СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проведение мероприятия – ответственное действие, к которому нужно тщательно подготовиться. И речь сейчас идет не о заучивании фраз и сценария, а о технической части. Нужно построить декорации, рассчитать количество требуемого инвентаря как для выступления, так и для гостей. Не зная, сколько придет людей, сложно указать как минимум требуемое количество стульев. Уместятся ли все пришедшие в выбранном помещении? Стоит ли арендовать большой зал, если придет только 10 человек?

Для подсчета количества участников используются стандартные средства – опрос с занесением на листик участников или опрос в виде гугл-форм. Эти методы имеют множество недостатков и требует уделение большого количество времени на себя.

Целью разработки данного дипломного проекта является создания информационной системы учета участников мероприятий, на котором будут собраны предложения всех мероприятий с описанием к ним с возможностью пользователям выбирать самим, на какие мероприятия они пойдут, а администратору в режиме реального времени видеть то, сколько участников собираются прийти к нему.

Основные задачи, которые должны быть выполнены в процессе создания дипломного проекта:

— охарактеризовать объект и существующие методы решения задачи;

— создать информационную модель системы;

— проектирование наглядного и удобного в использовании интерфейса;

— создать для пользователя все условия по работе с информацией;

— протестировать информационную систему;

— разработать руководство пользователю.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧ АВТОМАТИЗАЦИИ

1.1 Характеристики объекта и существующие методы решения задач объекта

Мероприятие – это действие людей, которые объединены, какой-либо общей целью. Это общее определение, у которого есть несколько нюансов. Например, группа подростков, собравшаяся выпить пива у подъезда, не может назвать свою встречу мероприятием. И это происходит не потому, что словарный запас этой прослойки общества порою скуден, а потому что это сборище не ставит своей целью культурное развитие [1].

Таким образом, можно сделать вывод, что мероприятие – это встреча группы людей, которые собрались культурно просветиться, узнать что-то новое или поведать другим свою точку зрения. А также мероприятием называется встреча людей в количестве большем, чем две персоны.

Мы поняли, как называется событие, когда небольшая группа людей собирается с целью просвещения или полезного времяпрепровождения. Теперь нужно понять, когда мероприятие можно называть публичным. Чаще всего такое определение можно встретить у демонстрации, митинга, театрального вечера, открытия выставки и т. д. Можно сделать вывод, что публичное мероприятие – это собрание большого количества незнакомых друг с другом людей, которые объединены общей целью.

Но, например, ярмарку, которая отлично подходит под определение, сложно назвать публичным мероприятием. Все люди имеют одинаковую цель купить продукты, но каждый это делает для себя.

Другой пример, театр. На представлении люди тоже получают удовольствие, причем каждый для себя. Но ведь это публичное мероприятие. Все

дело в том, что во втором примере актеры играют для всего зала, отдают свое время и дарят талант, чтобы зрители получили эстетическое наслаждение.

Утренники, творческие вечера или игры в детском парке - все это подходит под наше определение. Детское мероприятие – это деятельность взрослых, направленная на развлечение своих маленьких зрителей. Организованный детский праздник будет отличаться от игры одноклассников тем, что в первом случае есть видимый лидер, который задает правила. Аниматор диктует условия и смотрит за тем, чтобы они были выполнены.

Конечно, необязательно чтобы взрослые развлекали детей, они могут устроить мероприятия силами самих детей. Яркие примеры этого: утренники, школьные концерты и театральные постановки. Все это прививает маленьким членам общества командный дух, ответственность за свои поступки. Дети избавляются от комплексов, преодолевают скромность, учатся демонстрировать свои многочисленные таланты.

По сути, высказывание о том, что детские праздники отличаются от взрослых лишь возрастным контингентом, в корне не верно, в этом мы убедились с вами из вышеприведенных тезисов. Так что, как говорится, доверяй, но проверяй.

Часто возникает потребность знать, кто присутствовал на каком-либо мероприятии, или кто собирается присутствовать. Делается это по разным причинам. Как пример – банально нужно перед стартом мероприятия собрать достаточно стульев и выбрать подходящее помещение, чтобы все желающие поместились.

Если дело касается более ответственных мероприятий, как просветительская деятельность, активность клубов, принудительно-добровольные мероприятия – тут подсчет уже нужен для определения дальнейшей жизни человека, нужно ли ему идти на повторную встречу, достаточно ли он активен и т.д.

В настоящее время подсчет ведется ручным способом – записью на листочках. За некоторое время (недели, дни) до начала мероприятия

опрашивается каждый на желание пойти, и все согласившиеся вносятся в общий список.

1.2 Анализ существующих аналогов решения задач объекта

По запросу «система учета участников мероприятий», поисковик выдает только результаты о единой базе участников протестных акций, но никак о подсчете участников простых мероприятий. Если же искать информацию по запросу «система подсчета участников мероприятий», то в большей степени поисковик выдает результаты о применении искусственного интеллекта для подсчета участников напрямую на самом мероприятии в режиме реального времени, а не до начала мероприятия.

Как было сказано в предыдущем разделе, часто применяется ручной учет с использованием ручки и листика. Более продвинутый вариант такого учета – гугл-формы. Создается опрос на то, придет ли человек на мероприятие или нет, отправляется по ссылке всем знакомым и просят его пройти.

Проблемой такого подхода является то, что если человек постоянно участвует в мероприятиях, то он может запросто потеряться во всех этих одинаковых ссылках, забыть где-то отметиться.

1.3 Цели, требования и ожидаемые технико-экономические результаты создания информационной системы

Целью разработки данного дипломного проекта является создания информационной системы учета участников мероприятий, на котором будут собраны предложения всех мероприятий с описанием к ним с возможностью пользователям выбирать самим, на какие мероприятия они пойдут, а администратору в режиме реального времени видеть то, сколько участников собираются прийти к нему.

Требования к режиму гостя

— Просмотр всех мероприятий

— Просмотр подробностей мероприятия

— Возможность зарегистрироваться в системе

— Возможность войти в систему

Требования к режиму пользователя

— Наследование от возможностей гостя

— Просмотр своей истории мероприятий

— Подтверждение участия в мероприятии

— Отмена участия в мероприятии

Требования к режиму администратора

— Наследование от возможностей гостя

— Добавление категорий

— Изменение категорий

— Удаление категорий

— Добавление пользователей

— Изменение пользователей

— Удаление пользователей

— Добавление мероприятий

— Изменение мероприятий

— Удаление мероприятий

— Просмотр количества участников выбранного мероприятия

— Просмотр списка участников выбранного мероприятия

— Просмотр списка мероприятий любого пользователя

Проведем расчет себестоимости и цены научно-технической продукции.

Себестоимость продукции представляет собой выраженные в денежной форме затраты организации на ее производство и реализацию.

Расчет полной себестоимости единицы продукции осуществляется по следующим калькуляционным статьям затрат:

— материалы и комплектующие изделия;

— затраты на оплату труда:

а) основная заработная плата научно-технического персонала;

б) дополнительная заработная плата научно-технического персонала;

— отчисления в Фонд социальной защиты населения Республики Беларусь;

— отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний;

— прочие прямые расходы;

— накладные расходы.

Полная себестоимость разработки проекта рассчитывается по формуле 1:

$$С_п=МЗ +ЗОТ+ ОСН+БГС+ПР+ НР \quad (1)$$

где МЗ – материалы и комплектующие изделия (отсутствуют);

ЗОТ – затраты на оплату труда;

ОСН – отчисления в Фонд социальной защиты населения;

БГС – отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний;

ПР – прочие прямые расходы;

НР – накладные расходы.

При разработке программного обеспечения (а именно его мы и разрабатываем), элемент затрат на материалы и комплектующие отсутствует.

В статью «Затраты на оплату труда» включаются основная и дополнительная заработная плата всех работников, непосредственно занятых выполнением разработки.

Произведем расчет основной заработной платы по формуле 2:

$$З_о=\sum_{i=1}^{n} Т_{дi}\ Ф_п \cdot К_{пр}, \qquad (2)$$

где $Т_{дi}$ – дневная тарифная ставка i-го исполнителя (денежные единицы);

$Ф_п$ – плановый фонд рабочего времени (т.е время, в течение которого работники i-ой категории принимали участие в разработке), дней;

Кпр – коэффициент премирования за выполнение плановых показателей (1,2 – 1,4).

Дневная тарифная ставка i-го исполнителя рассчитывается следующим образом:

$Т_{дi} = Т_{мi} / Фр$,

где $Т_{мi}$ – месячная тарифная ставка i-го исполнителя, руб.;

$Ф_р$ – среднемесячный фонд рабочего времени, дни.

Расчетная норма рабочего времени в днях на 2021 г. при пятидневной рабочей неделе составляет 256 дней. Тогда среднемесячный фонд рабочего времени составляет:

Фр = 257/12≈21 дней

Месячная тарифная ставка каждого исполнителя ($Т_м$) определяется путем умножения действующей месячной тарифной ставки первого разряда ($Т_{м\ 1р}$) на тарифный коэффициент ($Т_к$), соответствующий установленному тарифному разряду (выбрать из единой тарифной сетки для определенной категории работников, которые будут работать над проектом):

$$Т_м= Т_{м\ 1р} \cdot Т_к \qquad (3)$$

$Т_м= 1,4 * 195 = 273$

Расчеты можно представить в виде таблицы 1.1

Таблица 1.1 - Расчет основной заработной платы научно-производственного персонала

Категория работников	Кол-во, чел	тарифный коэф-т	Месячная тарифная ставка соотв. разряда	Дневная тарифная ставка	Плановый фонд рабочего времени	Оплата за отработанное время одного работника	Основная заработная плата одного работника с учетом премии	Основная заработная плата с учетом премии, всего
1	2	3	4	5	6	7	8	9
Программист-руководитель проекта	1	1,4	273	13	50	650	910	910
Итого								910

Дополнительная заработная плата научно-технического персонала включает выплаты, предусмотренные действующим законодательством за непроработанное время (оплата очередных и дополнительных отпусков, выплата за выслугу лет и др.) определяется по формуле:

Дополнительная заработная плата рассчитывается по формуле 4:

$$З_д = З_о \cdot Н_д / 100\%, \quad (4)$$

$$З_д = 910*20/100 = 182$$

где $Н_д$ – норматив дополнительной заработной платы (20%).

В целом затраты на оплату труда составят:

$$ЗОТ = З_о + З_д.$$

$$ЗОТ = 910 + 182 = 1092$$

– Отчисления в фонд социальной защиты населения на социальные нужды. Размер затрат определяется в процентах от суммы основной и дополнительной заработной платы всех категорий работников, причастных к выполнению данной разработки:

$$\text{ОСН} = \frac{\text{ЗОТ} * S_{\text{осн}}}{100},$$

$S_{\text{осн}}$ – ставка отчислений на социальные нужды, % (ставка – 34%)

$$\text{ОСН} = \frac{1092 * 34}{100} = 371{,}28$$

– Отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний рассчитывается по следующей формуле:

$$\text{БГС} = \frac{\text{ЗОТ} * \text{Н}_{\text{БГС}}}{100},$$

Нбгс – норматив отчислений в Белгосстрах установленный для организации, %

$$\text{БГС} = \frac{1092 * 6}{100} = 65{,}52$$

Все остальные затраты включаются в себестоимость единицы продукции косвенным путем в процентах от основной заработной платы.

– прочие прямые расходы. Данная статья затрат включает в себя затраты на приобретение, перевод специальной научно-технической информации, использование технических средств связи и т.д.

$$ПР = З_о * Н_{пр} /100,$$

где $Н_{пр}$ – норматив прочих расходов для расчета себестоимости и отпускной цены НИОКР, %

$$ПР = 910*10 /100 = 91$$

– накладные расходы. Данная статья затрат в равной степени относится ко всем выполняемым НИОКР и включает в себя затраты на содержание и текущий ремонт зданий, сооружений, оборудования, инвентаря, расходы по охране труда и т.д.

$$НР = З_о * Н_{НР} /100,$$

где $Н_{пр}$ – норматив накладных расходов для расчета себестоимости и отпускной цены НИОКР, %

$$НР = 910*150/100 = 1365$$

Таблица 1.2 – Затраты на приложение

Наименование статьи затрат	Условнбое обозначение	Значение, руб.	Примечание
Материалы и комплектующие изделия	МЗ	0	При разработке ПО отсутствуют
Затраты на оплату труда Научно-производственного персонала, всего В т.ч. - основная заработная плата - дополнительная заработная плата	ЗОТ Зо Зд	1092 910 182	$ЗОТ = З_о + З_д$ $З_д = З_о \cdot Н_д/100\%$
Отчисления в фонд социальной защиты населения	ОСН	371,28	$ОСН = \frac{ЗОТ * S_{осн}}{100}$
Отчисления в Белгосстрах	БГС	65,52	$БГС = \frac{ЗОТ * Н_{БГС}}{100}$
Командировочные расходы	КР	0	Отсутствуют
Услуги сторонних организаций	УСО	0	Отсутствуют
Прочие прямые расходы	ПР	91	$ПР = З_о * Н_{пр} /100$
Накладные расходы	НР	165	$НР = З_о * Н_{НР} /100$
Полная себестоимость	**Сп**	**1784,8**	**Сп=МЗ+ЗОТ+ОСН+БГС+ПР+НР +КР+УСО**
Налог на добавленную стоимость	НДС	356,96	НДС = Сп*Sндс/100, Где Sндс – ставка налога на добавленную стоимость, % (20)
Отпускная цена НИОКР	**Цотп**	**2141,76**	**Цотп = Сп+НДС**

В ходе расчетов было получено, что полная себестоимость проекта сети составляет 1784,8 руб., прогнозируемая цена проекта с учетом необходимых отчислений и налогов – 2141,76 руб.

Данная сумма затрат оправдана, т.к. разработка позволит вести точный учет количества участников мероприятий, получая не только общее количество участников для какого-либо отдельного мероприятия, но также и подробный список пользователей и таким образом увеличить доходность, за счет возможности быть более ориентированным как по количеству участников (что позволит точно рассчитывать сумму на арендуемые помещения), как и быть ориентированным на отдельных пользователей.

2 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

Под информационной системой (ИС) понимается прикладная подсистема в программной реализации, которая ориентирована на сбор, систематизацию, хранение, поиск и обработку информации требуемого типа.

Информационные системы предназначаются для автоматизации процессов, подчиняются конкретной логике обработки информации, предполагают ее хранение, а также представление в виде отчетов. При помощи разработки информационной системы реально производить интеграцию информационных ресурсов предприятий, применять их для более эффективного управления [2].

Основная цель информационных систем — эффективное внедрение всех ресурсов, увеличение значения управленческих принимаемых решений, ориентированных на повышение доходов.

Процесс разработки ИС:

Определение требований. Разработка любой системы начинается с постановки задачи. ИС, как правило, создается для большого количества пользователей. Каждый из них предъявляет собственные требования к системе. На этом этапе необходимо выявить всех потенциальных пользователей ИС, и для каждого из них составить список требований к ней. Так будут сформулированы основные функциональные требования к системе [3].

Этап анализа. Аналитическая модель структурирует функциональные требования к системе. Она описывает уже внутренний вид системы, используя язык разработчиков. Она представляет собой анализ каждого варианта использования и определяет его дальнейшую реализацию.

Этап проектирования. Это самый трудоемкий этап разработки информационной системы. На данном этапе необходимо разработать проекционную модель всей системы в целом и каждого из ее блоков. Для каждой задачи, которая будет реализована в рамках системы, необходимо описать возможные методы ее решения. Эти методы следует сравнить между собой по

критериям, значимым с точки зрения системы, на основании чего выбрать лучший из них. Именно этот метод должен быть реализован впоследствии в программе. Также на этом этапе происходит проектирование базы данных. Сложные информационные системы, как правило, структурированы, т.е. представляют собой совокупность нескольких функциональных блоков. На этапе проектирования должна быть строго описана функциональность каждого из блоков. Здесь же обосновывается выбор методов интеграции блоков в единый информационный комплекс.

Этап реализации. На этапе реализации происходит непосредственно написание программы на выбранном языке программирования. В техническом задании должен быть обоснован выбор именно этого языка, а также выбор СУБД и иных программных средств.

Этап тестирования. На этапе тестирования необходимо проверить корректность функционирования системы в нормальных условиях функционирования (когда в систему вводятся корректные исходные данные), в граничных условиях (когда на вход подаются допустимые, но редко используемые параметры или граничные параметры) и в экстремальных условиях (когда на вход системы подаются некорректные данные). Модель тестирования должна описывать результаты, которые были получены при обработке всех этих данных.

Этап внедрения и сопровождения. На этом этапе происходит обеспечение стабильной работы и снижение рисков возникновения сбоев в работе информационных систем; оперативное исправление технических неполадок в работе систем; предоставление новых версий, обновлений и дополнений, консультации по вопросам эксплуатации и администрирования информационных систем; консультации по установке и настройке новых версий, обновлений, дополнений и т.д.

Оценка эффективности ИС. На этом этапе собираются отзывы у клиента о процессе использования информационной системы и выявляются требования по улучшению ее работы.

2.1 Модель функционирования объекта

Включают широкий спектр символических моделей [4]:

— модель жизненного цикла систем, описывающую процессы существования системы от зарождения замысла до прекращения функционирования;

— модели операций, выполняемых объектом и представляющих описание взаимосвязанной совокупности процессов функционирования отдельных элементов объекта при реализации тех или иных функций объекта (в их состав могут входить модели надежности, характеризующие выход элементов системы из строя под влиянием эксплуатационных факторов, и модели живучести, характеризующие выход элементов системы из строя под влиянием целенаправленного воздействия внешней среды);

— информационные модели, отображающие во взаимосвязи источники и потребители информации, виды информации, характер ее преобразования, а также временные и количественные характеристики данных;

— процедурные модели, описывающие порядок взаимодействия элементов исследуемого объекта при выполнении различных операций (например, обработка материалов, деятельность персонала, использование информации, в том числе и различные процедуры принятия управленческих решений);

— временные модели, описывающие процедуру функционирования объекта во времени распределения ресурса «время» по отдельным компонентам объекта.

Модель обычно является комбинацией физических, математических и гибридных элементов либо целиком реализована в виде компьютерной имитации.

На практике для прогнозирования ожидаемого поведения сложной системы в целом обычно нецелесообразно строить модель функционирования, способную отразить все детали поведения системы. Поэтому при испытаниях системы в целом ее наблюдаемое поведение обычно анализируется на двух уровнях. Первый уровень относится к сквозным показателям функционирования, определенным в требованиях к системе. Второй - к тем подсистемам и компонентам, которые характеризуются каким-то критическим поведением. Последний уровень особенно важен, когда сквозное испытание не дает ожидаемого результата и требуется найти источник отклонений.

Решение о том, насколько детальной должна быть модель, применяемая для испытаний на уровне системы, является главным образом функцией системной инженерии, поскольку требуется искать компромисс между риском, сопутствующим игнорированию включения в модель некоторых функций, и ценой их включения. Поскольку испытать все невозможно, при выборе того, что подвергать испытаниям в первую очередь (а значит, и того, что должна прогнозировать модель), необходимо принимать во внимание результаты анализа относительных рисков игнорирования некоторых характеристик.

Проектирование, построение и валидация моделей функционирования системы - сама по себе трудная задача, при решении которой должны применяться те же системно-инженерные методы, что и при разработке самой системы. Но в то же время нужно стремиться ограничить затраты на моделирование экономически целесообразной долей от всех затрат на разработку системы. Соблюдение баланса между реалистичностью и стоимостью моделирования - одна из самых трудных задач системной инженерии.

Методология *IDEF*0 – это одна из самых известных методологий функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов [5].

Отличительной особенностью *IDEF*0 является ее акцент на соподчиненность объектов. В *IDEF*0 рассматриваются логические отношения между работами, а не их временная последовательность.

Стандарт *IDEF*0 представляет организацию как набор модулей. Здесь существует правило – наиболее важная функция находится в верхнем левом углу, кроме того, есть правило стороны:

– стрелка входа всегда приходит в левую кромку активности;

– стрелка управления – в верхнюю кромку;

– стрелка механизма – нижняя кромка;

– стрелка выхода – правая кромка.

Описание выглядит как «черный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня.

Также для того, чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях дается описание того, какой смысл вкладывается в данную активность либо стрелку.

Как можно понять из определения, методология описывает бизнес-процессы, их иерархичность и взаимодействие в рамках определенной системы.

Входной информацией являются: незарегистрированный гость, желание прийти на мероприятие и информация о проведении нового мероприятия.

Механизмами являются нормативно-правовые документы и требования к безопасности.

Управляющие механизмы – Пользователь, администратор и ИС (информационная система). Именно благодаря администратору идет заполнение информации о мероприятиях и просмотр участников. Пользователь выбирает для себя мероприятия, а ИС позволяет регистрироваться пользователей, мероприятия и вести подсчет участников.

На рисунке 2.1 показана диаграмма деятельности информационной системы учета участников мероприятий.

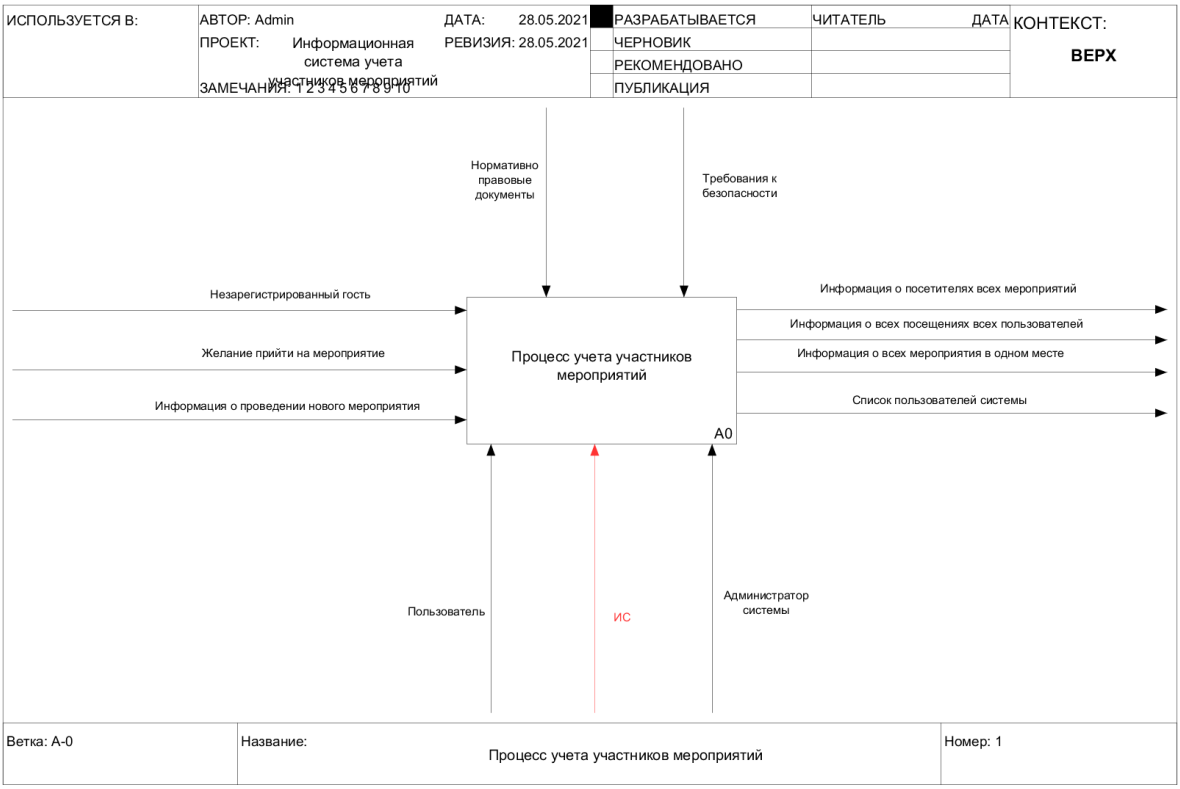

Рисунок 2.1 – Бизнес-процесс учета участников мероприятий

Декомпозиция контекстной диаграммы представлена на рисунке 2.2. Дна состоит из следующих логических частей: Обновлении информации о мероприятиях, регистрации пользователя, выбор мероприятий для посещения и просмотр участников мероприятий.

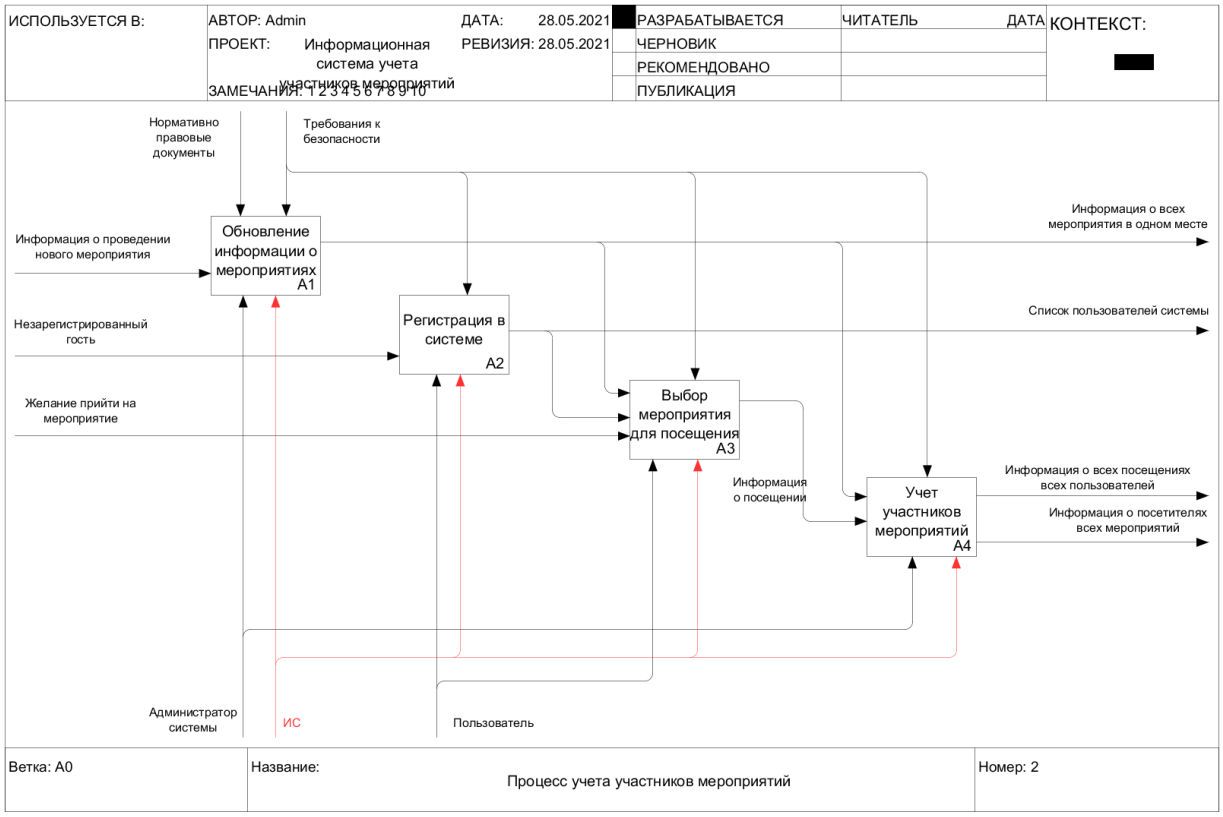

Рисунок 2.2 – Декомпозиция контекстной диаграммы

На рисунках 2.3-2.6 отобразим результат декомпозиции каждого процесса.

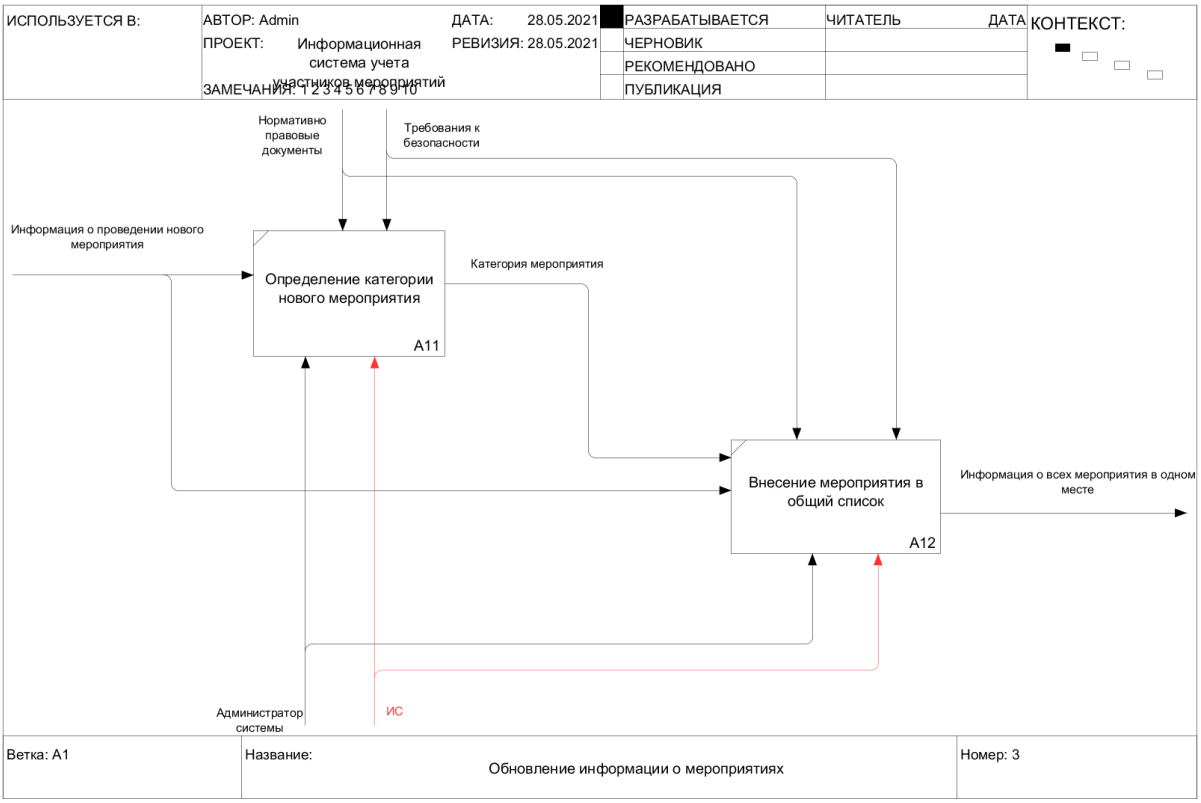

Рисунок 2.3 – Декомпозиция процесса «Обновление информации о мероприятии»

На данном этапе администратор системы вносит информацию о новом мероприятии в систему.

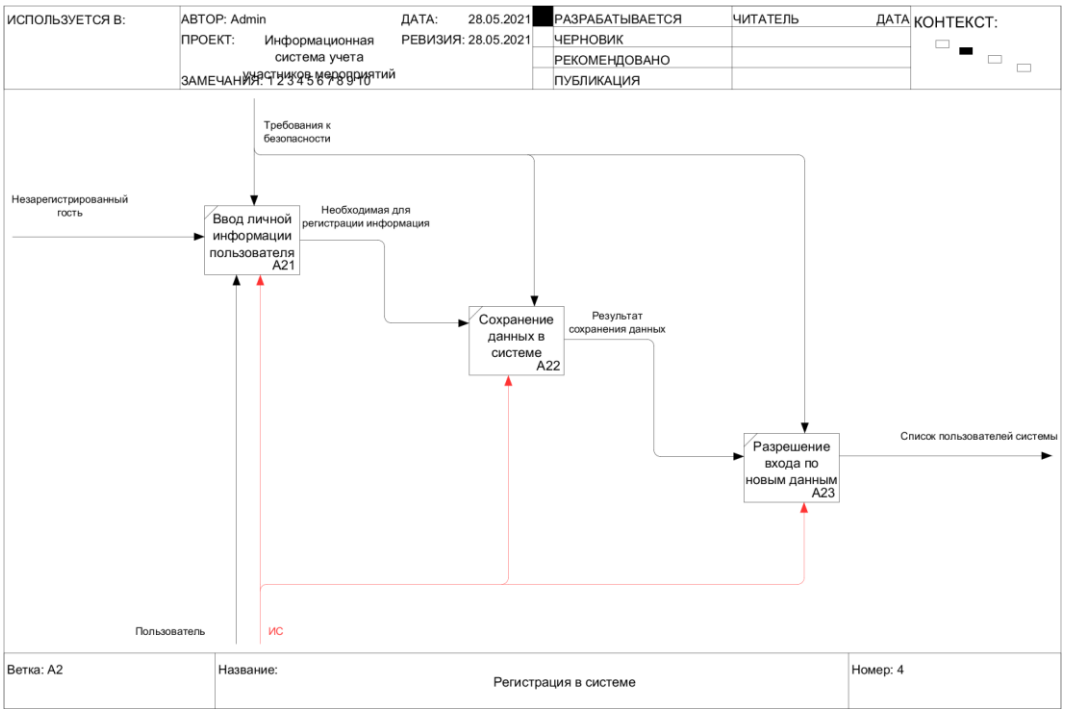

Рисунок 2.4 – Декомпозиция процесса «Регистрация в системе»

Данный процесс позволяет всем гостям (незарегистрированным пользователям) зарегистрироваться в системе для дальнейшего участия в мероприятиях. В данном случае под участием подразумевается соглашение на участие в мероприятии и добавление его к себе в список мероприятий.

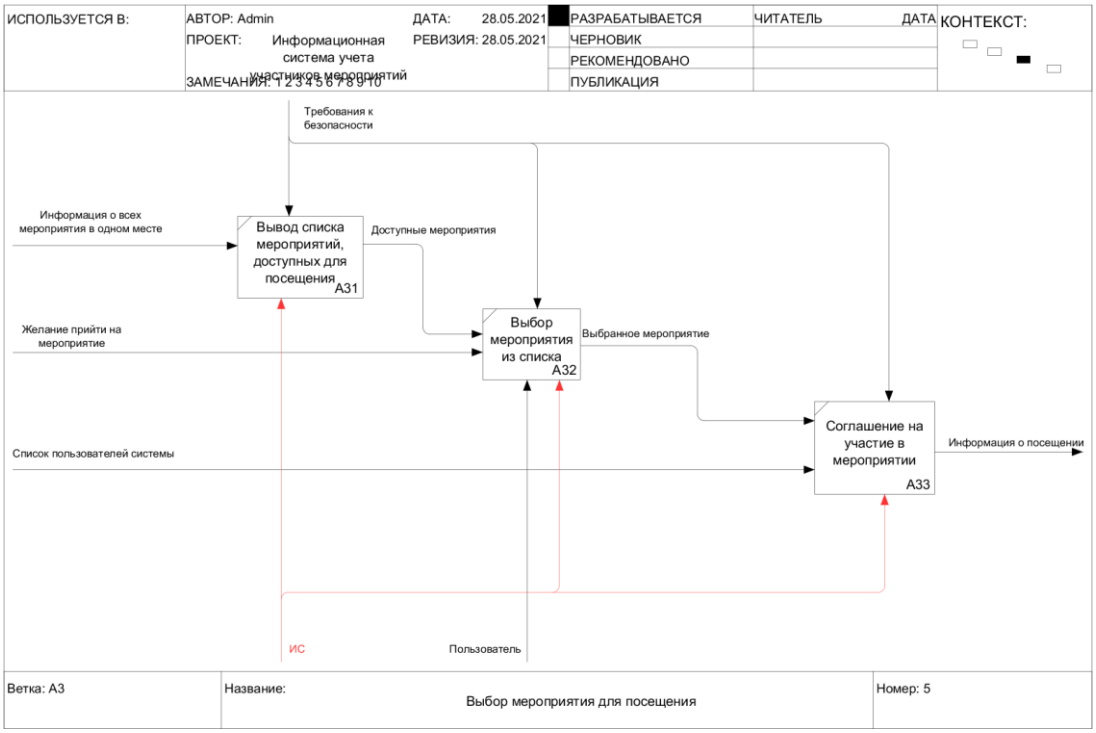

Рисунок 2.5 – Декомпозиция процесса «Выбор мероприятия для посещения»

Данный процесс представляет собой процесс выбора пользователем мероприятия для посещения и соглашение на участие в нем.

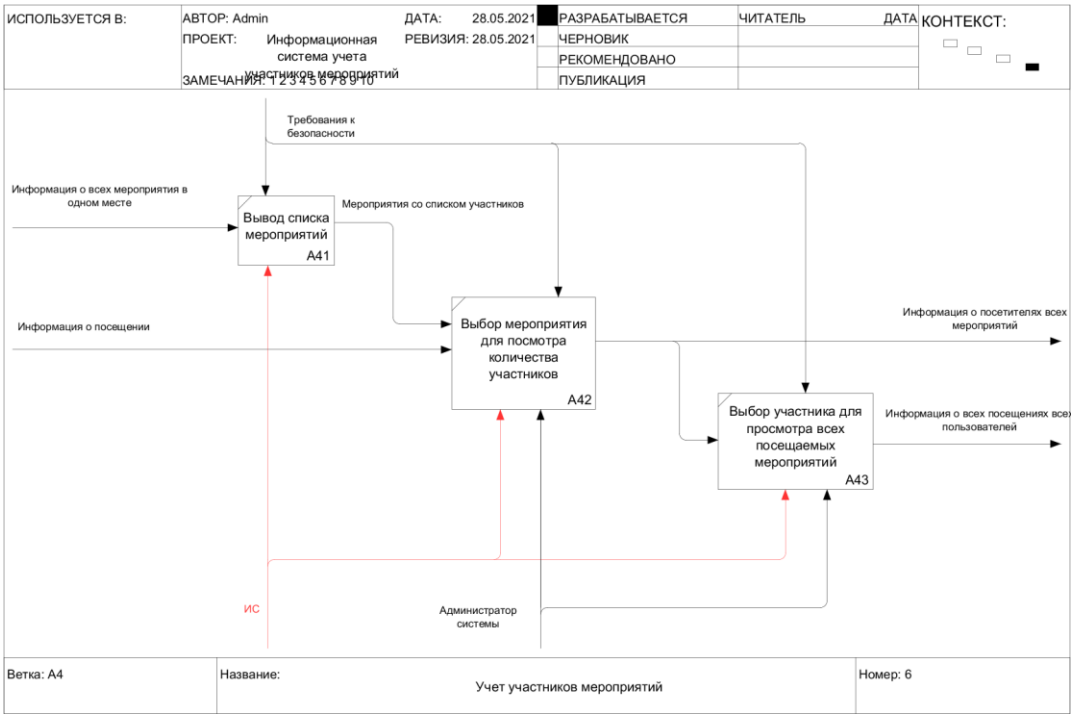

Рисунок 2.6 – Декомпозиция процесса «учет участников мероприятий»

Данный процесс позволяет администратору системы выбирать мероприятие из списка и просматривать список всех его участников, а также просматривать список всех посещений мероприятий отдельно выбранного участника.

2.2 Информационное обеспечения

Информационное обеспечение — совокупность системы классификации и кодирования, системы показателей, языков записи данных, унифицированных систем документации и массивов информации, используемых в автоматизированных системах управления [6].

Информационное обеспечение включает:

— состав информации, т. е. перечень информационных единиц или информационных совокупностей (показателей, констант, переменных, документов, других сообщений, необходимых для решения комплекса задач системы);

— структуру информации и закономерности ее преобразования, т. е. правила построения показателей, документов, агрегации и декомпозиции информационных единиц, преобразования информационных единиц в цепочке «вход — система — выход»;

— характеристики движения информации, т. е. количественные оценки потоков информации (объем, интенсивность), определение маршрутов движения документов, построение схем документооборота, временные характеристики функционирования источников информации, получения первичных данных, использования исходных данных, продолжительности хранения, старения и обновления данных.

Иными словами, в этом подразделе следует разобрать, какая информация нужна для построения системы, как с ней будет взаимодействовать пользователь и как она будет храниться и обрабатываться.

В данном подразделе будет представлено описание информационной модели данных и ее структура, которая обеспечивает работу системы. При этом данное описание будет, как в текстовом формате, так и при помощи диаграммы в нотации *IDEF*1*X* в *MYSQL Workbench*.

*IDEF*1*X* является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой называется универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы [7]. Логическая модель данных является визуальным представлением структур данных, их атрибутов и бизнес-правил. Логическая модель представляет данные таким образом, чтобы они легко воспринимались бизнес-пользователями.

Информационная модель в нотации *IDEF*1*X* представлена на рисунке 2.7.

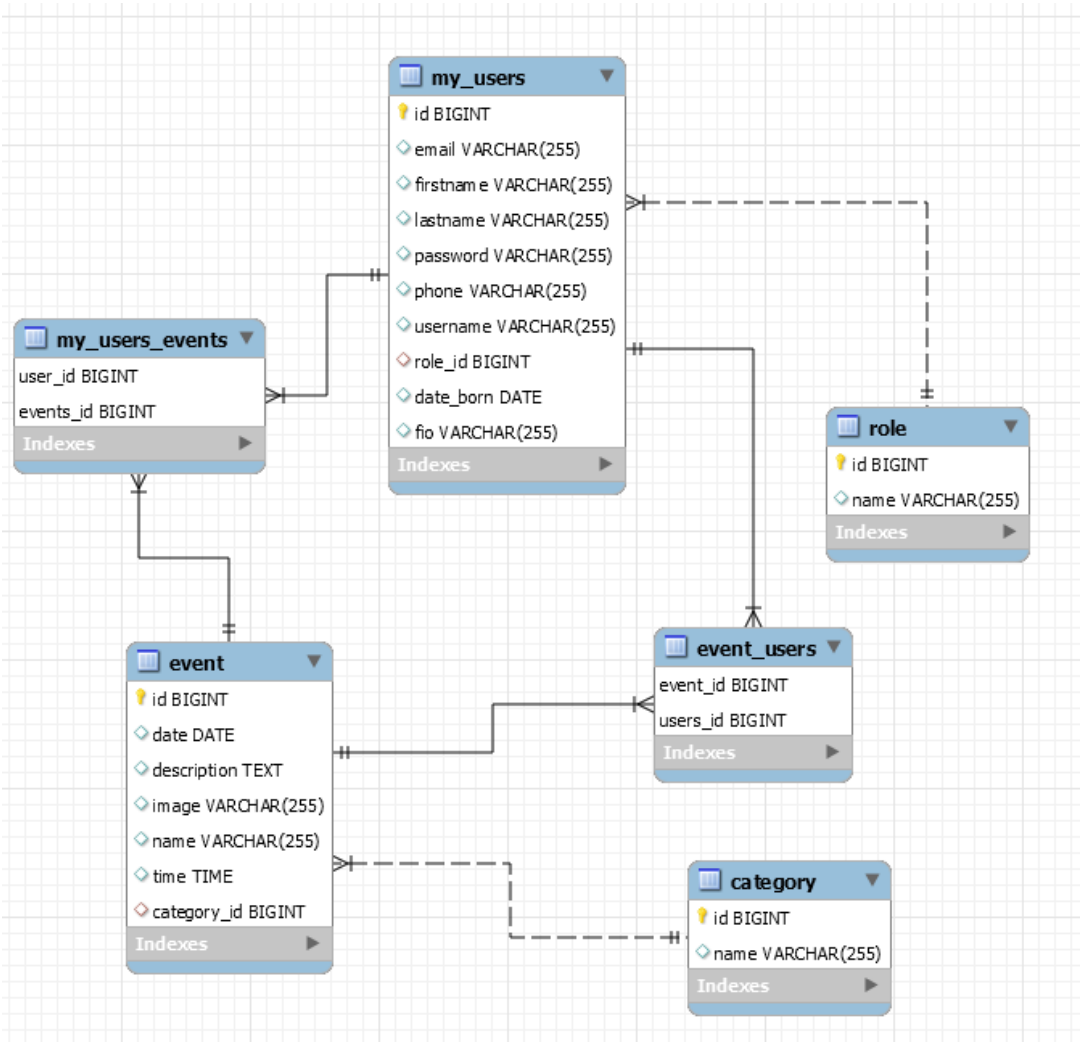

Рисунок 2.7 – Информационная модель

В данной информационной модели присутствует шесть сущностей. Перечень сущностей:

1. Роль. Хранит список ролей системы.
2. Пользователь. Хранит информацию о всех пользователях системы.
3. Категория. Хранит информацию о категориях мероприятий.
4. Мероприятие. Хранит информацию о проводимых мероприятиях.
5. Пользователи событий. Хранит информацию о всех пользователях события.
6. События пользователей. Хранит информацию о всех событиях пользователя

Спецификация программного средства содержит описание или ссылки на описания исполняемого программного обеспечения, исходных файлов и информацию о программной реализации, включая информацию проекта построения, компиляции, построения и процедуры модификации.

Описание спецификации вариантов использования будет произведено в графическом варианте при помощи диаграммы *UML* 2.0 – диаграммы вариантов использования [8].

Для данного проекта диаграмма вариантов использования представлена на рисунке 2.8.

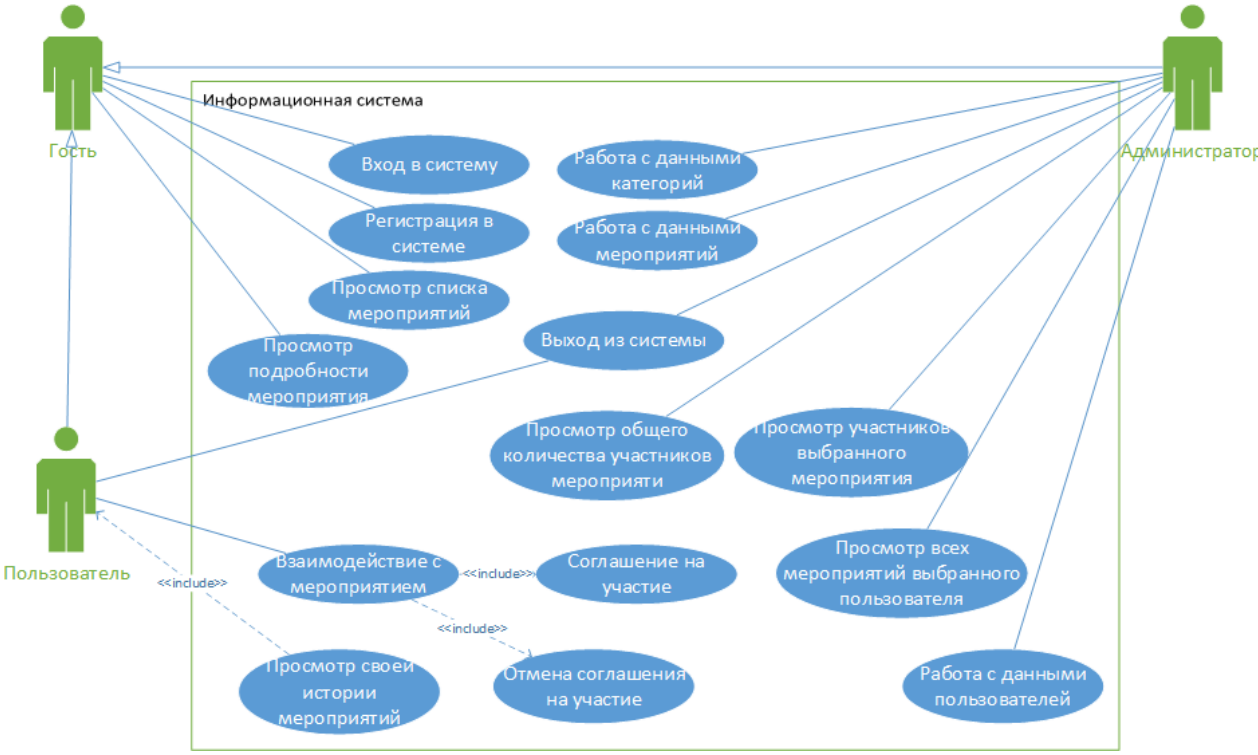

Рисунок 2.8– Диаграмма вариантов использования

Данная диаграмма демонстрирует все варианты использования системы всеми основными актёрами. Всего есть 3 актера: гость, пользователь и администратор.

Гость может просматривать общий список мероприятий и подробности о каждом из них. Также может зарегистрироваться в системе и войти в нее. Как только гость вошел в систему, он становится пользователем.

Пользователь уже имеет больше прав: он может согласиться или отказаться от участия в мероприятии. А также просматривать свою историю посещения.

Администратор имеет больше всего прав в системе. Он может напрямую работать с данными из таблиц категорий, пользователей и мероприятий. Кроме того, при просмотре информации о мероприятии, ему доступен просмотр общего количества участников мероприятия и предоставлен список всех участников. ПО желанию, администратор может перейти на страницу истории любого пользователя и посмотреть, в каких мероприятиях пользователь участвовал, и в каких собирается участвовать.

2.3 Программное обеспечение

Программное обеспечение напрямую зависит от стека технологий и архитектуры разрабатываемого приложения. Архитектура данного проекта – это веб-приложение. Архитектура приложения во многом предопределило и технологии, используемые для построения проекта.

Основные технологии, применяемые при создании приложения:

— серверный язык *JAVA*;

— язык запросов *SQL* (*MySQL*);

— фреймворк *Spring*;

— язык разметки *HTML*;

— таблица стилей *CSS*;

— клиентский язык *JavaScript*.

Уже на этом этапе можно определить необходимое программное обеспечение для разработки:

— IntelliJ IDEA;

— MySQL Server;

— Workbench;

— Visio Professional.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Язык Java появился в 1995 году – 90-е годы были вообще урожайными на новые языки и концепции программирования. В таком Эдеме языков важно было не заблудиться, по ошибке приняв за Священный Грааль технологию, которая не пройдет испытания временем. Java прошел испытания, хотя и очень долгие. Очень не рекомендуется путать этот язык с JavaScript – они по виду похожи, но это совсем разные языки [9].

Вероятно, в Java впервые реализовали концепцию того, что язык должен быть максимально изолирован от платформы разработки, чтобы применять его без изменений везде: в компьютерах, часах, сотовых телефонах, бытовой технике. С «железной частью» должна была справляться виртуальная машина (JVM), которая, собственно, и создавалась индивидуально под каждое устройство. Сам же язык был неизменен и в качестве результата выдавал байт-код.

С самого начала было известно, что код не может исполняться очень быстро, но многие устройства не требовали высокой скорости исполнения. Кроме того, со временем появились оптимизирующие компиляторы, так что, в среднем, программа на Java работает раза в 2-3 медленнее, чем на C++.

Постоянное сравнение с C/C++ здесь не случайно: многие современные языки взяли за основу его конструкции и синтаксис, так что, бывает, узнать сходу язык очень трудно. Вместе с тем, Java с тех пор сильно «размножилась», и даже J#, J и прочие аналоги являются не родными братьями, а лишь подобием.

Сама идея языка, вполне, кстати, достаточного для создания софта любой сложности, была сначала не понята: был ли, мол, смысл создавать между аппаратурой и кодом промежуточные слои исполняющих машин? Со временем сомнения рассеялись: появилась мультиязычная платфома .

NET, и даже в Windows появились слои – аппаратно-зависимые, платформо-независимые. Самое же простое объяснение – софт стал очень

сложным, а программисты очень ленивыми, чтобы переписывать программы под каждый отдельный аппарат.

Но вернемся к языку. Как уже говорилось, чем-то он похож на C++, чем-то на старый добрый Бейсик. Нет сейчас ни одного языка, который бы не хвалился своими возможностями ООП, и Java здесь не отличается от канонов: классы и объекты здесь используются везде, даже в самых примитивных задачах вроде вывода строки на экран. Из особенностей можно отметить, что все объекты в языке создаются только динамически, а все функции являются методами классов.

Множественное наследование не поддерживается, как в C++, как и «опасные» указатели. ООП дает много преимуществ, но и требует слишком многого – в случае Java памяти устройства никогда не будет слишком много.

В остальном же, имеются библиотеки классов для практически всех задач; преимущественно – под написание клиентских и серверных приложений. Хозяин Java – Oracle – успешно использует язык для использования в разработках своей одноименной СУБД. На сегодняшний день язык считается наиболее востребованным на рынке.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET [10].

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена 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 — 5.2.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring

предоставляет бóльшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Следующие две технологии – это *HTML* и *CSS*, которые предопределены архитектурой проекта. Имеется единственное обоснование их выбора – это веб-архитектура разрабатываемого программного средства.

HTML – язык гипертекстовой разметки, который используется для структурирования и отображения веб-страницы и ее контента. Для разработки данного программного средства будет применяться версия этого языка *HTML*5 [11].

CSS – это формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки *HTML*. Применяемая версия данной технологии – *CSS*3 [12].

Последняя из основных применяемый технологий – это *JavaScript*. С помощью данного языка программирования, можно создавать динамически обновляемый контент, управлять мультимедиа, анимировать изображения. Что касается обоснований выбора в качестве клиентского языка *JavaScript*, то здесь выбор достаточно очевиден, ведь конкурентов у данного языка в данной сфере практически нет [13].

Лишь некоторые достоинства данного языка: возможность навигации и управления по *DOM HTML*-страницы, возможность управления браузером,

гибкость подхода объектно-ориентированного программирования, поддержка асинхронности.

Для проектирования программного средства используется язык графического описания *UML*.

UML – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML*-моделью. *UML* был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. *UML* не является языком программирования, на основании *UML*-моделей возможна генерация кода.

Microsoft Office Visio Professional 2016 – это профессиональное решение для создания технических и деловых диаграмм, предназначенных для систематизации и наглядного представления различных данных, процессов и систем [14]. Диаграммы Microsoft Office Visio Professional позволяют без труда осуществлять визуализацию и обмен различной информацией с высочайшей точностью, надежностью и эффективностью, недостижимыми при использовании текстовых и числовых данных. Microsoft Office Visio Professional содержит все возможности стандартного выпуска, а также расширенный набор шаблонов и функций, включая визуализацию данных и подключение к данным.

Версия Microsoft Office Visio 2016 предоставляет новые инструменты в помощь для интуитивного создания диаграмм, включая новые и усовершенствованные формы и трафареты, улучшенные эффекты и темы, а также функции соавторства, упрощающие совместную работу. Пользователи Visio 2016 могут более динамично выстраивать диаграммы за счет привязки форм к данным реального времени, а затем делиться результатами со своими коллегами через web-браузер с помощью служб Visio Services в SharePoint, даже

если у этих пользователей не установлено ПО Microsoft Office Visio 2016. Версия Microsoft Office Visio 2016 поддерживает только операционные системы Windows 8 и 7.

MySQL — это быстрая и простая в использовании СУБД, используемая для многих малых и крупных предприятий [15]. MySQL разрабатывается, продается и поддерживается MySQL AB, шведской компанией. MySQL становится настолько популярным по многим веским причинам —

MySQL выпущен под лицензией с открытым исходным кодом. Таким образом, вам нечего платить, чтобы использовать его.

MySQL — очень мощная программа сама по себе. Он обрабатывает большой набор функций самых дорогих и мощных пакетов баз данных.

MySQL использует стандартную форму известного языка данных SQL.

MySQL работает во многих операционных системах и со многими языками, включая PHP, PERL, C, C ++, JAVA и т. Д.

MySQL работает очень быстро и хорошо работает даже с большими наборами данных.

MySQL очень дружелюбен к PHP, самый ценный язык для веб-разработки.

MySQL поддерживает большие базы данных, до 50 миллионов или более строк в таблице. Предельный размер файла по умолчанию для таблицы составляет 4 ГБ, но вы можете увеличить его (если ваша операционная система может его обработать) до теоретического предела 8 миллионов терабайт (ТБ).

MySQL настраивается. Лицензия GPL с открытым исходным кодом позволяет программистам модифицировать программное обеспечение MySQL в соответствии со своими специфическими средами.

MySQL выпущен под лицензией с открытым исходным кодом. Таким образом, вам нечего платить, чтобы использовать его.

MySQL Workbench — это унифицированный визуальный инструмент для архитекторов баз данных и разработчиков БД. MySQL Workbench предоставляет возможность моделирование данных, разработку SQL и комплексные инструменты администрирования для конфигурации сервера,

администрирования пользователей, резервного копирования и многое другое. MySQL Workbench доступен на Windows, Linux и Mac OS X.

MySQL Workbench предоставляет визуальные инструменты для создания, выполнения и оптимизации SQL-запросов. Редактор SQL предоставляет цветовую подсветку синтаксиса, автозаполнение, повторное использование фрагментов SQL и историю выполнения SQL. Панель подключения к базе данных позволяет разработчикам легко управлять стандартными подключениями к базе данных, включая MySQL Fabric. Обозреватель объектов обеспечивает мгновенный доступ к схеме базы данных и объектам.

Сегодня об IntelliJ IDEA знает, без преувеличения, весь мир. Хотя платформа ориентирована, в первую очередь, на Java-кодинг, ее универсальность позволяет работать с разными языками. Конечно, здесь нет такого массированного набора плагинов, как у любого open-source конкурента, но зато в их качестве можно быть относительно уверенным. Если сравнивать эту среду с операционными системами, то IntelliJ будет сродни Mac OS — закрытая, слегка консервативная, но надежности не занимать. Да и нет больших сложностей в работе. Как только вы захотите создать новый проект, интуитивное меню подскажет, как это сделать [16].

Но лишь сегодня у IntelliJ IDEA красивый и интуитивный интерфейс, который позволяет разобраться даже новичку. Несколько лет назад, когда команда только стартовала, они разрабатывали свою среду разработки на основе тех же продуктов с открытым кодом. Сначала это даже не должно было быть полноценной IDE. Вместо этого компания создавала инструменты, которые должны были взаимодействовать со средой, обладающей популярностью в те годы — JBuilder. Но с учетом того, что последняя была далека от совершенства, первая IntelliJ IDEA увидела мир уже через год после начала работы стартапа.

В первую очередь, команда сконцентрировалась на создании средств для рефакторинга и анализа. Как раз из этого родилась современная IDEA. Развиваясь в этом направлении, создатели смогли разработать продукт, в котором код пишется частично вами, а частично машинным интеллектом. А все

начиналось с того, чтобы обеспечить вовсе утилитарные функции: комфортное переименование классов, методов, автоматическое определение и прочие.

Фреймворк Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними [17].

– Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).

– View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

– Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Вся логика работы Spring MVC построена вокруг DispatcherServlet, который принимает и обрабатывает все HTTP-запросы (из UI) и ответы на них. Рабочий процесс обработки запроса DispatcherServlet'ом проиллюстрирован на следующей диаграмме:

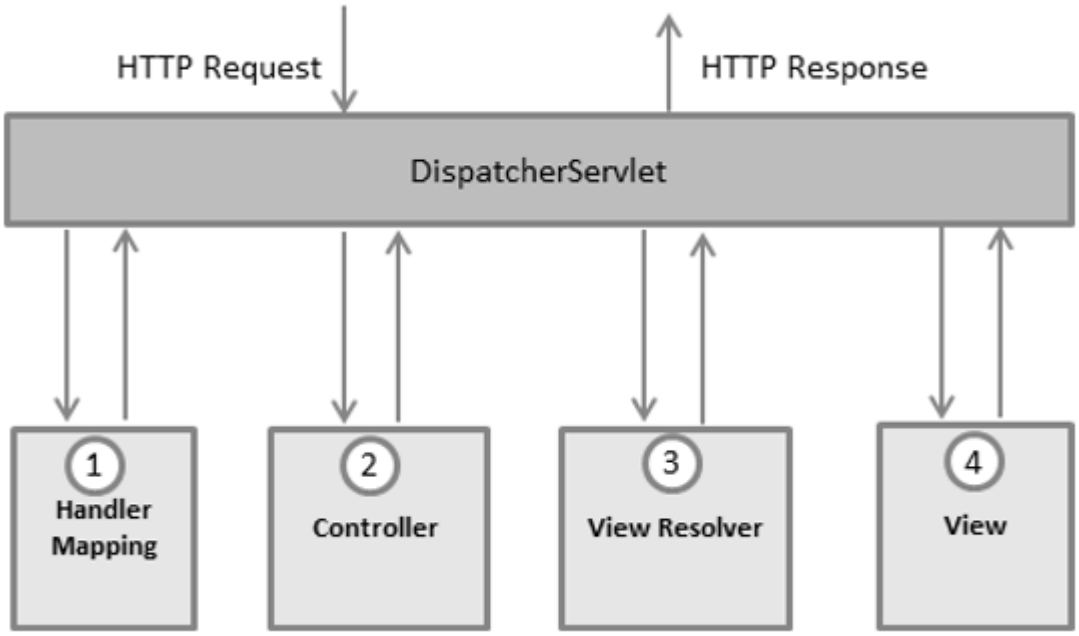

Рисунок 2.9 – Рабочий процесс обработки запроса

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

− После получения HTTP-запроса DispatcherServlet обращается к интерфейсу HandlerMapping, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.

− Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в DispatcherServlet имя Вида (View).

− При помощи интерфейса ViewResolver DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени.

− После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

Все вышеупомянутые компоненты, а именно, HandlerMapping, Controller и ViewResolver, являются частями интерфейса WebApplicationContext extends ApplicationContext, с некоторыми дополнительными особенностями, необходимыми для создания web-приложений.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций. Аннотация @Controllerannotation указывает, что конкретный класс является контроллером. Аннотация @RequestMapping используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика.

Аннотация Controller определяет класс как Контроллер Spring MVC. В первом случае, @RequestMapping указывает, что все методы в данном Контроллере относятся к URL-адресу "/hello".

Следующая аннотация @RequestMapping(method = RequestMethod.GET) используется для объявления метода printHello() как дефолтного метода для

обработки HTTP-запросов GET. Вы можете определить любой другой метод как обработчик всех POST-запросов по данному URL-адресу.

Вы можете написать вышеуказанный Контроллер по-другому, указав дополнительные атрибуты для аннотации @RequestMapping следующим образом

@RequestMapping(value = "/hello", method = RequestMethod.GET) Атрибут «value» указывает URL, с которым мы связываем данный метод (value = "/hello"), далее указывается, что этот метод будет обрабатывать GET-запросы (method = RequestMethod.GET). Также, нужно отметить важные моменты в отношении приведённого выше контроллера:

– Вы определяете бизнес-логику внутри связанного таким образом служебного метода. Из него Вы можете вызывать любые другие методы.

– Основываясь на заданной бизнес-логике, в рамках этого метода Вы создаёте Модель (Model). Вы можете добавлять аттрибуты Модели, которые будут добавлены в Вид (View). В примере выше мы создаём Модель с атрибутом «message».

– Данный служебный метод возвращает имя Вида в виде строки String. В данном случае, запрашиваемый Вид имеет имя «hello».

Spring MVC поддерживает множество типов Видов для различных технологий отображения страницы. В том числе — JSP, HTML, PDF, Excel, XML, Velocity templates, XSLT, JSON, каналы Atom и RSS, JasperReports и проч. Но чаще всего используются шаблоны JSP, написанные при помощи JSTL или HTML файлы, используя Thymeleaf.

2.4 Техническое обеспечение

Техническое обеспечение (ТО) - комплекс технических средств, предназначенных для работы информационной системы, а также

соответствующая документация на эти средства и технологические процессы [18].

Техническое обеспечение — технические средства, аппаратура и оборудование, используемые в информационных технологиях.

В техническом обеспечении можно выделить:

— аппаратные компоненты;

— телекоммуникационную аппаратуру и элементы;

— дополнительные компоненты.

Под аппаратными компонентами понимают компьютеры, устройства сбора, накопления, обработки информации, средства оргтехники и т.д.

При написании диплома использовался ноутбук с следующими характеристиками, описанными в таблице 2.1.

Таблица 2.1. Характеристика используемого ноутбука.

Общая информация	
Дата выхода на рынок	2018 г.
Продуктовая линейка	Dell G
Процессор	
Платформа (кодовое название)	Intel Coffee Lake (2018)
Процессор	Intel Core i7
Модель процессора	Intel Core i7 8750H
Количество ядер	6
Тактовая частота	2 200 МГц
Turbo-частота	4 100 МГц
Энергопотребление процессора (TDP)	45 Вт
Встроенная в процессор графика	Intel UHD Graphics 630
Экран	

Диагональ экрана	15.6"
Разрешение экрана	1920 x 1080
Частота матрицы	60 Гц
Технология экрана	IPS
Поверхность экрана	матовый
Экран	несенсорный
Оперативная память	
Тип оперативной памяти	DDR4
Частота оперативной памяти	2666 МГц
Объём памяти	8 ГБ
Максимальный объём памяти	32 ГБ
Всего слотов памяти	2
Хранение данных	
Конфигурация накопителя	HDD 1000 ГБ + SSD 128 ГБ
Тип накопителя	SSD+HDD
Ёмкость накопителя	1000+128 ГБ
Скорость вращения	5400 RPM
Количество слотов для SSD (формат M.2)	1 слот
Интерфейс установленного SSD	SATA
Графика	
Модель видеокарты	NVIDIA GeForce GTX 1050 Ti 4 ГБ
Локальная видеопамять	4 ГБ

3 ПОРЯДОК ВВОДА В ДЕЙСТВИЕ И МЕТОДИКА ПРОВЕДЕНИЯ ИСПЫТАНИЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Требования к системному программному обеспечению и техническим средствам

Для обеспечения полноценной работы программы на персональном компьютере требуется:

— процессор AMD Athlon X4 840 3.10 GHz;

— ОЗУ: не менее 4Гб;

— HDD: не менее 250Гб;

— видеоадаптер: DirectX9;

— видеопамять: не менее 512 МБ;

— Windows 7 и выше;

— MS SQL 8 для базы данных;

Из программного обеспечения, для запуска и работы приложения требуется:

— Java 11;

— Maven 3.6.0

— MySQL 8 (с данным для входа: логин root, пароль 1234, если данные другие – изменить соответственно в проекте в файле application.properties)

3.2 Рекомендуемый метод инсталляции

Программа является веб-приложением, что означает, что непосредственной установки приложения как такового нет, есть лишь запуск приложения с предустановкой требуемого программного обеспечения.

Для запуска проекта необходимо выполнить следующие пункты

1) Установить JDK версии 11

2) Установить Maven версии 3.0.6

3) Установить MySQL 8. При установке указать имя пользователя “root” и пароль “1234”

4) Открыть консоль. В консоли перейти в папку проекта

5) В консоли написать команду “mvn eclipse:eclipse”, если при разработке использовалась IDE Eclipse или “mvn idea:idea”, если использовалась IDEA. С помощью этой команды будут загружены все необходимые проекту зависимости, она исполняется лишь раз

6) Для запуска приложения в консоли написать “mvn clean package spring-boot:run”. Эта команда подгрузит оставшиеся зависимости, соберет и запустит проект. Окончанием запуска приложения будет служить соответствующая строка консоли.

7) В любом браузере перейти по адресу http://localhost:8080/

Приложение само создаст базу данных со всеми требуемыми таблицами при первом запуске.

3.3 Руководство пользователя

Взаимодействие с системой происходит по разному, в зависимости от роли пользователя. При открытии приложения, пользователю показывается главная страница, отображенная на рисунке ниже.

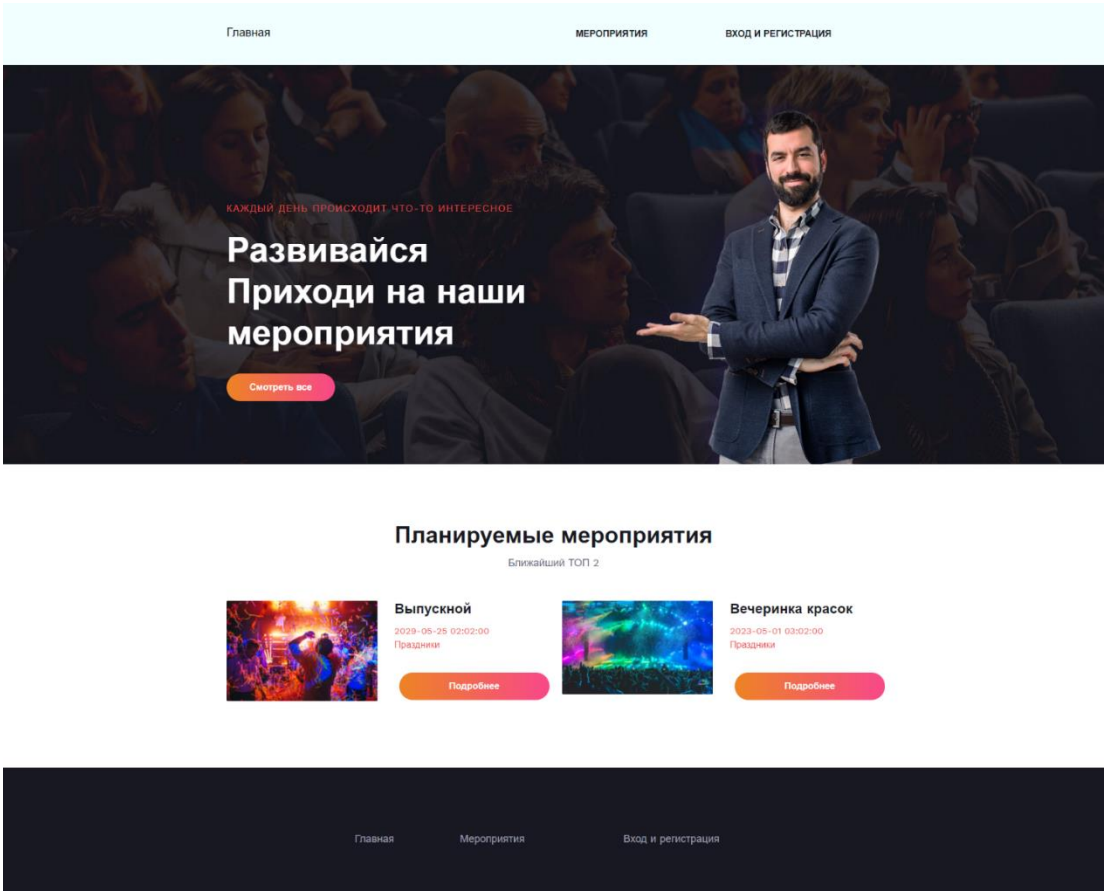

Рисунок 3.1 – Главная страница

На ней отображены два ближайших мероприятия и кнопки взаимодействия. Так как сейчас мы видим страницу от имени гостя, то можем лишь просмотреть список мероприятий, подробности каждого и зарегистрироваться или войти в систему.

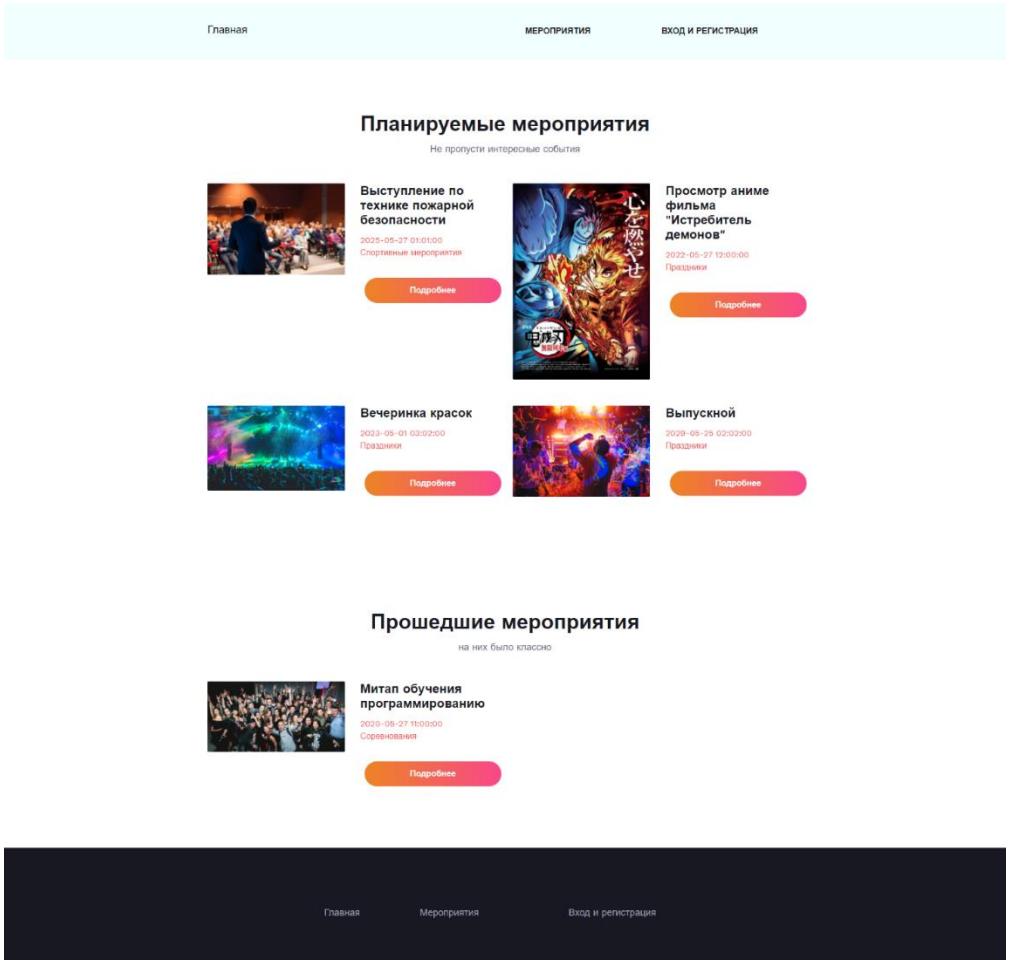

Рисунок 3.2 – Страница мероприятий

На странице мероприятий автоматически идет разделение на будущие и прошедшие. Первым делом показываются планируемые мероприятия.

При выборе варианта подробнее, будет открыто описание выбранного мероприятия.

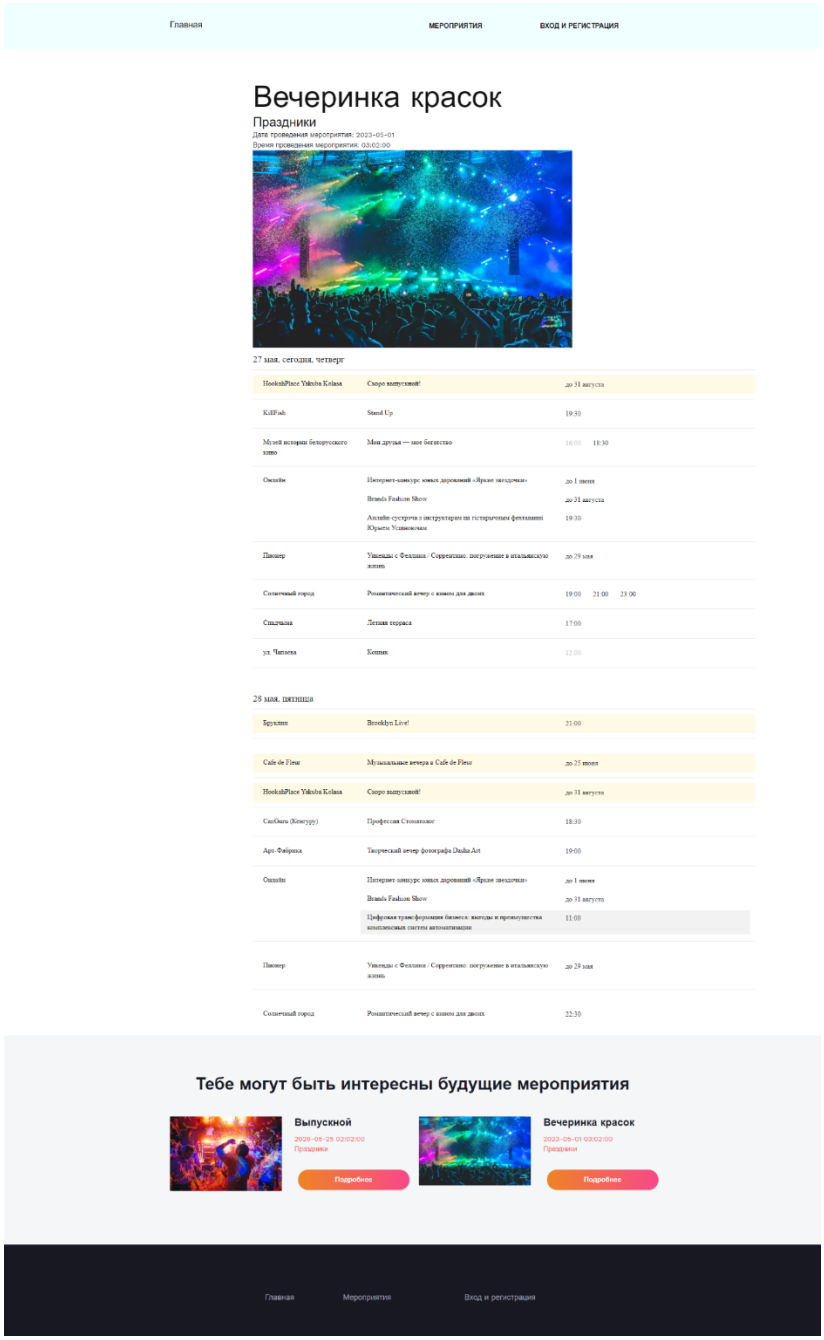

Рисунок 3.3 – Страница подробности мероприятия

Описание мероприятия может выглядеть по разному, так как при его создании разрешается форматирование текста по аналогии с вордом. Так что можно даже отформатровать текст в ворде и вставить его в специального поле при добавлении мероприятия.

Гость имеет мало вариантов взаимодействий с системой, поэтому зарегистрируем нового пользователя, перейдя на страницу регистрации.

Заполнив все поля и отправив данные, получаем уведомление о результате регистрации.

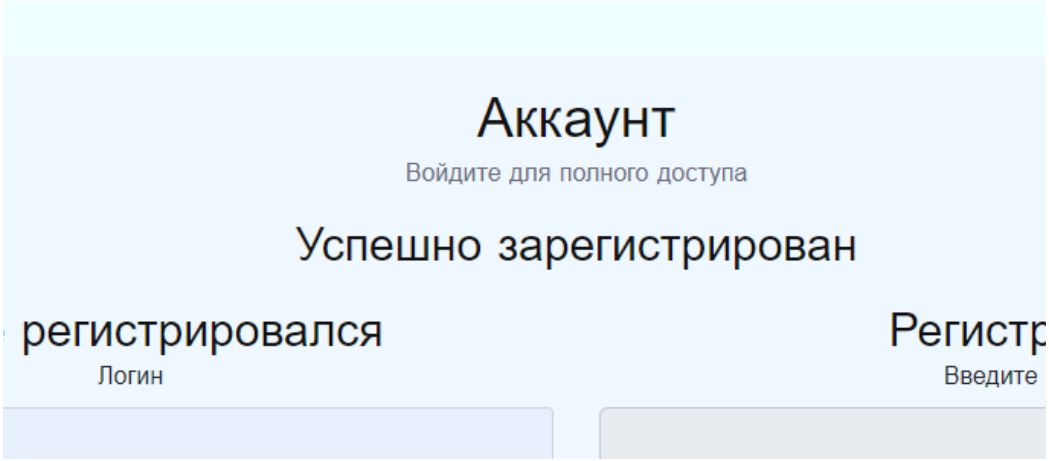

Рисунок 3.4 – Результат регистрации аккаунта

Результат успешен, пользователь зарегистрирован, и мы можем войти в систему. Входим.

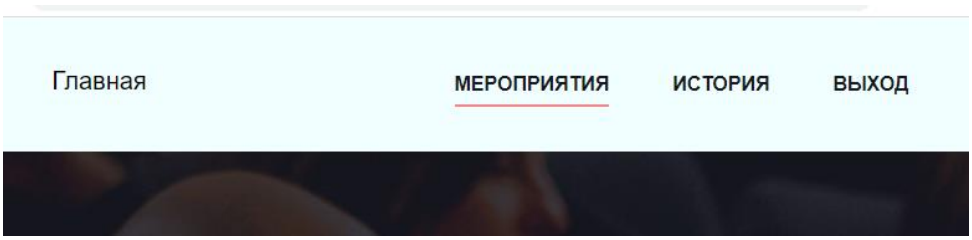

Рисунок 3.5 – Меню пользователя, вошедшего в систему

Из рисунка 3.5 можно заметить, что меню для пользователя изменилось. Кроме того, теперь на страницах планируемых мероприятий есть кнопка подтверждения участия.

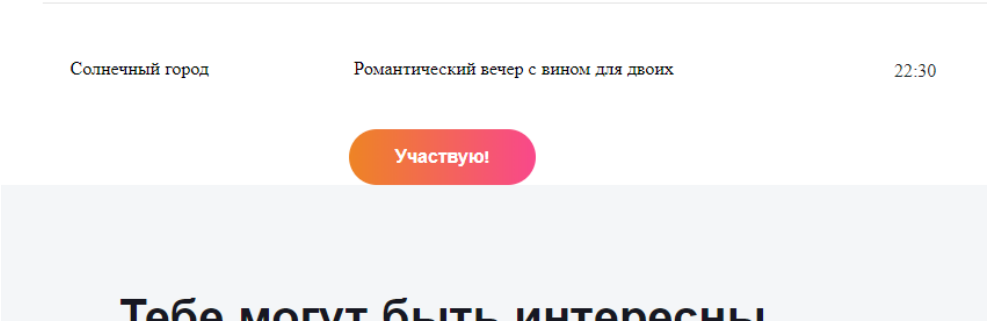

Рисунок 3.6 – Кнопка участия на странице подробности мероприятия

Или, если пользователь уже подтвердил участие для этого мероприятия, кнопка отказа от участия.

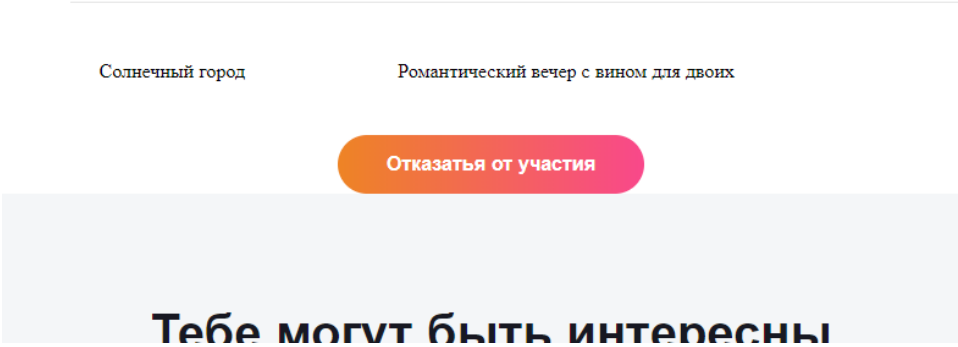

Рисунок 3.7 – Кнопка отмены участия на странице подробности мероприятия

Стоит сказать, что кнопок подтверждения и отмены участия для уже прошедших мероприятий нет, так что не получиться что-то сделать у мероприятия задним числом.

Подтвердим участие в трех мероприятиях и перейдем на страницу истории.

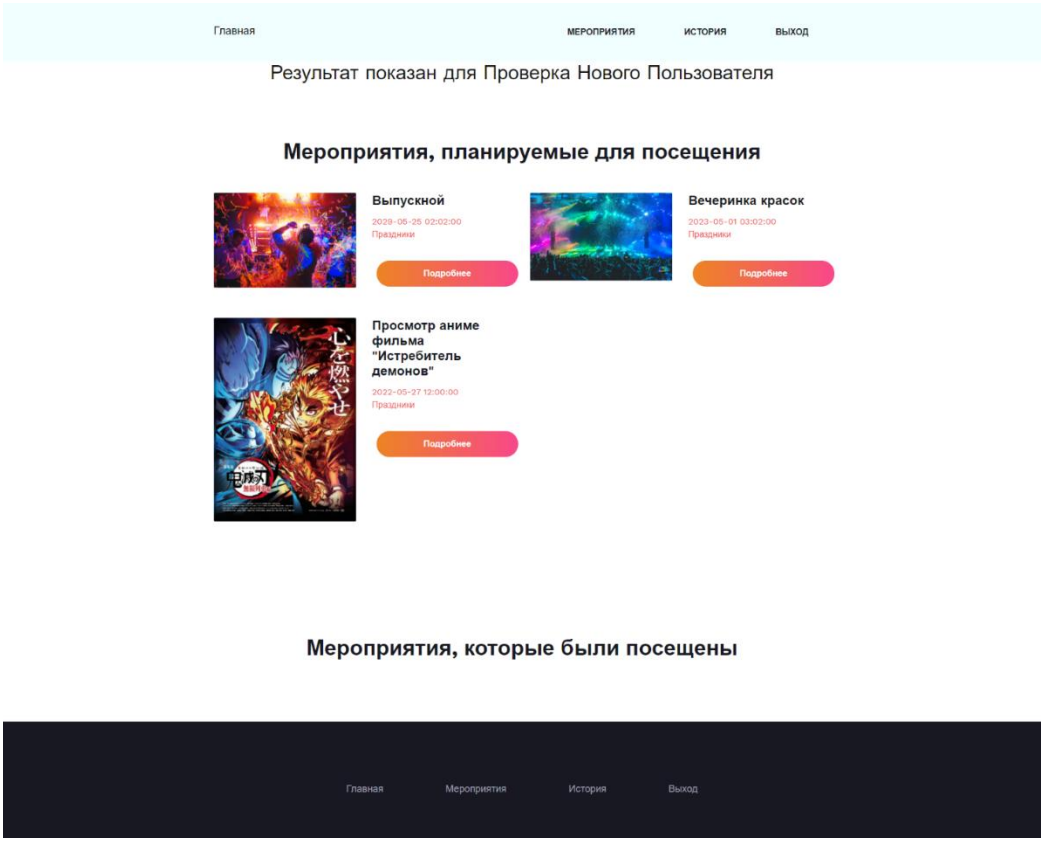

Рисунок 3.8 – Просмотр истории от имени пользователя

На странице отображены все мероприятия, отмечены пользователем для посещения.

Войдем в систему от имени администратора. У нас также изменится меню.

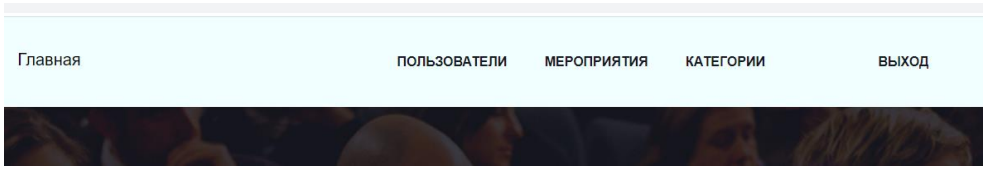

Рисунок 3.9 – Меню для администратора

От имени администратор доступны прямые взаимодействия с данными пользователей, категорий мероприятий и самих мероприятий.

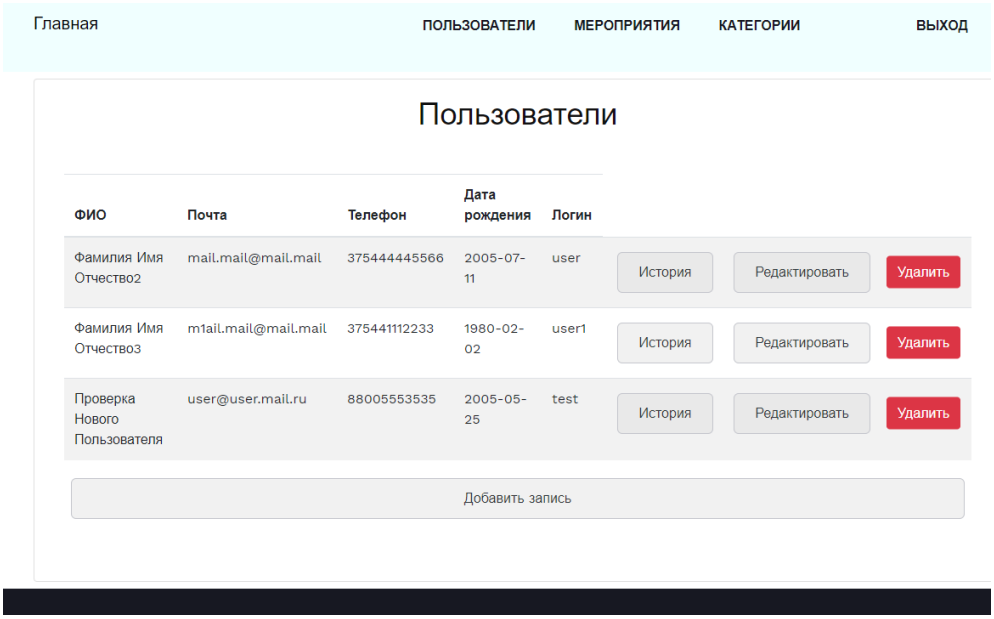

Рисунок 3.10 – Страница работы с пользователями

На странице работы с пользователями можно их редактировать, добавлять, удалять или просматривать историю их мероприятий.

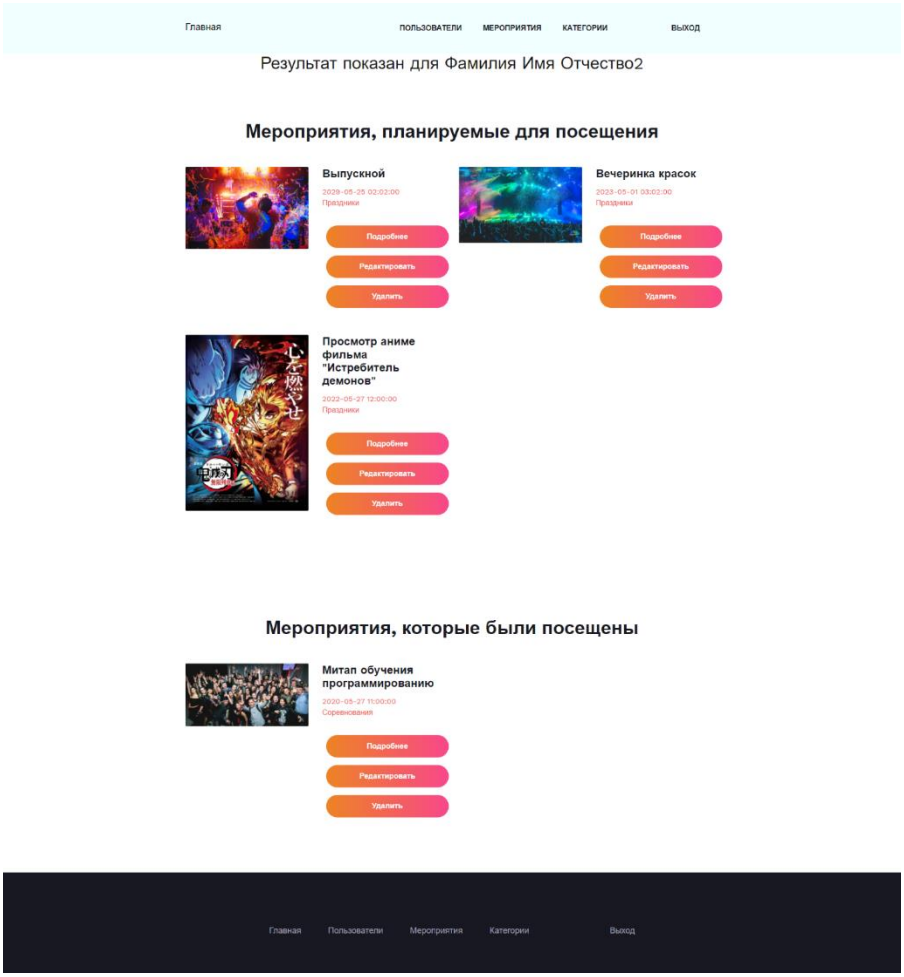

Рисунок 3.11 – Страница истории мероприятий

На рисунке выше как раз показан пример просмотра всех мероприятий выбранного пользователя от имени администратора.

Если нужно что-то изменить у пользователя, достаточно выбрать соответственный пункт и изменить данные.

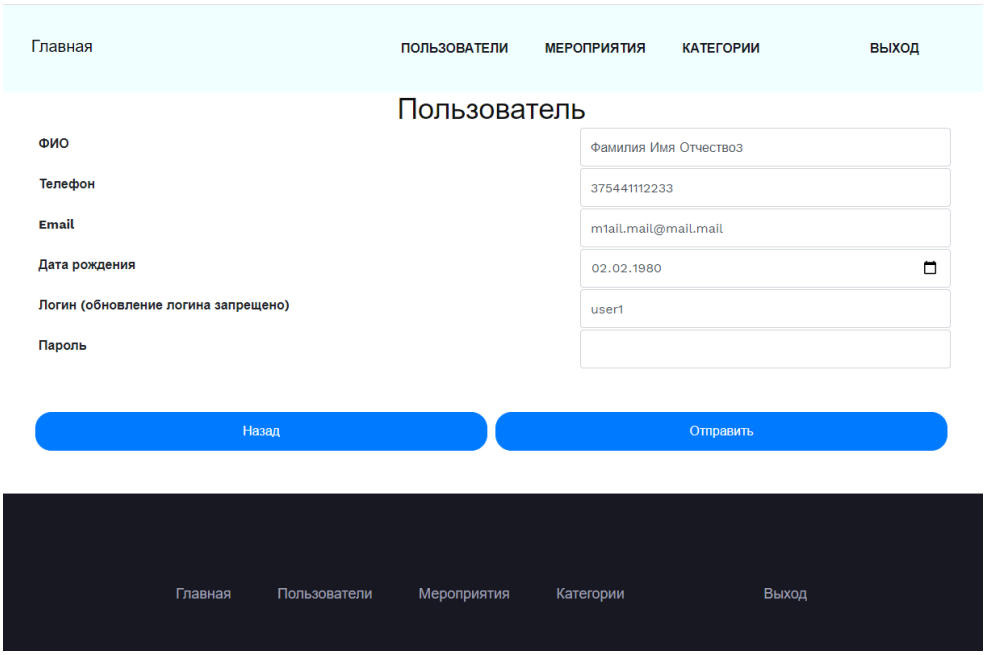

Рисунок 3.12 – Страница редактирования/создания пользователя

Для сортировки мероприятий используются категории, с которыми также взаимодействует администратор.

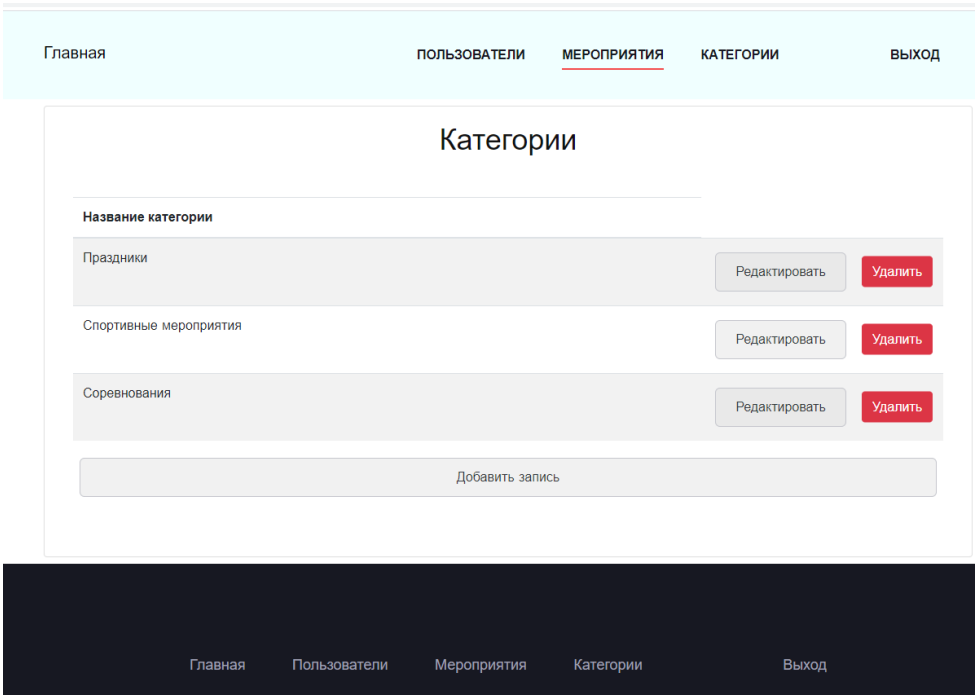

Рисунок 3.13 – Страница работы с категориями

При просмотре списка мероприятий, администратору доступна работа с каждым из них или добавление нового.

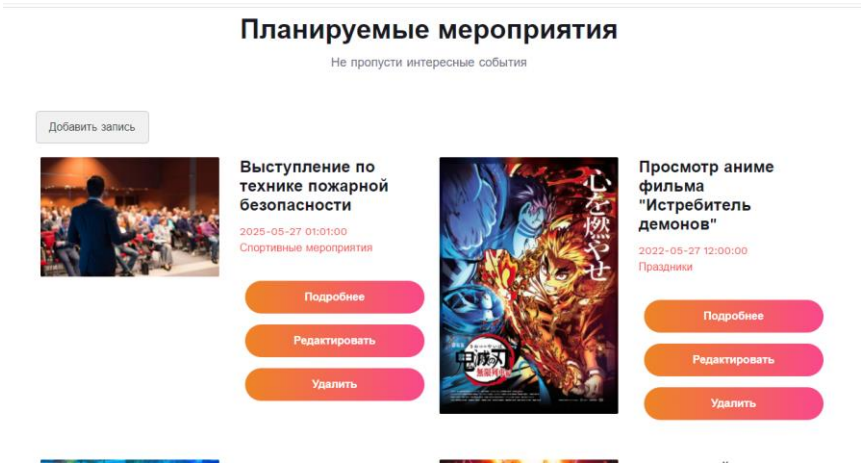

Рисунок 3.14 –Меню взаимодействия с мероприятиями от имени администратора

Как было описано выше, при добавлении/изменении мероприятия используется поле, поддерживающее форматирование текста.

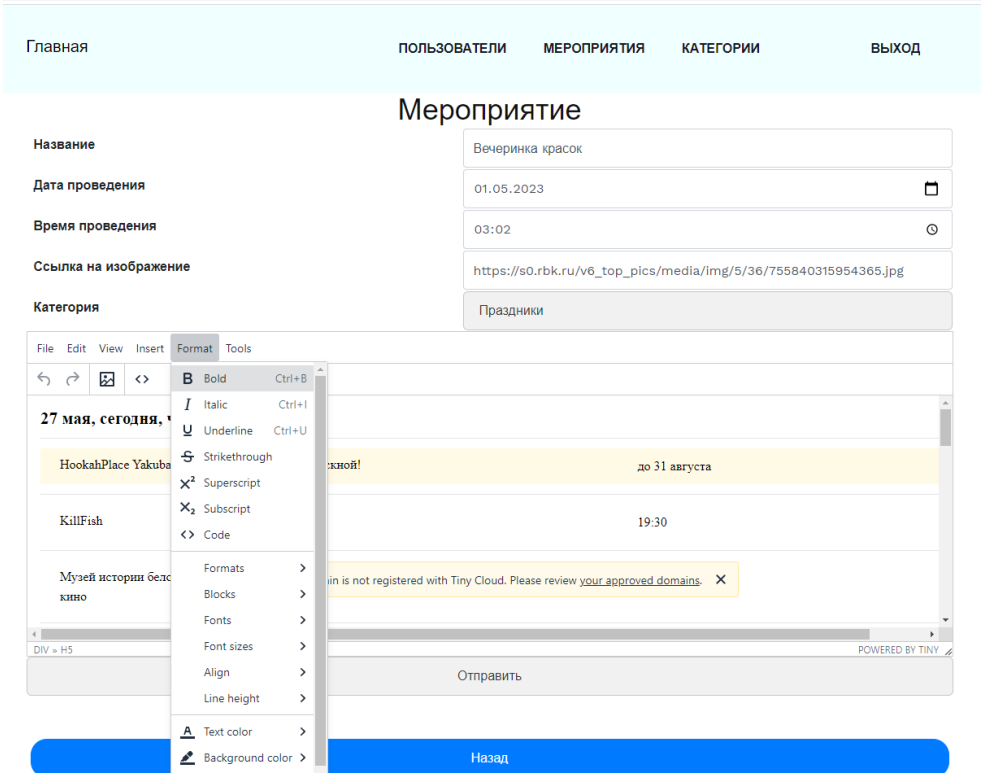

Рисунок 3.15 – Страница добавления/редактирования мероприятия

При изменении дат мероприятий, система автоматически выбирает, где их отображать: в прошедших или планируемых. Изменим дату проведения для выбранного мероприятия.

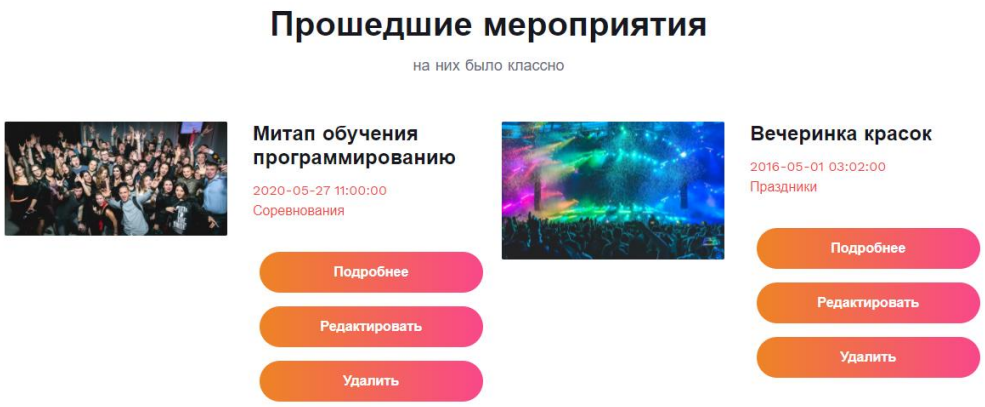

Рисунок 3.16 – Изменение места отображения мероприятия, которое закончилось

Как видно, при выборе у мероприятия даты проведения как прошедшую, оно начинает отображаться в списке уже прошедших.

Стоит отметить главную часть, при просмотре подробностей любого мероприятия, администратору одновременно отображается количество участников мероприятия и полный его список с подробностями по каждому участнику.

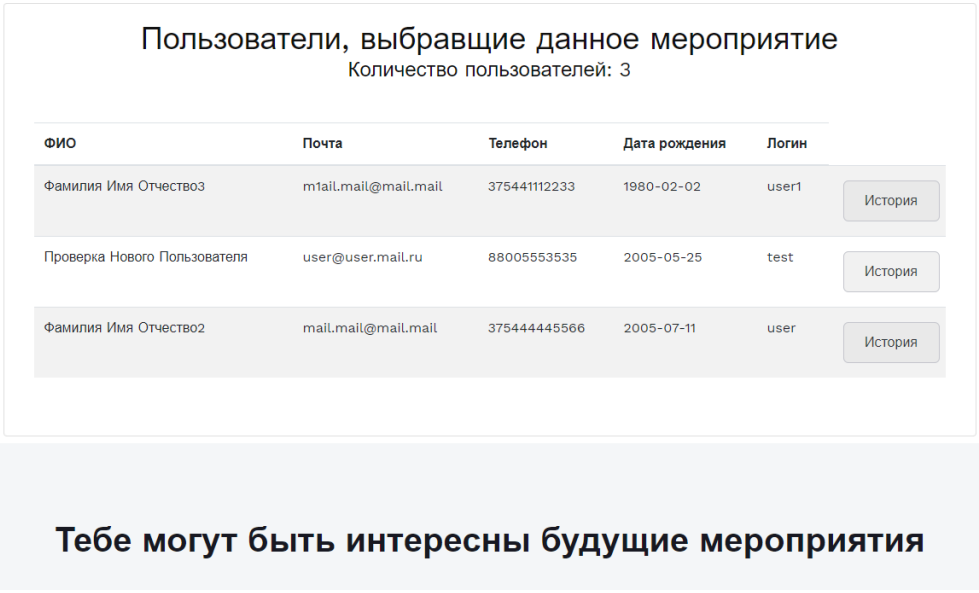

Рисунок 3.17 – Отображение списка пользователей и их количества для выбранного мероприятия

С этой страницы, как и со страницы работы с пользователями, администратор может перейти на страницу истории мероприятий каждого пользователя.

3.4 Тестирование разработанной системы

Для проверки работы спроектированного ПС разработаны тесты, с помощью которых можно оценить корректную работу веб-сервиса.

В таблице 3.1 приведены результаты тестирования для формы авторизации пользователя, которая отображается на главной странице веб-сервиса, были рассмотрены более вероятные сценарии поведение пользователей.

Таблица 3.1 – Набор тестов авторизации в веб-сервисе

Место проведения теста	**Содержание теста**	**Ожидаемый результат**	**Отметка о прохождении теста**

Страница входа с адресом /login	Введены некорректные данные в любое из полей, нажата кнопка «Войти».	Сообщение пользователю об ошибке, очистка заполненных полей.	Да
Страница входа с адресом /login	Введен корректный логин и пароль, нажата кнопка «Войти».	Отображение главной страницы сайта, отображение соответствующих функциональных возможностей.	Да

В таблице 3.2 приведены результаты тестирования возможностей администратора веб-сервиса. Проверен функционал добавления нового мероприятия и редактирования уже существующей информации о мероприятии. Рассмотрены основные сценарии использования администратором.

Таблица 3.2 – Набор тестов для проверки возможностей администратора

Место проведения теста	**Содержание теста**	**Ожидаемый результат**	**Отметка о прохождении теста**
Страница «Мероприятия», добавление записи	Все поля корректно заполнены, нажата кнопка «Отправить».	Переход на страницу с списком мероприятий, где есть только что добавленный	Да
Страница «Мероприятия», добавление записи	Любое из полей не заполнено, нажата кнопка «Отправить».	Отображение информации об ошибке, ожидание заполнения полей	Да
Страница «Мероприятия», редактирование записи	Изменение информации в поле времени, перенос время с будущего числа на прошедшее. Нажата кнопка «Отправить».	Отображение страницы мероприятий с учетом изменённых данных, т.е. мероприятие показывается уже не в блоке планируемых, а в прошедших.	Да
Страница «Услуги тарифа»	Ожидание вывода информации.	Отображение информации об	Да

		услугах только выбранного тарифа.	

Таблица 3.3 – Набор тестов для проверки возможностей пользователя

Место проведения теста	**Содержание теста**	**Ожидаемый результат**	**Отметка о прохождении теста**
Главная страница	Выбрать мероприятия для просмотра	Переход на страницу подробностей. Если пользователь не выбирал участие в мероприятии, отображается кнопка участия, иначе кнопка отмены участия	Да
Главная страница, страница подробностей	Выбрать мероприятия для просмотра, которое еще не было отмечено для участия. Нажать кнопку участия.	Переход на страницу подробностей. Вместо кнопки участия отображается кнопка отмены участия	Да

В таблице 3.3 приведены результаты тестирования возможностей пользователя веб-сервиса. Рассмотрены основные сценарии использования возможностей веб-сервиса пользователем.

Таблица 3.4 – Набор тестов отображения информации

Место проведения теста	**Содержание теста**	**Ожидаемый результат**	**Отметка о прохождении теста**
Все страницы	Перейти на любую из вкладок в произвольном порядке.	Корректное отображение информации, отображение того,	Да

		чего ожидает пользователь.	
Все вкладки	Увеличение масштаба отображения средствами браузера.	Появление скроллов справа и снизу, корректное отображение информации.	Да
Все вкладки	Обновление страницы используя средства браузера.	Состояние веб-сервиса до обновления и после не изменилось.	Да
Меню браузера	Нажатие функциональных кнопок браузера «Назад» и «Вперед» в произвольный момент работы веб-приложения.	Корректное поведение веб-сервиса. Отображение информации.	Да

Были проведены экспериментальные проверки на реальных данных. Разработанный веб-сервис полностью реализует все поставленные задачи. Весь реализованный функционал работает корректно, что было подтверждено как наборами тестов, так и визуальным тестированием.

ЗАКЛЮЧЕНИЕ

Итогом написания дипломного проекта является программное веб-приложение, которое предоставляет собой сайт с мероприятиями с возможностью просмотра подробностей по каждому мероприятию для любого пользователя, а также просмотру количества и списка участников каждого мероприятия для администратора. Данное приложение является доступным и понятным любому пользователю, располагает удобным и минималистичным интерфейсом, а также соответствует всем требованиям, предъявленным к проекту.

Данная программа является web- приложением, что говорит о том, что множество пользователей, расположенных на определенном расстоянии друг от друга, могут одновременно общаться с сервером.

Вся используемая информация хранится в базе данных, разработанной для данного проекта. Доступ к базе данных был реализован только со стороны веб-сервиса.

Безопасность данных обеспечивается за счет разграничения прав доступа, системы авторизации.

Разработанная программа позволяет произвести быстрый и удобный учет участником мероприятий, учет зарегистрированных пользователей, а также видеть списки посещений каждого пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мероприятие - это... А что это такое? [Электронный ресурс]. – Режим доступа: www.syl.ru/article/340279/meropriyatie---eto-a-chto-eto-takoe. – Дата доступа: 28.05.2021.

2. Информационные системы [Электронный ресурс]. – Режим доступа: https://www.softacom.ru/ru_informationsystemsdevelopment. – Дата доступа: 28.05.2021.

3. Разработка требований к информационной системе [Электронный ресурс]. – Режим доступа: https://searchinform.ru/services/outsource-ib/zaschita-informatsii/razrabotka-trebovanij-po-ib/razrabotka-trebovanij-k-informatsionnoj-sisteme/. – Дата доступа: 28.05.2021.

4. Структура процесса моделирования и содержание его этапов. [Электронный ресурс]. – Режим доступа: studopedia.su/17_39489_metod-postroeniya-dereva-vzaimosvyazey.html. – Дата доступа: 28.05.2021.

5. IDEF0. Знакомство с нотацией и пример использования. [Электронный ресурс]. – Режим доступа: https://trinion.org/articles/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya/. – Дата доступа: 28.05.2021.

6. Виды обеспечений ИС [Электронный ресурс]. – Режим доступа: https://studizba.com/lectures/10-informatika-i-programmirovanie/298-informacionnye-sistemy/3903-41-vidy-obespecheniy-is.html. – Дата доступа: 28.05.2021.

7. Инструментарий для моделирования и реинжиниринга бизнес-процессов [Электронный ресурс]. – Режим доступа: https://studbooks.net/2036211/informatika/modelirovanie_dannyh_notatsii_idef1x. – Дата доступа: 28.05.2021.

8. UML — диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/47940/. – Дата доступа: 28.05.2021.

9. Серверные языки: Java (обзор) [Электронный ресурс]. – Режим доступа: https://codomaza.com/article/servernye-jazyki-java-obzor. – Дата доступа: 28.05.2021.

10. Что такое Spring Framework? От внедрения зависимостей до Web MVC [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/490586/. – Дата доступа: 28.05.2021.

11. Основы HTML [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics. – Дата доступа: 28.05.2021.

12. CSS [Электронный ресурс]. – Режим доступа: semantica.in/blog/css.htm. – Дата доступа: 28.05.2021.

13. Обоснование выбора языка программирования для создания интернет магазина [Электронный ресурс]. – Режим доступа: https://studbooks.net/2258837/informatika/obosnovanie_vybora_yazyka_programmirovaniya_sozdaniya_internet_magazina. – Дата доступа: 28.05.2021.

14. Microsoft Visio Professional 2016 [Электронный ресурс]. – Режим доступа: https://itpro.ua/product/microsoft-visio-professional-2016/?tab=description. – Дата доступа: 28.05.2021.

15. База данных MySQL [Электронный ресурс]. – Режим доступа: https://coderbook.ru/sql/база-данных-mysql/. – Дата доступа: 28.05.2021.

16. IntelliJ IDEA — решение для топ-разработчиков. [Электронный ресурс]. – Режим доступа: https://webformyself.com/intellij-idea-reshenie-dlya-top-razrabotchikov/ – Дата доступа: 28.05.2021.

17. Spring MVC — основные принципы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/336816/. – Дата доступа: 28.05.2021.

18. Информационные системы [Электронный ресурс]. – Режим доступа: elib.bsu.by/bitstream/123456789/225446/1/Информационные%20системы.pdf. – Дата доступа: 28.05.2021.