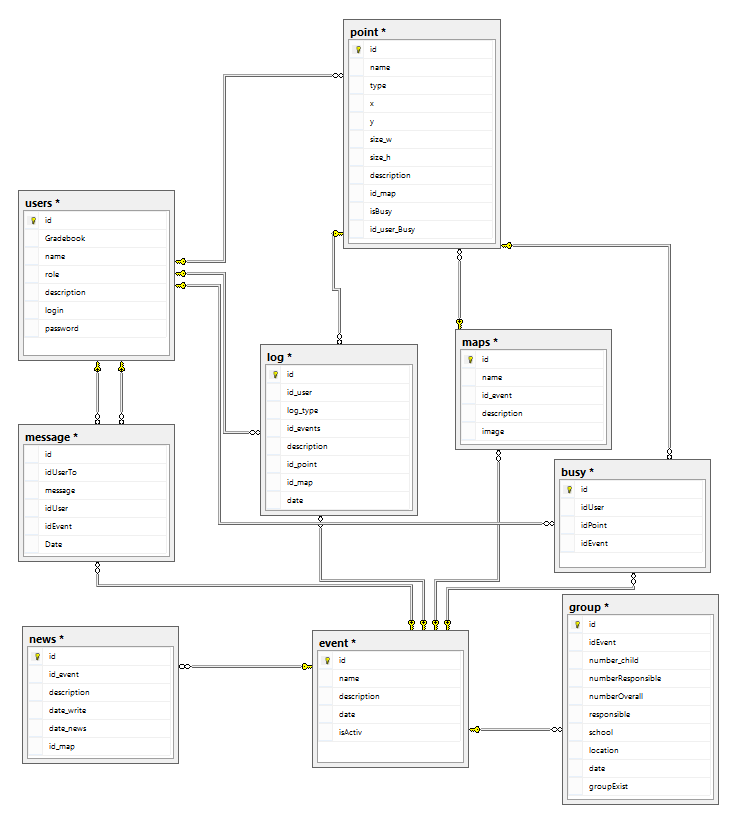
| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| id\_user | Целочисленный | Код пользователя |
| log\_type | Текстовый | Тип лога |
| id\_events | Целочисленный | Код события |
| description | Текстовый | Описание лога |
| id\_point | Целочисленный | Код точки |
| date | Дата | Дата логирования |

**Таблица 3.9**

**Описание таблицы event – таблица событий**

| Наименование поля | Тип данных | Описание |
| --- | --- | --- |
| id | Целочисленный | Идентификатор |
| idEvent | Целочисленный | Код события |
| number\_child | Целочисленный | Кол-во детей |
| numberResponsible | Целочисленный | Кол-во ответственных |
| numberOverall | Целочисленный | Общее кол-во людей |
| responsible | Текстовый | Ответственный |
| school | Текстовый | Учебное заведение |
| location | Текстовый | Город |
| date | Дата | Дата прихода |
| groupExist | Целочисленный | Наличие группы |

На рис. 3.2 приведена структура базы данных, в котором реализуется один видсвязи: «Один к одному»



**Рис. 3.2. Структура базы данных.**

### 3.1.7. Метод взаимодействия в сети

REST (Representational State Transfer — «передача репрезентативного состояния») — метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса [1].

В широком смысле REST поддерживает концепцию построения распределённого приложения, при которой компоненты взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине [2].

В качестве необходимых условий для построения распределенных REST-приложений [23] Филдинг перечислил следующие:

* Клиент-серверная архитектура.
* Сервер не обязан сохранять информацию о состоянии клиента.
* В каждом запросе клиента должно явно содержаться указание о возможности кэширования ответа и получения ответа из существующего кэша.
* Клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. При этом клиент может не знать о существовании промежуточных узлов, за исключением случаев передачи конфиденциальной информации.
* Унифицированный программный интерфейс сервера. Филдинг приводил URI в качестве примера формата запросов к серверу, а в качестве примера ответа сервера форматы HTML, XML и JSON, различаемые с использованием идентификаторов MIME.

Филдинг указывал, что приложения, не соответствующие приведённым условиям, не могут называться REST-приложениями. Если же все условия соблюдены, то, по его мнению, приложение получит следующие преимущества:

* надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
* производительность (за счет использования кэша);
* масштабируемость;
* прозрачность системы взаимодействия, особенно необходимая для приложений обслуживания сети;
* простота интерфейсов;
* портативность компонентов;
* легкость внесения изменений;
* способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

### 3.1.8. Тип отправляемых пакетов

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом.

Несмотря на происхождение от JavaScript, формат считается языконезависимым и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

За счёт своей лаконичности по сравнению с XML, формат JSON может быть более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, в таком ключе он уместен в задачах обмена данными как между браузером и сервером , так и между самими серверами.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

* Набор пар ключ: значение. В различных языках это реализовано как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением — любая форма.
* Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных: как правило, любой современный язык программирования поддерживает их в той или иной форме. Они легли в основу JSON, так как он используется для обмена данными между различными языками программирования.

В качестве значений в JSON используются структуры:

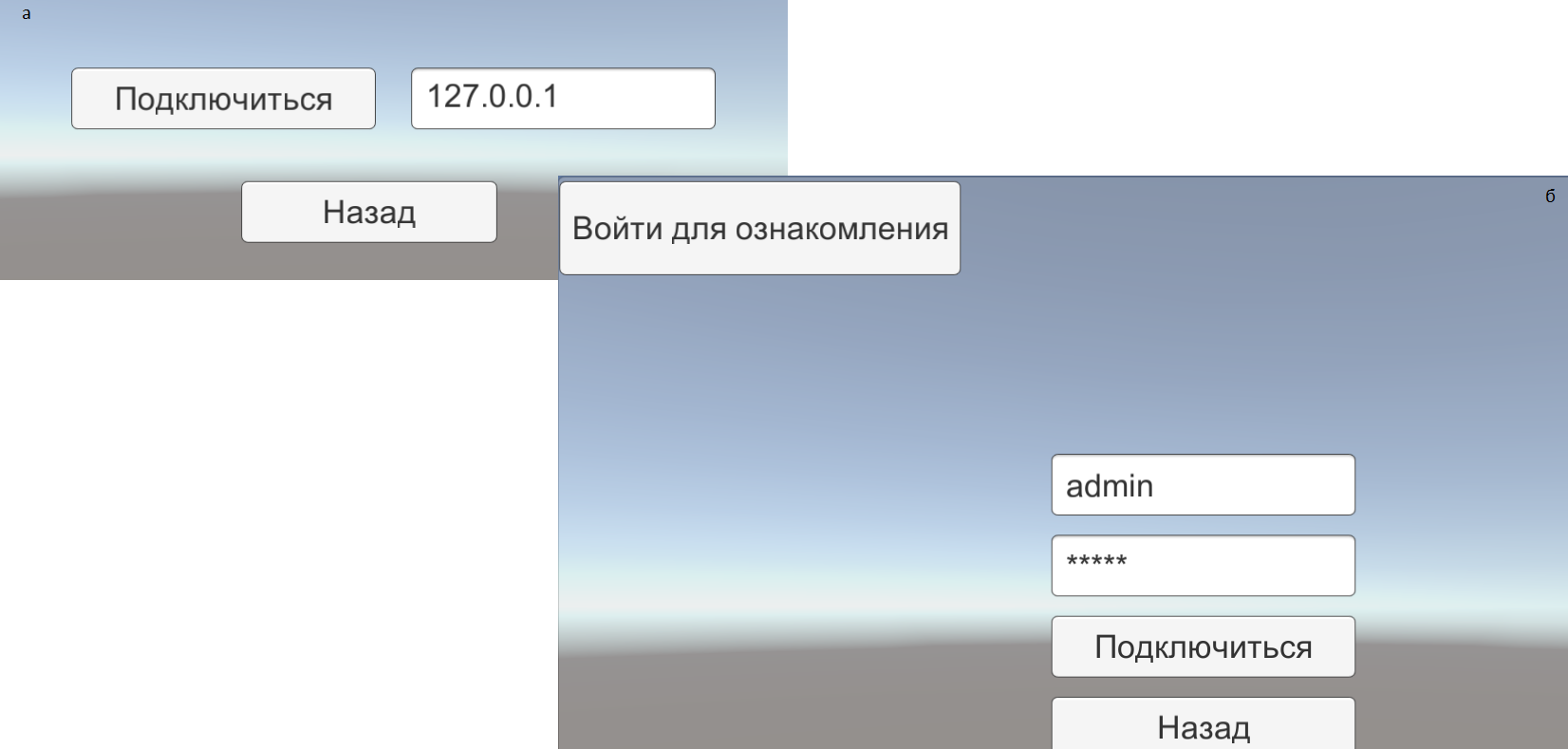
* Объект — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* Массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми.
* Значение может быть строкой в двойных кавычках, числом, объектом, массивом, одним из литералов: true, false или null. Т.о. структуры могут быть вложены друг в друга.
* Строка — это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\».

Строка очень похожа на одноимённый тип данных в языках С и Java. Число тоже очень похоже на С- или Java-число, за исключением того, что используется только десятичный формат. Пробелы могут быть вставлены между любыми двумя синтаксическими элементами.

### 3.2. Описание интерфейса и примеры работы приложения.

Интерфейс программного обеспечения разработан в максимально простой и понятной форме и обеспечивает удобный доступ к функциям системы.

Для входа в приложение, необходимо ввести ip адрес веб-сервера, и пройти аутентификация, введя логин и пароль. Или нажать кнопку «Войти для ознакомления»

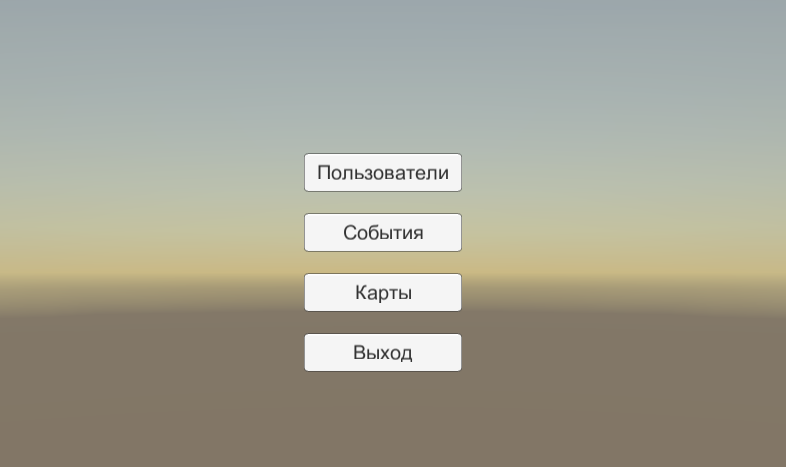


**Рис. 3.3. Процесс аутентификация.**

### 3.2.1. Описание интерфейса редактора

Интерфейс редактора доступен только пользователям с ролей «Администратор».

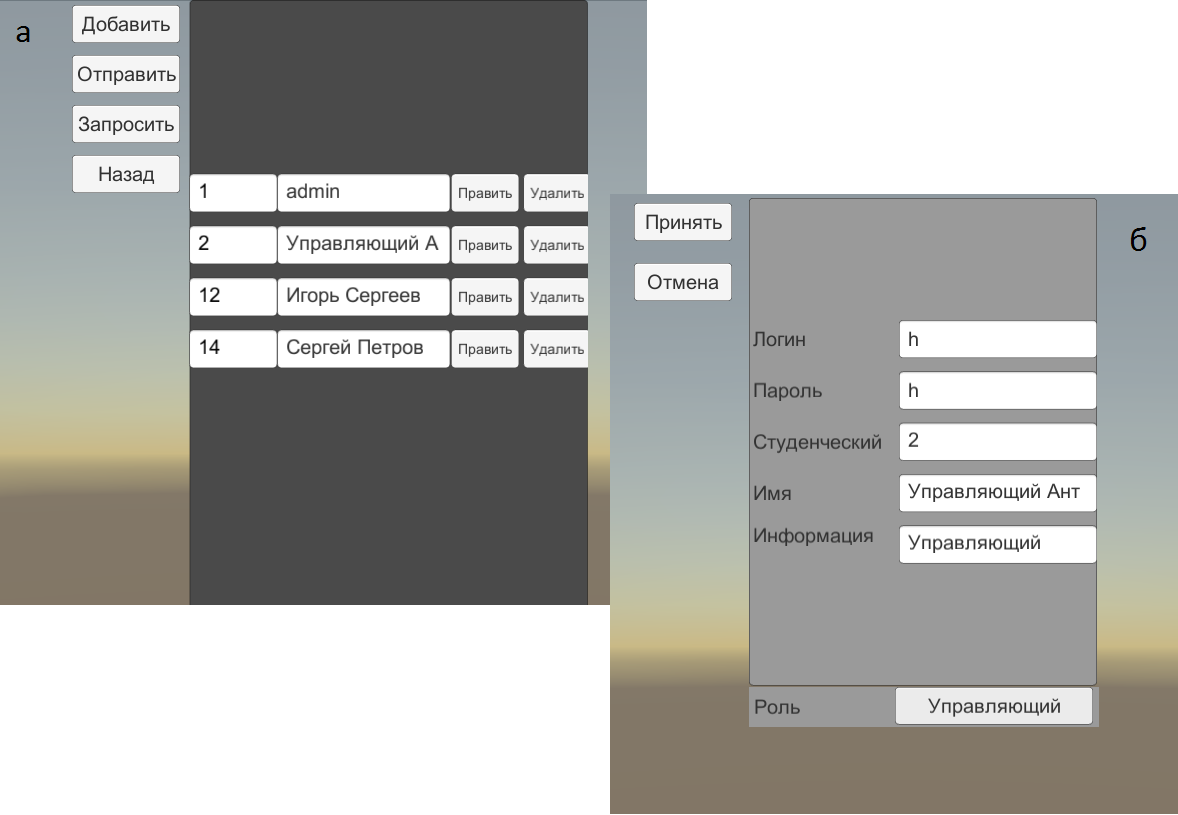
При входе в приложение становится доступной панель с вариантами изменения дынных



**Рис. 3.4. Процесс выбора вариантов редактирования.**

Процесс изменения данных пользователей состоит из:

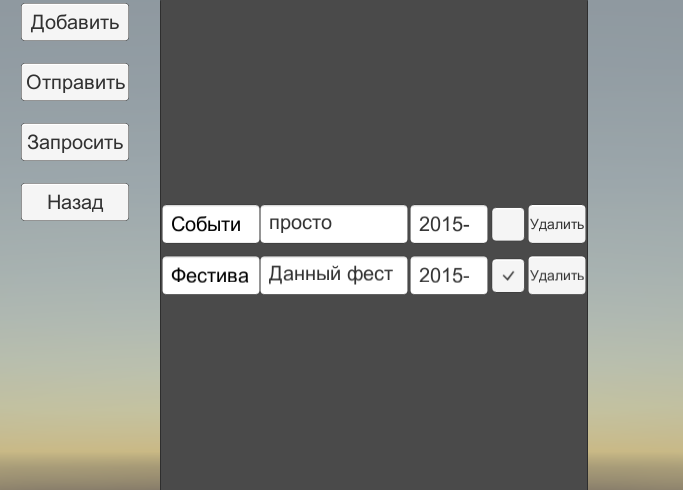
1. запроса данных пользователе й;
2. изменение полученных данных, путём вызова панели редактирования пользователя;
3. отправка данных пользователей на сервер.



**Рис. 3.5. Процесс редактирования пользователей (а – запрос данных; б – панель редактирования).**

Процесс изменения данных событий состоит из:

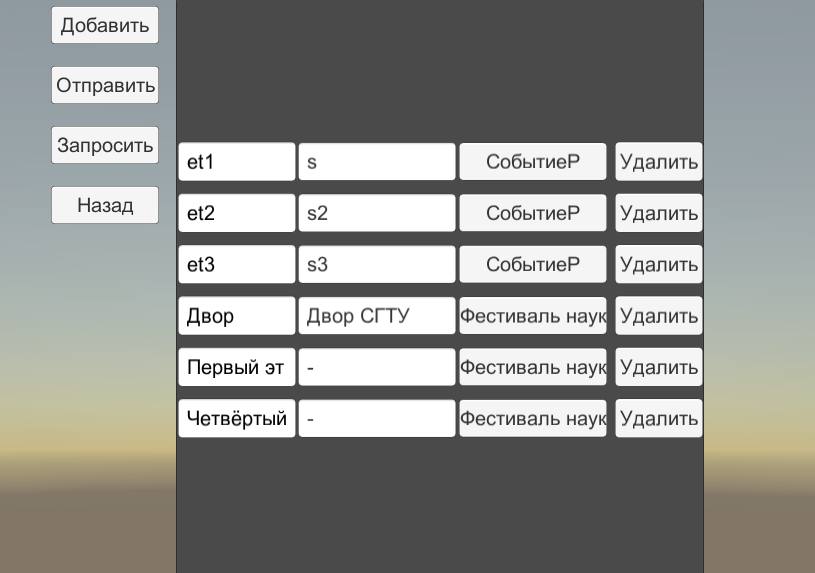
1. запроса данных событий;
2. изменение полученных данных в текущей панели;
3. отправка данных событий на сервер.



**Рис. 3.6. Процесс редактирования событий.**

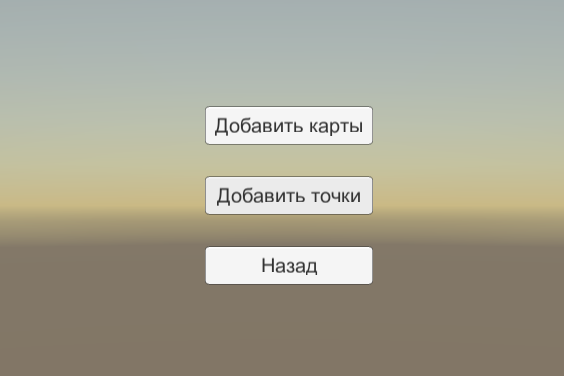
Процесс изменения данных карт состоит из:

1. запроса данных карт;
2. изменение полученных данных в текущей панели;
3. отправка данных карт на сервер.



**Рис. 3.7. Процесс редактирования событий.**

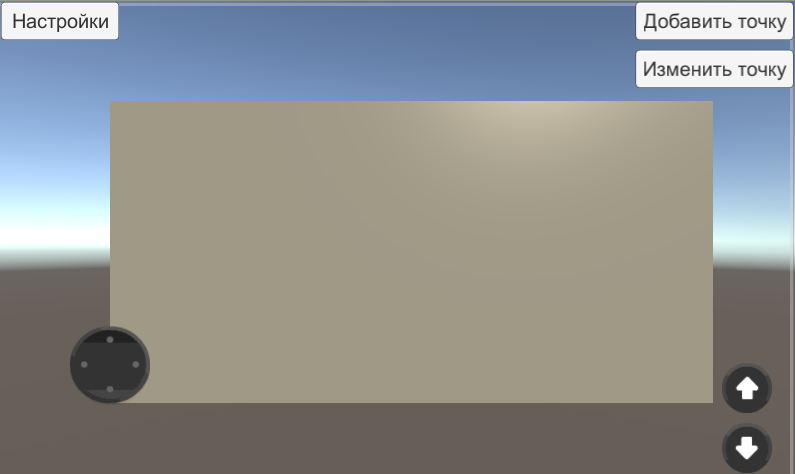
Администратор может также создавать активные точки а менять изображения карт. Для этого требуется нажать кнопку «Добавить точки»



**Рис. 3.8. Процесс перехода в редактор точек.**

После перехода в редактор точек, откроется новая сцена с функциональными объектами:

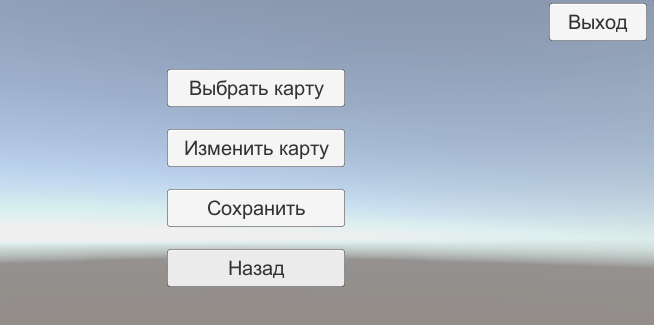
* джойстик передвижении камеры;
* кнопки изменения масштабирования;
* кнопки добавления и изменения точек;
* кнопка настройки.



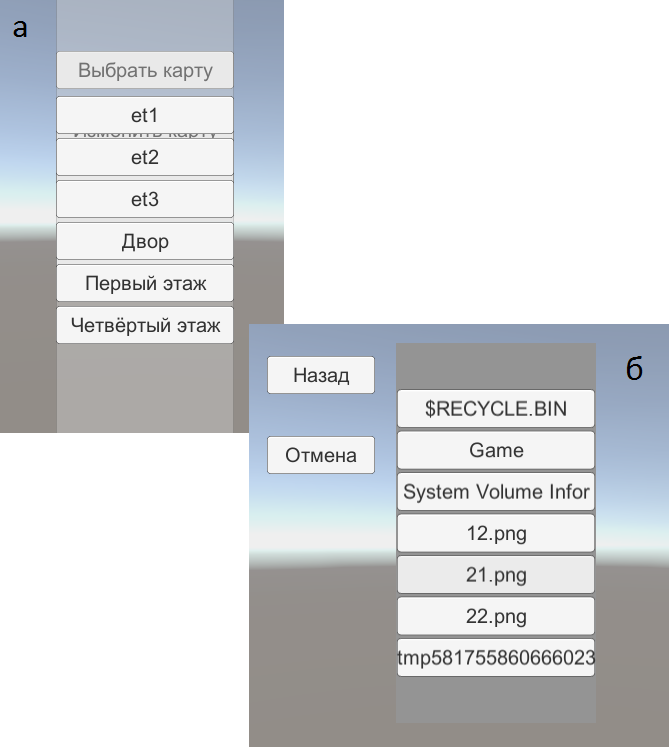
**Рис. 3.9. Панель редактирования.**

Кнопка настройки открывает меню со следующим функционалом:

* кнопка «Выход» производит выход из приложения:
* кнопка «Выбрать карту» производит выбор карты с сервера;
* кнопка «Изменить карту» производит изменение изображения карты;
* кнопка «Сохранить» сохраняет настройки выбранной карты;
* кнопка «Назад» возвращает в панель редактирования



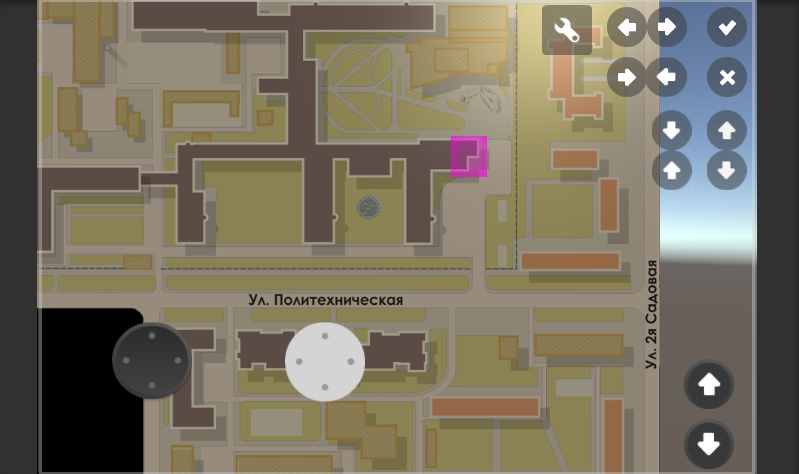
**Рис. 3.10. Панель настроек.**

****

**Рис. 3.11. Панель настроек (а – панель выбора карты с сервера; б – панель выбора изображения карты).**

При добавлении точки становится доступным новый джойстик для перемещения точки, а также кнопки редактирования:

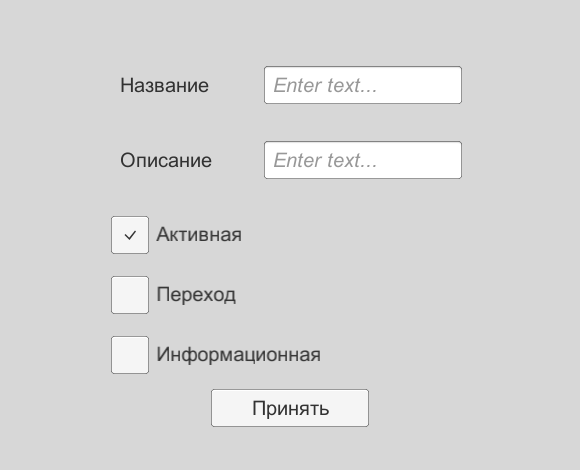
* увеличение и уменьшения длинны точки;
* увеличение и уменьшения ширины точки;
* фиксирование точки;
* удаление точки;
* изменение данных точки.



**Рис. 3.12. Панель редактирования с дополнительными функциями.**

При изменении данных точки открывается панель со следующими параметрами:

* название точки;
* описание точки;
* тип точки:
  + активная – тип точки, используемый для описания аудиторий мастер классов, предназначена для ознакомления остальными участниками и непосредственной работай «Гида»;
  + переход – тип точки, предназначенный лаконичного перехода между картами;
  + информационная - тип точки, необходимый для описания стендов, и различной информации, требуется для ознакомления.



**Рис. 3.13. Панель редактирования данных точки.**

#### 3.2.1. Описание интерфейса основной программы

Для аутентифицированных пользователей доступны интерактивная панель с интерпретированной картой, имеющая следующий функционал:

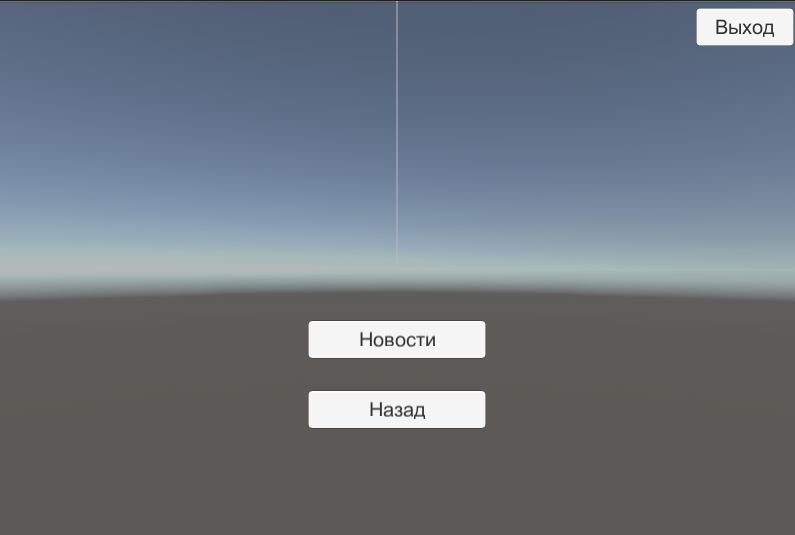
* джойстик передвижении камеры;
* кнопки изменения масштабирования;
* кнопка выбора карты для смены отображения;
* кнопка дополнительных функций.



**Рис. 3.13. Интерактивная панель.**

Кнопка дополнительных функций открывает панель со следующим функционалом:

* Роль «Гость»:
  + - новости мероприятия;
    - выход.
* Роль «Гид» расширяет гостевой функционал добавляя чат.
* Роль «Встречающий» расширяет гостевой функционал:
  + - чат;
    - создание встречи
* Роль «Управляющий» расширяет гостевой функционал:
  + - чат;
    - создание позиции встречающего;
    - формирование отчётности.

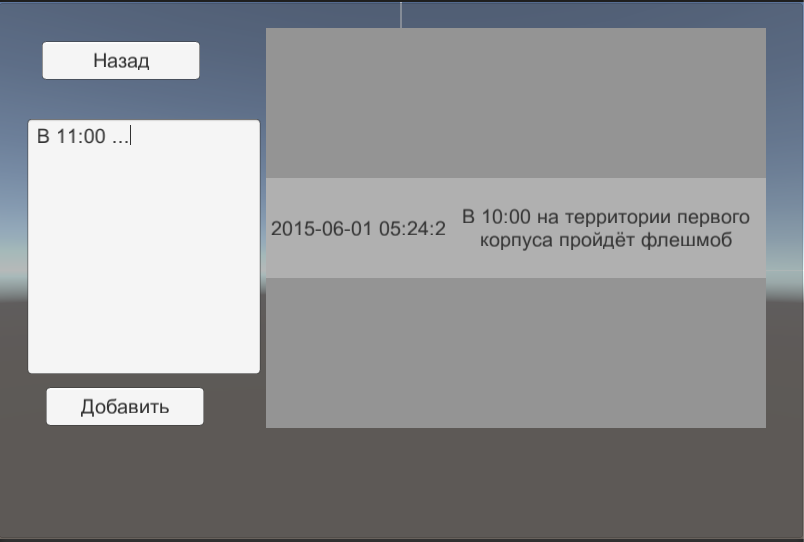


**Рис. 3.14. Панель дополнительных функций «Гость».**



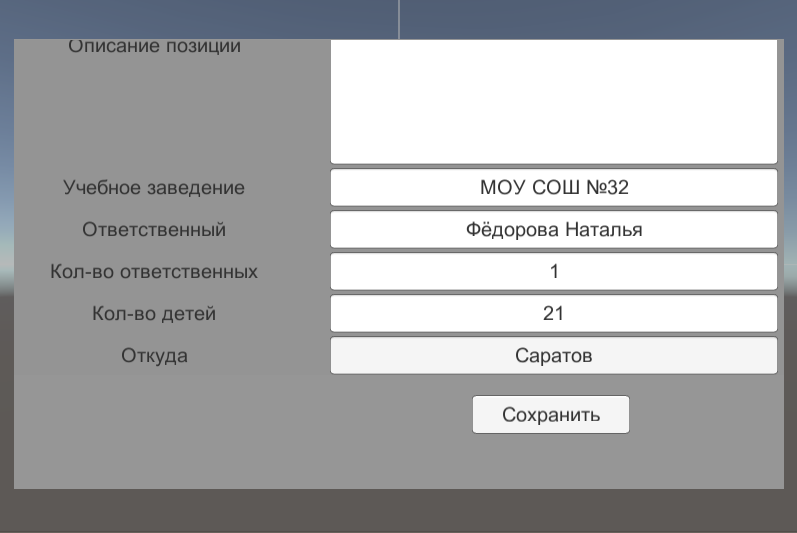
**Рис. 3.14. Панель дополнительных функций «Управляющий».**

Пользователь «Упраляющий» может рассылать персоналу новости о мероприятии



**Рис. 3.15. Панель новостной рассылки, роль «Управляющий».**

Пользователь с ролью «Встречающий» занимает позицию и добавляет новоприбывшую группу.



**Рис. 3.16. Добавление информации о группе**

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы было написано программное обеспечение для автоматизации «Фестиваля науки» СГТУ им. Ю. А. Гагарина, которое позволяет улучшить управление людскими ресурсами и снизить трудоёмкость сбора статистики мероприятия.

Решение данной задачи выполнялось с помощью клиент-серверного приложения. Разработанное программное обеспечение позволяет создавать интерактивные объекты карты, загружать новые изображения и формировать отчёты.

Программное обеспечение клиента разработано в среде Unity3D используя Microsoft Visual Studio 2012 на языке C#, приложение сервера разработанно в IntelliJ IDEA 14.

Разработанное программное обеспечение может быть использовано для поддержки проведения «Фестиваля науки»

Основными пользователями программного обеспечения являются люди, отвечающие за проведени.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Архитектура REST [Электронный ресурс]. – 2014 – URL: <http://habrahabr.ru/company/croc/blog/219695/> (Дата обращения 16.06.2014).
2. Архитектура REST [Электронный ресурс]. – 2008 – URL: http://habrahabr.ru/post/38730/ (Дата обращения 14.05.2015).
3. Хо К. Spring 3 для профессионалов [Текст] / К. Хо, Р. Харроп - Издательский дом “Вильямс”, 2012. 880.
4. Смартфон [Электронный ресурс]. – 2014,URL: http://ru.wikipedia.org/wiki/Смартфон (Дата обращения: 23.01.2015)
5. Операционная система [Электронный ресурс]. – 2014, URL: http://ru.wikipedia.org/wiki/Операционная\_система (Дата обращения: 23.01.2014)
6. Android. [Электронный ресурс]. – 2014,URL: http://www.android.com/meet-android/ (Дата обращения: 30.01.2015)
7. Apple-iOS7. [Электронный ресурс]. – 2014,URL: https://www.apple.com/ru/ios/ (Дата обращения: 12.02.2015)
8. Windows Phone. [Электронный ресурс]. – 2014,URL: http://ru.wikipedia.org/wiki/Windows\_Phone (Дата обращения: 25.09.2014)
9. Лигуори Р. Java 8. Карманный справочник [Текст] / Р. Лигуори, П. Лигуори - Издательский дом “Вильямс”, 2014. 256.
10. Руководство Unity [Электронный ресурс]. – 2015 – URL: https://components.xamarin.com/gettingstarted/json.net (Дата обращения 11.03.2015).
11. Json.NET [Электронный ресурс]. – 2015 – URL: http://docs.unity3d.com/ru/current/Manual/ (Дата обращения 11.01.2015).
12. Джеймс Р. SQL: полное руководство[Текст] / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель - Издательский дом “Вильямс”, 2015. 960 с.
13. Кригель А. SQL. Библия пользователя. Язык запросов SQL, 2-е издание [Текст] / А. Кригель, Б. Трухнов- Издательский дом “Вильямс”, 2009. 752с.
14. @RequestMapping [Электронный ресурс]. – 2015 – URL: http://ru.java.wikia.com/wiki/@RequestMapping (Дата обращения 15.03.2015).
15. Шилдт Г. Полный справочник по C# [Текст] / Г. Шилдт. – М.: Издательский дом “Вильямс”, 2004. – 752 с. : ил.
16. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4 [Текст] / Э. Троелсен; пер. с анг. Я. Волкова, А. Моргунова, Н. Мухина - Издательский дом “Вильямс”, 2011. 1392 с.
17. Эккель Б. Философия Java [Текст] / Б. Эккель; пер. с анг. Е. Матвеев – Питер: Питер, 2009. 640 с.
18. Макконнелл С. Совершенный код [Текст] /С. Макконнелл; пер. с анг. В. Вшивцев – Питер: Питер, 2008. 896 с.
19. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# [Текст] / Д. Рихтер – Питер: Питер, 2012. 928 с.
20. Виейр Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс [Текст] / Р. Виейр - Издательский дом “Вильямс”, 2009. 816 с.
21. Уоллс К. Spring в действии. 3-издание [Текст] / К. Уоллс - ДМК Пресс, 2013. 752 с.
22. Крейтон Р. Unity 3D Game Development by Example [Текст] / Р. Крейтон - Packt Publishing, 2010. 384 с.
23. Некоторые тонкости реализации RESTful приложений [Электронный ресурс]. – 2013 – URL: http://eax.me/rest/ (Дата обращения 24.08.2014).
24. AJAX JSON [Электронный ресурс]. – 2012 – URL: http://www.wisdomweb.ru/AJAX/json.php (Дата обращения 18.09.2014).
25. Коматинени С. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов [Текст] / С. Коматинени, Д. Маклин - Издательский дом “Вильямс”, 2012. 880.
26. Седжвик Р. Алгоритмы на Java [Текст] / Р. Седжвик, К. Уэйн - Издательский дом “Вильямс”, 2013. 848.
27. Кормен Т. Алгоритмы: построение и анализ Java [Текст] / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн - Издательский дом “Вильямс”, 2014. 1328.
28. Отношения классов от UML к коду – [Электронный ресурс]. – 2012 - URL: <http://habrahabr.ru/post/150041/> (Дата обращения: 22.02.2015).
29. Ларман К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование [Текст] / К. Ларман- Издательский дом “Вильямс”, 2012. 736.
30. Албахари Д. C# 5.0. Справочник. Полное описание языка [Текст] / Д. Албахари, Б. Албахари - Издательский дом “Вильямс”, 2013. 1008.
31. SADT, Нотация IDEF0 коду – [Электронный ресурс]. – 2012 - URL: http://habrahabr.ru/sandbox/31234/ (Дата обращения: 20.07.2014).
32. О принципах построения модели IDEF0 - [Электронный ресурс]. - 2015 - URL: http://www.intuit.ru/studies/courses/2195/55/lecture/1630 (Дата обращения: 5.05.2015).
33. Spring MVC — JavaConfig либо конфигурация проекта без XML файлов - [Электронный ресурс]. - 2014 - URL: http://habrahabr.ru/post/226663/ (Дата обращения: 12.08.2014).
34. Spring изнутри. Этапы инициализации контекста- [Электронный ресурс]. - 2014 - URL: http://habrahabr.ru/post/222579/ (Дата обращения: 15.08.2014).
35. Запись данных в формате JSON- [Электронный ресурс]. - 2014 - URL: http://habrahabr.ru/post/222579/ (Дата обращения: 23.08.2014).