ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ИНТЕРАКТИВНОЙ КАРТЫ НИУ «БелГУ»

Выпускная квалификационная работа
обучающегося по направления подготовки 09.03.02
Информационные системы и технологии
очной формы обучения, группы 07001409
Скубникова Кирилла Сергеевича

Научный руководитель
к.т.н., доцент
Гахов Р.П.

БЕЛГОРОД 2018

РЕФЕРАТ

Разработка модуля автоматизации организации событий средствами Интернет. – Скубников Кирилл Сергеевич, выпуская квалификационная работа бакалавра. Белгород, ФГАОУВО Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 48, включая приложения 50, количество рисунков 31, количество использованных источников 32.

КЛЮЧЕВЫЕ СЛОВА: автоматизация, рассылка, мессенджер, шаблон электронного письма, права доступа, распределенная система, микросервис, прогрессивное веб приложение, сетевые технологии, инверсия контроля, масштабирование, одностраничное приложение, объектно-ориентированные технологии.

ОБЪЕКТ ИССЛЕДОВАНИЯ: процессы организации событий средствами Интернет.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: методы построение распределенных систем и агрегация каналов распространения информации.

ЦЕЛЬ РАБОТЫ: оптимизация использования человеческих ресурсов в рамках процесса организации событий путем создания программного инструмента.

ЗАДАЧИ ИССЛЕДОВАНИЯ: анализ существующих подходов реализации машинного зрения, обработки изображения и поиска на нём ключевой информации, анализ полученной информации, её пров ерка по базе знаний. Разработка информационной системы, позволяющей с высокой точностью определять подлинность лекарственного средства.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы была разработана программная реализация модуля автоматизации организации событий средствами Интернет, осуществляющая решение поставленных задач и достижение основной цели работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время уровень социализации современного информационного общества растет в геометрической прогрессии. Данную тенденцию поддерживает непрерывно прогрессирующий уровень интеграции информационных технологий в различные отрасли человеческой деятельности.

Общая потребность во внедрении информационных технологий в социальные и технологические процессы обусловлена повышением уровня сложности как организационных структур, так и мультипарадигмальных механизмов взаимодействия между ними. Также текущий вектор развития информационных технологий объясняется повсеместным распространением доступа к сети Интернет – только в Российской Федерации ежедневно порядка семидесяти восьми миллиона [1] человек (по состоянию на 2016 год) пользуется системой объединенных компьютерных сетей [2].

Потребность в упрощении, автоматизации бизнес-процессов возникает у потребителей всех уровней. Описанная тенденция развития информационного общества позволяет использовать сеть Интернет как площадку для реализации маркетинговых процессов.

Примером такого варианта использования является процесс организации событий, который включает в себя целый перечень гетерогенных процедур, затрагивающих, в случае предприятия, работу комплекса организационных структур.

Даже с имеющимся открытым доступом к всевозможным каналам и платформам распространения, демонстрации информации существует проблема их агрегации и минимизации трудозатрат при одновременном использовании [3]. Данный факт демонстрирует множество точек потенциальной избыточности использования человеческих ресурсов, предрасположенных для внедрения информационных автоматизированных систем [4].

Для организации событий принято использовать специальные онлайн площадки, являющиеся средством публикации информации и предоставляющие возможность регистрации и учета участников. Данный подход реализует механизмы привлечения юридических и физических лиц для их дальнейшего внедрения в бизнес-процессы компании. Определенному проценту пользователей такой способ организации может оказаться достаточным, но в действительности у крупных организаций требования и цели значительно выше.

У бизнеса любого уровня есть базы текущих и потенциальных клиентов. В случае с процессом организации событий, участие таких лиц является наиболее приоритетным [5]. При этом возникает задача распространения приглашений на мероприятия. Её решение позволит повысить стабильность уровня текущей бизнес-нагрузки и обеспечить ее потенциальный рост.

Помимо этого, существует вероятность возникновения непредвиденных обстоятельств, изменяющих место и время проведения событий. Данный фактор также подводит к проблеме последующего информирования участников мероприятия.

На текущий момент для решения определенных выше проблем не существует универсального решения. Чаще всего процесс организации событий включает в себя личное обращение к каналам связи и площадкам размещения информации о событиях. В связи с этим, в рамках описанного контекста, задача автоматизации и упрощения рассматриваемого процесса обладает высокой актуальностью, а отсутствие целостных решений только укрепляет её.

Предметом исследования работы являются м методы построения распределенных систем и агрегация каналов распространения информации, реализация которых позволит упростить и оптимизировать процесс отправки сообщений, оповещений, рассылок, а также предоставляющий площадку для организации событий с возможностью дальнейшей регистрации пользователей

и их учета для последующего материального планирования и привлечения новых клиентов.

Целью работы является оптимизация использования человеческих ресурсов в рамках процесса организации событий путем создания инструмента, обладающего интуитивно понятным интерфейсом, максимальным охватом доступных каналов доставки сообщений и информации удовлетворяющего всем современным экономическим и функциональным требованиям. Также необходимо выделить процесс организации событий средствами Интернет в качестве объекта данной работы. Данный проект реализуется в рамках заказа российского представительства компании «Pegasystems inc.» на платформе работодателя ООО «Технологии надежности». Поставленные цели определяют следующие задачи:

а) произвести анализ бизнес-процесса организации событий «Как есть» на основе построенный модели, отражающей текущий подход организации для выявления недостатков и структурных единиц, подлежащих оптимизации, а также формирования предложения по решению проблем;

б) сформировать рабочее техническое задание на основе пожеланий заказчика и выявленных недостатков текущей стратегии реализации бизнес-процесса для определения дальнейшего подхода к реализации проекта.

в) построить модель бизнес-процесса организации событий «Как должно быть», наглядно отражающей вносимые изменения в структуру компании;

г) проектирование архитектуры программной реализации для определения основных функциональных узлов разработки и взаимодействия между ними;

д) проектирование схем баз данных, удовлетворяющих определенной архитектуре и структурным требованиям;

е) программная реализация и тестирование.

Работа выполнена на 48 страницах, с приложениями – на 50, содержит 31 рисунок и ссылается на 32 источника.

1 Аналитическое обеспечение задачи

Неотъемлемым этапом реализации проектов любой сложности и направленности является проведение анализа цели работы для дальнейшего формирование ключевых требований, свойств и ограничений разрабатываемой системы.

1.1 Анализ текущего бизнес-процесса организации событий

На начальном этапе разработки модуля автоматизации организации событий средствами Интернет необходимо произвести анализ рассматриваемого бизнес-процесса. Это позволит сформировать структурированное понимание решаемой задачи и проблем в целом. Также формализация модели структуры устройства организационных единиц компании наглядно предоставляет возможность обоснованной оценки текущего порядка осуществления процессов и задач внутри компании.

Построение модели бизнес-процесса организации событий «Как есть» производится в нотации IDEF0, отображающей структуру и общие функции системы, а также связующие их потоки материальных объектов и информации [6]. Диаграммы, разработанные с применением данной нотации, позволяют комплексно оценить текущую степень загруженности отдельных функциональных узлов процесса, распределение обязанностей между отделами и конкретными должностными лицами. Также подобное представление отдельных действий в рамках целого процесса отражает движения материалов, объектов и обосновывает потребности в них [7].

На раннем этапе моделирования необходимо произвести глобальный обзор осуществляемой деятельности. Данный шаг позволит максимально широко оценить задействованные исполнительные, информационные, материальные ресурсы и продуцируемые результаты рассматриваемого

процесса. Инструментом решения данной задачи в нотации IDEF0 является контекстная диаграмма, отражающая внешнюю структуру бизнес-процесса [8].

Контекстная диаграмма процесса организации событий отражена на рисунке 1.1.

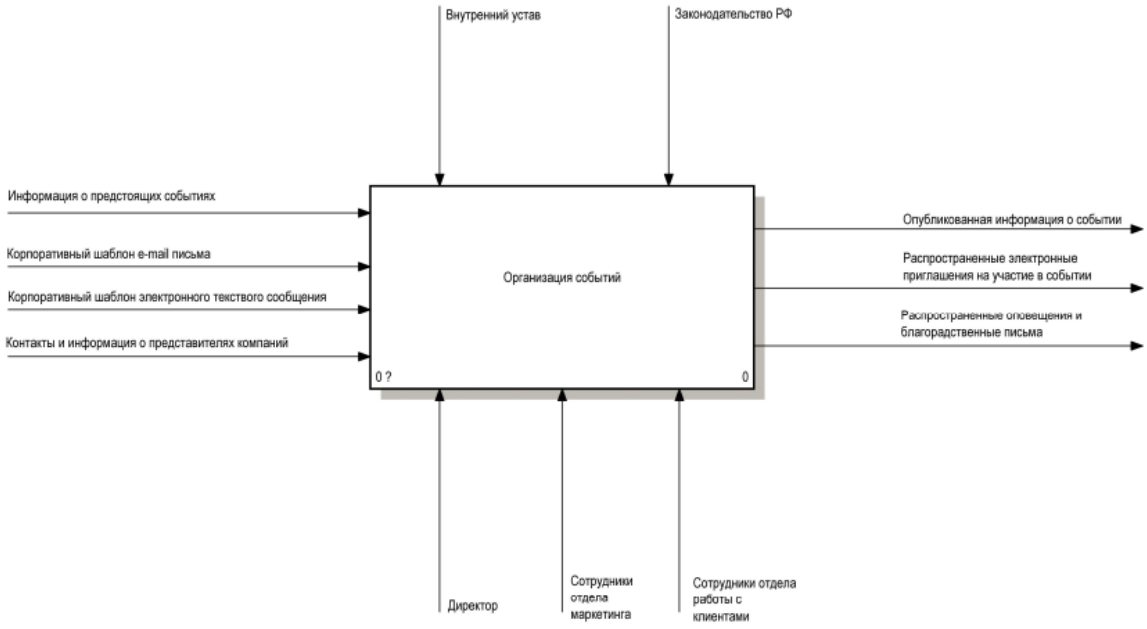

Рисунок 1.1 – Контекстная диаграмма процесса организации событий

На данном этапе необходимо детально рассмотреть основные компоненты текущего устройства процедуры организации событий. В качестве средств управления в процессе высшего уровня можно выделить следующие сущности:

– внутренний устав – регламентирует внутренний порядок проведения типовых операций и процедур;

– законодательство РФ – регулирует аспекты хранения и использования персональных данных пользователей, чья контактная информация задействована в процессе рассылки.

Входными данными рассматриваемого бизнес-процесса являются:

– информация о предстоящих событиях, которая содержит в себе тематику мероприятия, предварительную развернутую информацию;

– корпоративный шаблон E-mail письма представляет из собой разметку сообщения с учетом фирменного стиля и предусмотренными полями для последующего заполнения;

– корпоративный шаблон электронного текстового сообщения включает в себя набор типичных фирменных словесных вставок;

– контакты и информация о представителях компаний включает в себе идентификаторы в каналах распространения информации, используемые при рассылке, а также основные сведения о получателях: ФИО, наименование компании и занимаемая в ней должность.

В роли механизмов диаграммы определены:

– директор, принимающий непосредственное участие в большинстве из подпроцессов;

– сотрудники отдела маркетинга, отвечающие за оформление сообщений, формализацию и подготовку поступающей информации;

– сотрудники отдела работы с клиентами, реализующие внешнее взаимодействие и коммуникацию.

Продуктами рассматриваемого процесса являются:

– опубликованная информация о событии, являющаяся набором ссылок на задействованные ресурсы;

– распространенные электронные приглашение на участие в событии – экземпляры индивидуальных электронных писем и сообщений. Являются заполненными экземплярами шаблонов E-mail и текстовых рассылок;

– распространенные оповещения и благодарственные письма содержат в себе напоминания о приближающихся событиях и, по пришествию мероприятий, призыв к дальнейшему сотрудничеству на которые были приглашены адресаты.

В дальнейшем, перечисленные выше сущности будут использованы и более подробно отражены на диаграмме декомпозиции, позволяющей

детализировано проанализировать основные этапы деятельности в рамках осуществления основного процесса.

Диаграмма декомпозиции продемонстрирована на рисунке 1.2.

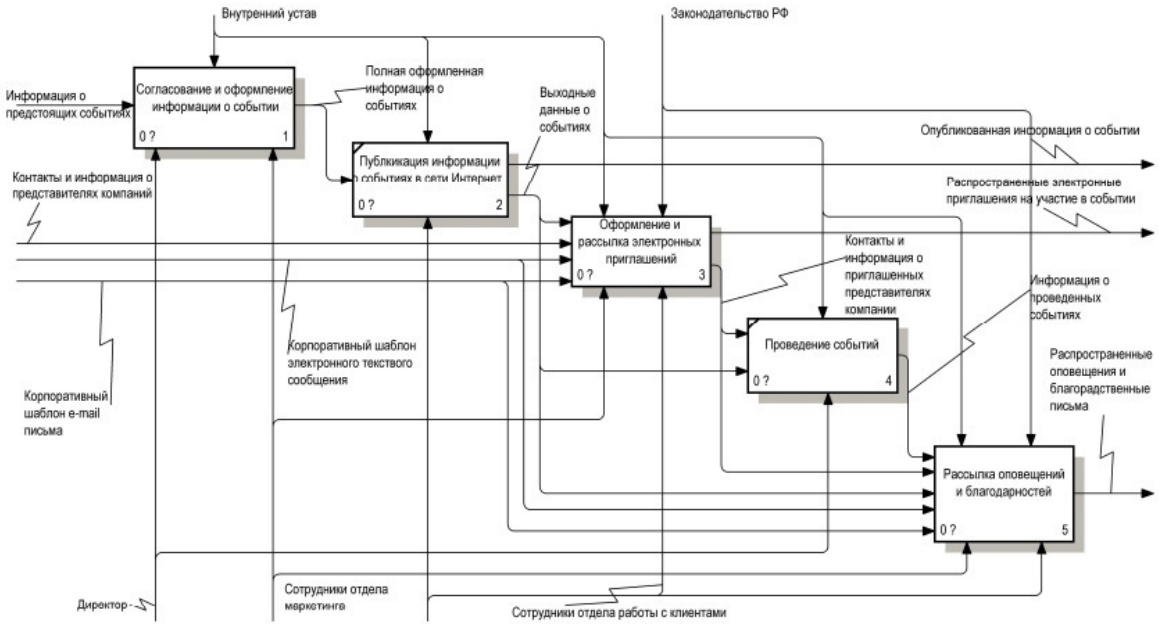

Рисунок 1.2 – Декомпозиция контекстной диаграммы

На данной диаграмме (рисунок 1.2) предоставляется возможность наблюдать основные этапы деятельности процесса:

– согласование и оформление информации о событии - формализация входной информацией с внедрением цифрового контента;

– публикация информации о событии в сети интернет – размещение новостных объявлений на специализированных ресурсах;

– оформление и рассылка электронных приглашений – заполнение корпоративных шаблонов и последующая рассылка;

– проведение событий – функция, имеющая косвенное отношение к рассматриваемому процессу, но включена в диаграмму из-за жесткой связки с остальными блоками, требующими выходную информацию;

– рассылка оповещений и благодарностей – процесс, схожий с описываемым выше. Основным отличием является использование

информации уже о проведенных событиях для её дальнейшего внедрения в шаблоны сообщений.

Далее проведем рассмотрение не атомарных подпроцессов, требующих дальнейшей декомпозиции и анализа.

На рисунке 1.3 показана диаграмма декомпозиции процесса согласования и оформления информации о событии.

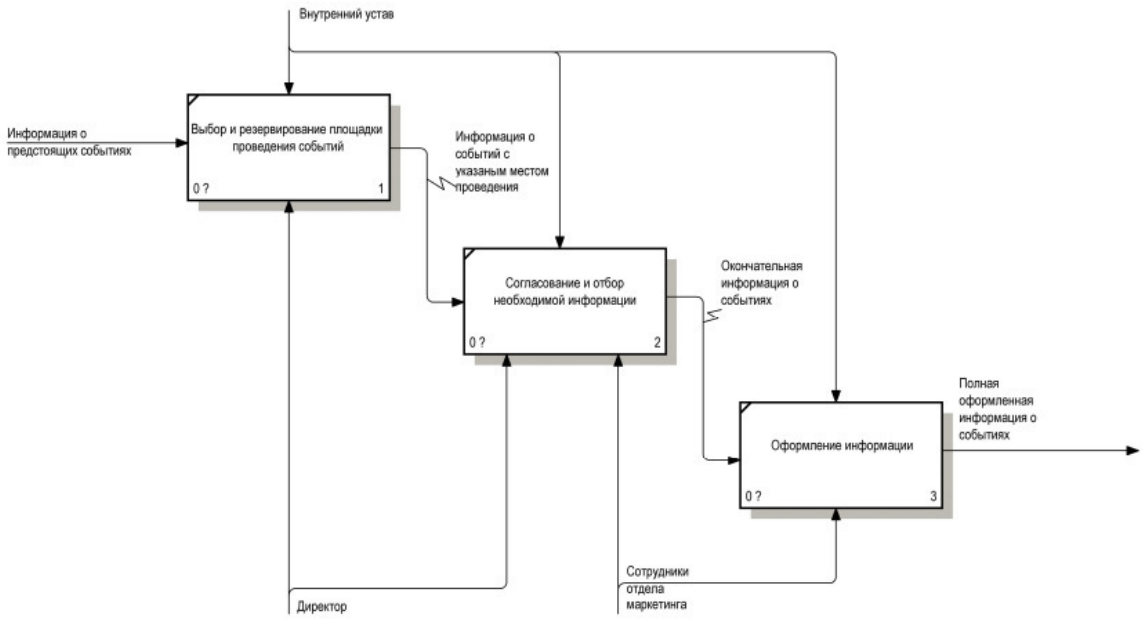

Рисунок 1.3 – Декомпозиция процесса согласования и оформления информации о событии

Рассматриваемая функция осуществляет преобразование и организацию поступающей информации о планируемых событиях. На данном этапе необходимо определить ключевые выходные данные, произвести их подготовку и оформление публикаций.

Явно выделяются потенциальные зоны автоматизации и перераспределения использования человеческих ресурсов – оформление публикаций можно производить автоматически на основе стандартных шаблонов площадок публикации событий.

Подобная ситуация складывается и с процессом оформления и рассылки электронных приглашений диаграмма которого продемонстрирована на рисунке 1.4.

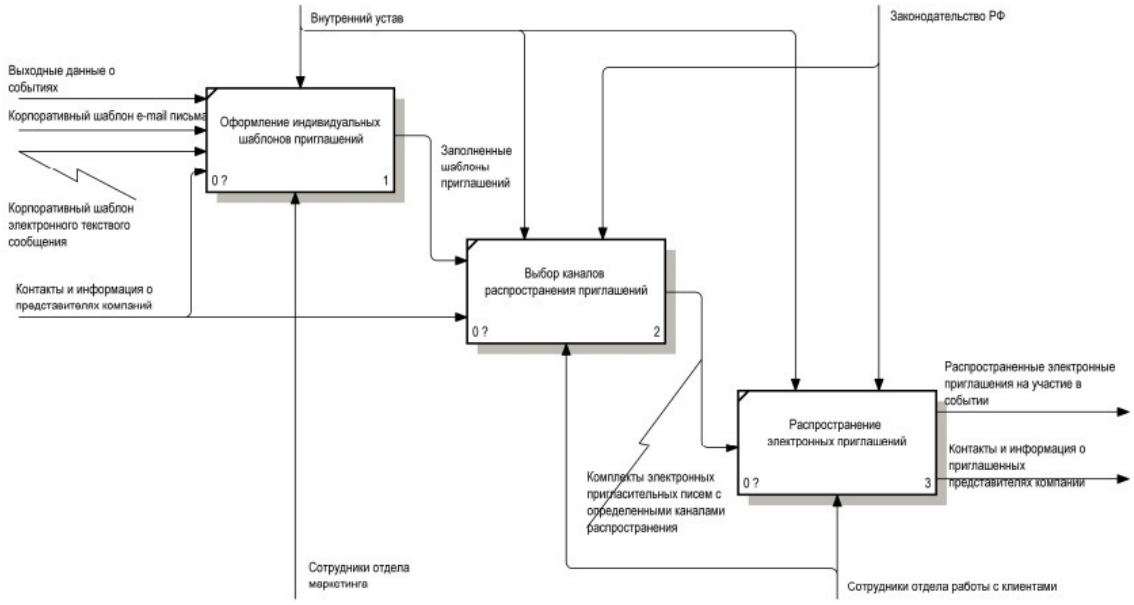

Рисунок 1.4 – Декомпозиция процесса оформления и рассылки электронных приглашений

На рассматриваемом этапе проводится оформление и рассылка цифровых сообщений. Основную часть времени занимает оформление корпоративных шаблонов без применения средств автоматизации.

Также наблюдается возможность ускорить процесс выбора каналов распространения сообщений на основе контактов представителей компании и дальнейшей рассылки.

Ключевым отличием процесса рассылки оповещений и благодарностей от оформления и рассылки электронных приглашений является наличие внутри процесса выходной информации о проведенных событиях.

Наличие такой информации сопровождается учетом общего количества посетителей мероприятия, участвующих лиц, получивших прямое приглашение, индивидуальных представителей и перечень не явившихся лиц. Данные сведения позволяют персонализировать будущие сообщения и

определить список адресатов для распространения благодарностей. Для наглядного представления необходимо осмотреть модель процесса, представленную на рисунке 1.5.

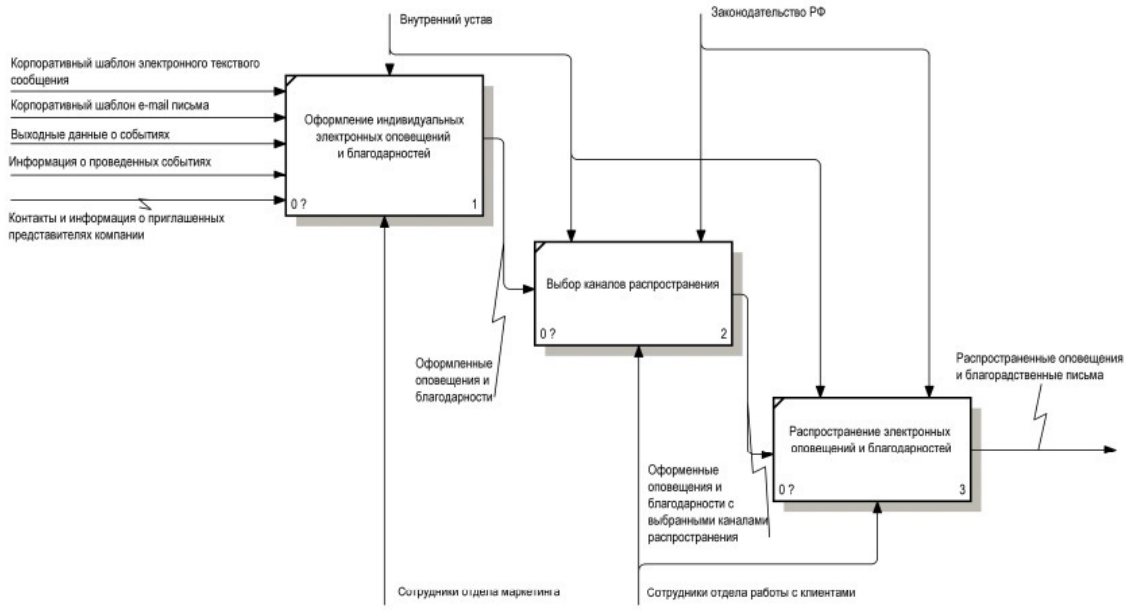

Рисунок 1.5 – Декомпозиция процесса рассылки оповещений и благодарностей

Сходство с процессом рассылки приглашений очевидно, однако не может иметь смежную функциональную структуру ввиду специфических ограничений проведения работ. Также слиянию блоков препятствуют дополнительные входные данные, которые при единой структуре будут образовывать критическую избыточность.

В крупной организации важно придерживаться принципа разделения обязанностей и формирования организационных участков, выполняющих атомарные операции. Такой подход позволит сохранить прозрачность ведения дел и обеспечить консистентность реализации бизнес-процессов.

Результатом проведения анализа текущей организации ведения процесса организации событий показывает ряд участков, подверженных избыточному использованию трудовых ресурсов и подлежащих оптимизации с целью повышения эффективности рабочего коллектива в целом.

1.2 Формирование рабочего технического задания

Для выделения и формирования требований к разработке необходимо разработать техническое задание. Подобная документация служит справочным средством для дальнейшей реализации проекта, а также играет роль первоначальной технической документации, описывающей ключевые требования [9].

1.2.1 Определение оптимизируемых узлов бизнес-процесса

На текущий момент процесс организации событий представляет из себя совокупность маршрутов между разноуровневыми организационными единицами без использования решений для автоматизации, упрощения процессов и формализации передаваемой информации. Для визуального представления действительного состояния организации используется формальный подход, наглядно отражающий основные глобальные узлы взаимодействия, реализованный посредством построения мнемосхемы, показанной на рисунке 1.6 [10]. Данная техника позволит компактно проанализировать бизнес-процесс.

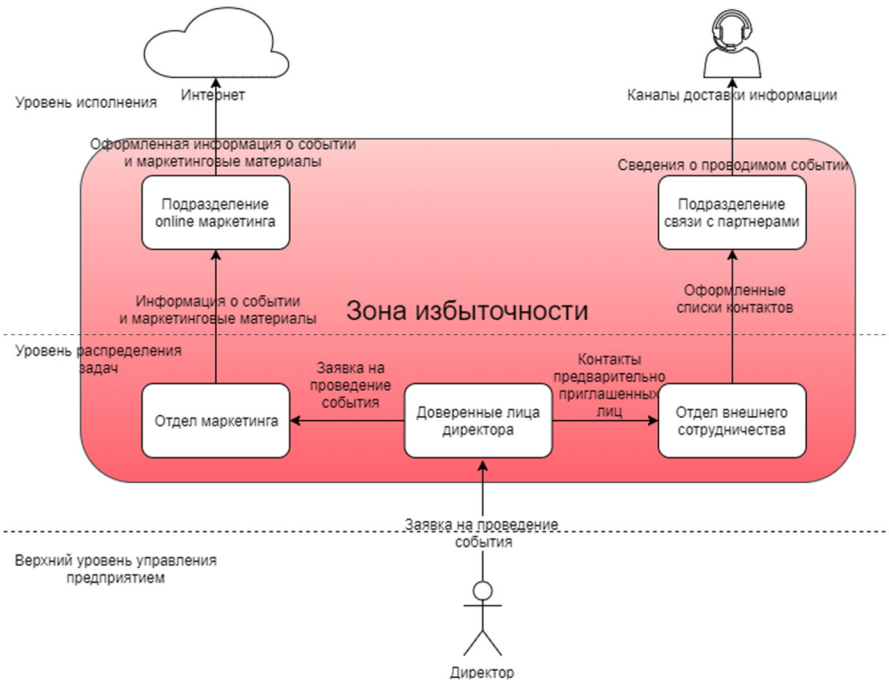

Рисунок 1.6 – Мнемосхема бизнес-процесса организации событий

Зона избыточности, выделенная красным цветом, обозначает маршруты и структурные единицы, требующие оптимизации. Построение подобной диаграммы возможно только после проведения детального моделирования бизнес-процессов компании.

1.2.2 Определение недостатков

Избыточность текущего решения очевидна – атомарная на первый взгляд операция требует участия нескольких организационных уровней. При этом образуются узкие места в виде подразделения связи с партнерами, требующие для реализации поставленной задачи колоссальных трудозатрат, что ведет к повышению материальных расходов предприятия.

В результате проведенного анализа в совокупности выявлены следующие недостатки текущего организационного решения:

– повышенная загруженность уровня распределения задач;

– повышенная загруженность уровня исполнения.

1.2.3 Формулировка предложения

В связи с этим для вышеперечисленных недостатков сформировано предложение автоматизации процесса организации событий. На организационном уровне маршруты и подразделения, находящиеся в зоне избыточности, больше не должны участвовать в рассматриваемом процессе.

Директор напрямую отправляет заявку на проведение события в автоматизированную систему, которая проводит публикацию события на разработанном корпоративном ресурсе, а также рассылку информации приглашенным лицам с использованием доступных современных каналов доставки информации (SMS – сообщения, мессенджеры, E-mail) [11].

1.2.4 Формулировка технического задания

Полное название разрабатываемой системы – модуль автоматизации организации событий средствами Интернет.

Основанием для разработки является договор на разработку программного обеспечения №224 от 19.10.17 г.

Заказчиком является представительство компании «Pegasystems inc.» в России. Исполнителем – сотрудник ООО «Технологии Надежности» Скубников Кирилл Сергеевич.

Сроки начала и окончания выполнения работ: 20.10.17 г. – 20.05.18 г.

Описание объекта управления, а также мнемосхема автоматизируемого бизнес-процесса описаны в главе 1.2.1. Процесс организации событий включает в себя подпроцессы публикации информации о событии в интернет-ресурсах, а также рассылки приглашений персонам. Также предусмотрена автономная регистрация пользователей, реализованная на данный момент через проприетарные инструменты.

Основными целями разрабатываемого модуля является устранение выявленных недостатков, а также значительная реорганизация и реструктуризация процесса организации событий. На данный момент механизмов автоматизации не используется. Целесообразность обусловлена излишней сложностью выполнения процесса. Программное обеспечение подразумевает замещение избыточных структурных компонентов.

Разрабатываемый программный модуль должен обеспечивать и реализовывать:

- разграничение прав доступа;
- принцип распределения компонентов системы;
- масштабируемость системы в целом;
- возможность создавать и публиковать события;
- возможность производить рассылку по выбранным каналам коммуникации;
- возможность объединять пользователей в структурные единицы – группы;
- возможность приглашения выбранных пользователей на событие;
- возможность приглашения персоны по имеющимся контактам на событие.

Программное обеспечение должно быть выполнено в виде веб-приложения с максимально гибкой, расширяемой архитектурой.

Работать с системой должен сам директор либо любой другой доверенный или назначенный человек, группа лиц.

Разграничение прав пользования системой должно однозначно определять доступные функциональные блоки текущему пользователю.

Входные данные системы представляют из себя набор сведений, подробного описания, дат и места проведения события, а также список лиц, приглашенных на участие.

Выходными данными системы являются опубликованная информация о событии на корпоративном портале, а также разосланные приглашения.

Разработка модуля системы автоматизации организации событий включает в себя следующие запланированные работы:

а) техническое совещание;

б) совещание с исполнителями;

в) обработка исходных данных;

г) разворачивание рабочей среды;

д) разворачивание продукционной среды;

е) черновая итерация цикла разработки;

ж) демонстрация концептуальной версии приложения;

з) выполнение рабочего проекта;

и) конфигурация среды разворачивания проекта;

к) тестирование разработанного программного продукта;

л) обучение персонала;

м) пуск программного обеспечения в эксплуатацию.

С точки зрения временных затрат заказчик придерживается гибкой концепции, регламентирующей адаптивное распределение выделенного времени на проект.

В связи с этим каждый этап не имеет строгих временных рамок, а возможность online-коммуникации с заказчиком позволяет производить сдачу каждого этапа непосредственно в момент исполнения.

Таким образом в результате проведенной работы было разработано техническое задание, которое обеспечит стабильное течение процесса разработки в дальнейшем.

Также заказчик намерен продолжать сотрудничество с исполнителями для дальнейшего внедрения новых функциональных модулей, поддержки текущего решения, и реализации постепенного внедрения разрабатываемого программного продукта в инфраструктуру предприятия с целью автоматизации рутинных процессов.

1.3 Построение модели оптимизированного бизнес-процесса организации событий

После проведенного анализа текущего порядка реализации процесса организации событий необходимо выполнить построение диаграмм, отражающих вносимые коррективы, сформулированные в разделе 1.2.

Проведение данной процедуры позволяет предоставить заказчику наглядную демонстрацию реструктуризации и оптимизации бизнес-процессов компании.

Построение модели оптимизированного бизнес-процесса организации событий, как и диаграммы анализа текущего течения реализации рассматриваемой задачи, проводится в нотации IDEF0. Подобная мера позволяет сохранить единообразие разрабатываемых аналитических материалов с целью дальнейшей оптимизации и адаптации наработок под изменяемые условия организационной структуры в целом.

Контекстная диаграмма бизнес-процесса организации событий «Как должно быть» показана на рисунке 1.7.

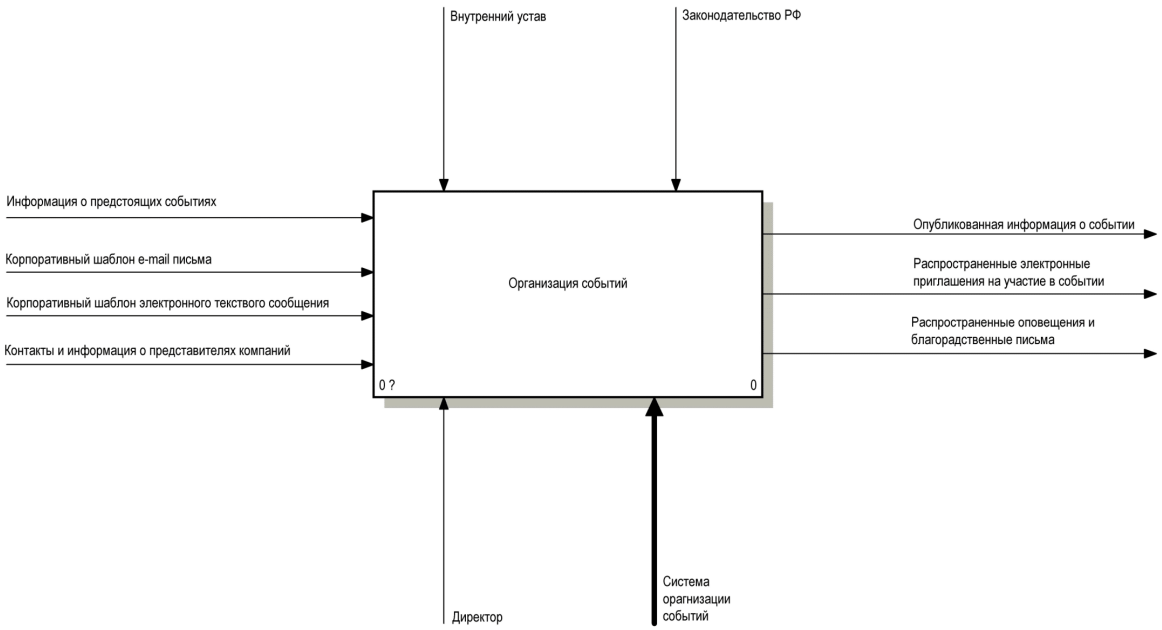

Рисунок 1.7 – Контекстная диаграмма процесса организации событий

Основным направлением оптимизации выбран подход перераспределения использования человеческих ресурсов и автоматизации избыточной рутинной деятельности с учетом пожеланий заказчика.

На представленной диаграмме (рисунок 1.7) можно наблюдать изменение перечня задействованных механизмов в деятельности по организации событий. Очевидным является факт уменьшения использования человеческого труда в рутинных операциях, легко поддающихся полной либо частичной автоматизации.

Среди новых сущностей можно определить систему организации событий, реализующую автоматизированную платформу выполнения процессов, связанных с формированием, оформлением данных, сообщений, а также агрегацией контактов и каналов распространения информации.

Под рассматриваемым понятием понимается разрабатываемый модуль, реализуемый в рамках проекта.

Для более полного восприятия внесенных изменений необходимо проанализировать диаграмму декомпозиции высшего уровня иерархии процессов, отображенной на рисунке 1.8.

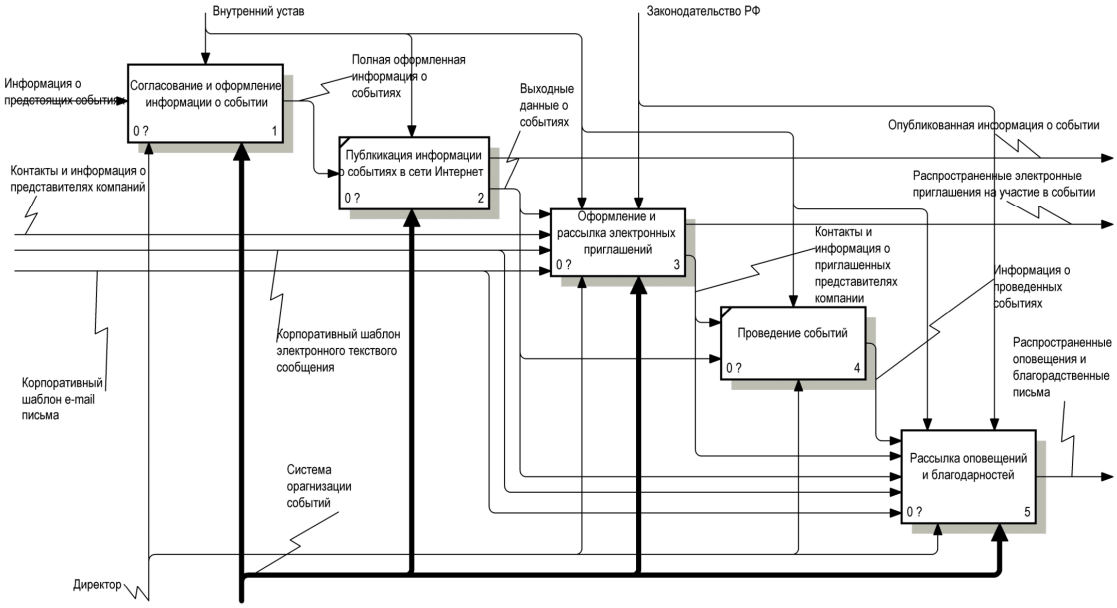

Рисунок 1.8 – Декомпозиция контекстной диаграммы

На рассматриваемой диаграмме наблюдается оптимизация использования человеческих ресурсов практически во всех функциональных блоках бизнес-процесса путем внедрения разрабатываемой автоматизированной системы.

Последующие диаграммы декомпозиции подпроцессов отражены в приложении А. Так, на рисунке А.1 подробно отражена оптимизированная структура процесса оформления и рассылки электронных приглашений. Основные изменения в течении функции рассылки электронных уведомлений и оповещений отражены на рисунке А.2. Согласование и оформление информации о событии также подверглось изменениям в работе (рисунок А.3).

Дальнейшее частное рассмотрение внесенных изменений не имеет смысла ввиду сохранения структуры реализации рассматриваемых процессов. Основные изменения заключаются в замещении избыточного использования человеческих ресурсов и их последующего перенаправления на более приоритетные задачи.

После проведенной работы можно целостно оценить вносимые изменения в структуры ведения процессов внутри компании.

1.4 Обзор существующих решений

В настоящее время существуют целые классы продуктов, частично реализующих требуемый функционал. Одним из таких классов являются почтовые клиенты [11].

Рассмотрение существующих способов решения задач проводится применимо к классам программного обеспечения. Детальный анализ конкретных реализации не имеет смысла ввиду общих недостатков и принципиальных ограничений.

Почтовые клиенты выполняют роль посредника работы с почтовыми серверами, предоставляющие интуитивно понятный процесс работы, а также, чаще всего, графический интерфейс. Из плюсов к решениям подобного рода можно отнести абсолютную доступность и простоту пользования, но данные преимущества формируют и основной недостаток – использование единственного канала распространения информации.

Также актуальными средствами распространения информации являются социальные сети. Естественное использование подобных каналов влечет за собой привлечение дополнительных человеческих ресурсов. Также рассматриваемый класс решений предоставляет программный интерфейс взаимодействия, требующий для автоматизированного использования разработку дополнительного программного обеспечения.

Наиболее популярным средством является мобильная телефония, а точнее SMS сообщения. Использование данного канала связи также требует наличия дополнительных человеческих ресурсов. Существуют услуги, доступные юридическим лицам, в спектр которых входит предоставление программного интерфейса работы, который, как и в предыдущем примере, требует разработку дополнительного ПО [12].

Существуют и системы агрегации и многопоточного использования вышеперечисленных каналов доставки сообщений. К сожалению, данные средства не подходят для использования в корпоративных продуктах из-за

частого нарушения пользовательского соглашений внедренных путей доставки сообщений.

На основе проведенного анализа можно однозначно сказать, что комплексного решения поставленных задач с учетом всех требований отсутствует. Данный факт подтверждает актуальность проводимой разработки.

Выводы по первому разделу.

Проведя всесторонний анализ текущего устройства бизнес-процесса организации событий, можно однозначно установить и подтвердить актуальность и востребованность реализуемого проекта.

В ходе проведенной работы было обеспечено комплексное аналитическое обеспечение решаемой задачи, в ходе которого были проведены:

– анализ текущей организации осуществление процесса организации событий;

– разработка технического задания с предварительным выявлением технологических зон автоматизации и вынесением предложения согласно представленным требованиям;

– реализация графического представления внесенных коррективов в бизнес-процесс компании путем построения диаграмм в нотации IDEF0;

– анализ текущих средств решения поставленной задачи.

2 Инфологическое обеспечение задачи

После проведенного анализа необходимо осуществить выбор проектных решений, удовлетворяющих сформированным требованиям в разделе 1. Также важно произвести проектирование структуры разрабатываемого программного обеспечения и используемых баз данных.

2.1 Обоснование и выбор проектных решений

Для дальнейшей программной реализации проекта необходимо выбрать подходящие архитектурные решения среди доступных.

Наиболее популярными подходами построения клиент-серверных приложений являются монолитная и микросервисная архитектуры [13].

Монолитная архитектура подразумевает обобщение всех действующих компонентов системы в единый функциональный модуль. Такой подход сочетает в себе ряд плюсов:

- стабильность скорости разработки;
- оптимальное использование ресурсов вычислительных машин;
- связанность компонентов системы, упрощающая внутреннюю коммуникацию.

Данное решение является оптимальным для разработки небольших программных продуктов с однозначной точкой завершения цикла разработки [14]. Но описанные выше достоинства монолитной архитектуры порождают ряд ключевых недостатков подобного подхода:

- наличие жесткой связности компонентов системы часто является избыточность и может способствовать усложнению разработки программного продукта;
- отсутствие возможности выполнять распределенные операции;

– исключительная затрудненность масштабирования функционала системы;

– формирование массивного узкого места системы, крах которого делает невозможным функционирование всех компонентов системы.

Основное требование, существенно ограничивающее структурную организацию ПО, является принцип распределения компонентов системы. Поэтому монолитная архитектура не позволит реализовать поставленные требования.

Другим популярным стилем построения систем является микросервисный подход. Ключевая особенность такого решения определяется общей концепцией инкапсуляции компонентов системы в отдельные функциональные блоки [15]. Подобная архитектура имеет следующие достоинства:

– потенциал для масштабирования функционала системы;

– отсутствие жестких зависимостей между компонентами системы;

– выход из строя отдельного компонента системы не ограничивает доступ к остальному функционалу.

Основной проблемой микросервисной архитектуры является высокая сложность реализации основных компонентов системы и их интерфейсов программного сетевого взаимодействия.

В ходе проведенного обзора двух популярных архитектурных стилей построения приложений были выявлены основные достоинства и недостатки, формирующие последующий выбор. Следующим этапом инфологического обеспечения решения задачи является выбор программных средств.

2.2 Выбор программных средств реализации проекта

Одним из наиболее влиятельных факторов, определяющих течение и скорость разработки, является набор используемых технологий реализации.

Используемые программные средства реализации проекта должны удовлетворять требованиям, сформированным с учетом архитектуры и функциональных особенностей архитектурных решений.

Грамотно подобранные средства позволят увеличить продуктивность и избежать нежелательных рутинных операций.

В связи с этим в качестве языка программирования выбран Java. Данный ЯВУ является кроссплатформенным, строго типизированным и мультипарадигмальным с преобладанием объектно-ориентированной методологии. Основные особенности рассматриваемого языка программирования позволяют вести параллельную разработку функциональных блоков приложения независимо от используемой системной архитектуры, а также обеспечивает высокую надежность в режиме работы при помощи встроенных средств профилирования и контроля используемых ресурсов. Java обладает широким набором реализованных паттернов и подходов программирования, однако для ускорения разработки принято использовать фреймворки – надстройки над оригинальным языком, имплементирующие массу дополнительного специфического функционала [16].

Самым популярным из подобного класса средств является Spring Framework – универсальный фреймворк, используемый для разработки корпоративных продуктов, отличающихся надежностью и высоким потенциалом масштабируемости приложения. Также встроенные механизмы работы позволяют в полной мере использовать и реализовывать особенности работы и сообщения распределенных систем [17].

В качестве средства управления базами данных в проекте используется PostgreSQL, в полном объеме удовлетворяющая всем современным запросам распределенных приложений.

Описанные выше средства относятся исключительно к реализации серверных приложений микросервисов и баз данных, связанных с ними.

Подходящим средством разработки клиентских приложений является платформа разработки масштабируемых веб-приложений. Ключевой особенностью Angular является активное использование концепции компонентов в связке с шаблоном проектирования «Модель-Представление-Контроллер». Такой подход позволяет вынести и инкапсулировать утилитарные части программной реализации с целью дальнейшего повторного использования. Также в пакете поставки платформы включены средства для разработки прогрессивных и кроссплатформенных приложений, что позволяет использовать одну реализацию в качестве веб и настольных приложений для различных операционных систем [18].

Для удобства работы с описанными выше программными средствами реализации проекта необходим универсальный, гибкий инструмент, позволяющий максимально ускорить процесс разработки ввиду принятой микросервисной архитектуры приложения [19].

В настоящее время таким средством является популярная среда разработки от компании JetBrains – IntellijIDEA, пользовательский интерфейс которой показан на рисунке 2.1.

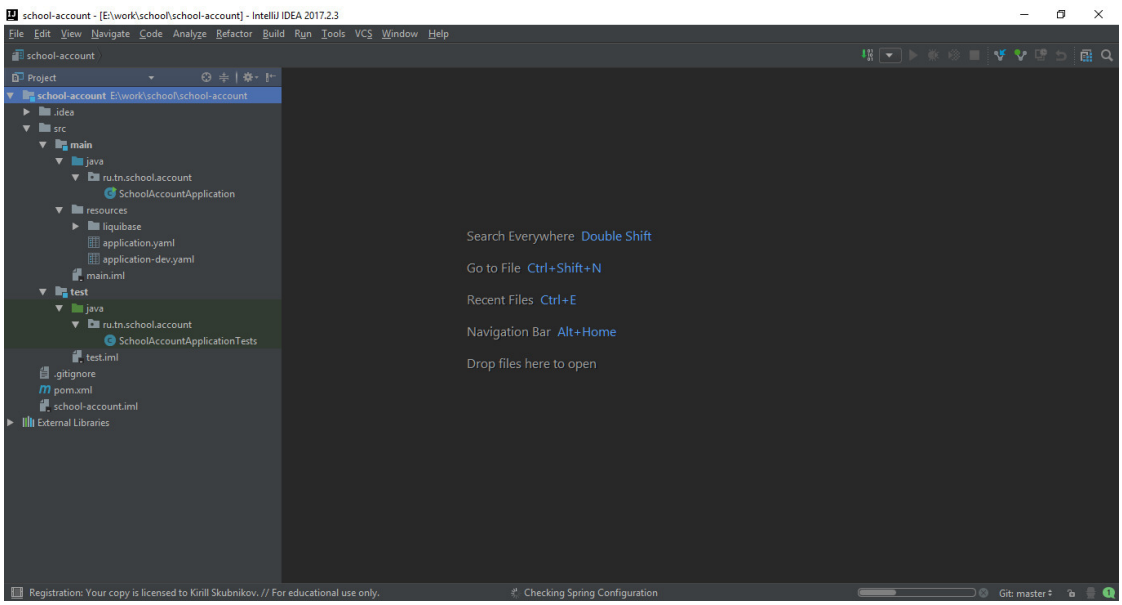

Рисунок 2.1 – Пользовательский интерфейс IntellijIDEA

Данная среда разработки позволяет работать со всеми средствами, описанными выше, а также позволяет напрямую создавать подключение к базе данных без использования внешних программных оболочек, повышающих нагрузку на рабочую станцию.

Для отладки взаимодействия узлов приложения используется Postman – популярный пакет инструментов для тестирования программных интерфейсов. Данный продукт распространяется по свободной модели с открытыми исходными кодами и позволяет инициировать обмен сообщениями с серверными приложениями используя HTTP – протокол передачи данных [20].

Клиентское приложение работает в любом современном веб-браузере, следовательно, для полноценного тестирования разработанного пользовательского интерфейса, необходимо его наличие. В данном случае приоритетным является Opera – браузер, основанный на движке WebKit и обеспечивает полную совместимость с другими популярными продуктами, предоставляя при этом высокое быстродействие и возможность интеграции прогрессивных веб приложений непосредственно в пользовательский интерфейс рабочего пространства.

Выбранные средства позволят повысить комфорт, скорость и стабильность как самого процесса разработки, так и финального программного продукта.

2.3 Проектирование концептуальной архитектурной схемы

Основным этапом разработки распределенного приложения, построенного на основе концепций микросервисного подхода, является проектирование схемы взаимодействия компонентов программного продукта. Данный шаг позволит наиболее четко инкапсулировать бизнес-логику внутри однозначных, четко структурированных узлах системы. Также подобная визуализация предоставляет возможность определить все возможные зависимости между функциональными блоками системы и избежать

избыточной жесткой связи для поддержания возможности беспроблемной поддержки разработанного решения и предупредить возможные проблемы функционального масштабирования продукта.

Схема для проекта автоматизации процесса организации событий средствами Интернет показан на рисунке 2.2.

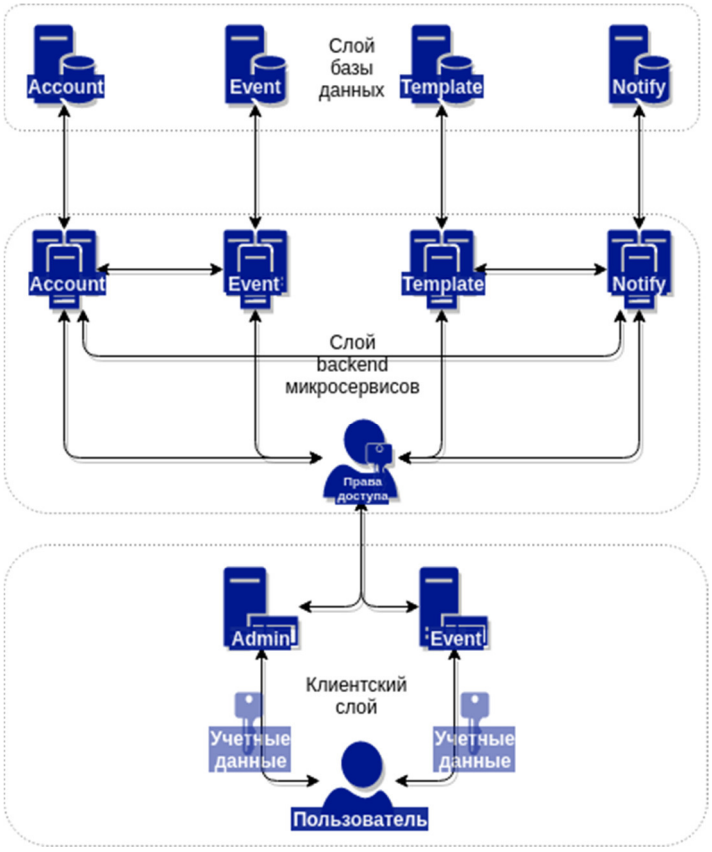

Рисунок 2.2 – Концептуальная архитектурная схема взаимодействия компонентов разрабатываемой системы

На рассматриваемой схеме (рисунок 2.2) отчетливо ограничены концептуальные слои приложения:

– слой баз данных, описывающий кластер серверов хранилищ данных;

– слой серверных микросервисов является суперузлом приложения, содержащим в себе основную бизнес-логику и препроцессоры данных;

– клиентский слой, представляющий совокупность интерактивных пользовательских интерфейсов площадки размещения информации о предстоящих событиях и узла администрирования системой.

Отдельно стоит рассмотреть слой серверных микросервисов:

– узел «Account» отвечает за манипуляции над учетными записями системы, а также включает в себя систему безопасности, используемую остальными компонентами слоя;

– узел «Event» включает в себя механизмы работы с информацией о событиях;

– узел «Template» производит работу по формированию, хранению и автоматизированному заполнению корпоративных шаблонов писем;

– узел «Notify» реализует функциональную имплементацию работы с программными интерфейсами задействованных каналов распространения информации.

Дальнейшие компоненты системы не требуют отдельного рассмотрения ввиду своей очевидной атомарности и однозначности. В связи с этим проектирование и обзор концептуальной архитектурной схемы можно считать оконченным.

2.4 Проектирование баз данных

Обеспечение надежного хранилища данных приложения крайне ответственная задача, требующая детальной проработки. Качественное проведенное проектирование баз данных поможет достичь требуемого быстродействия системы и избежать избыточного использования вычислительных ресурсов.

Дальнейшее рассмотрение устройства схем баз данных производится в рамках конкретного компонента системы, поскольку разрабатываемая система основана на микросервисной архитектуре.

Схема базы данных микросервиса управления пользовательскими учетными записями показана на рисунке 2.3.

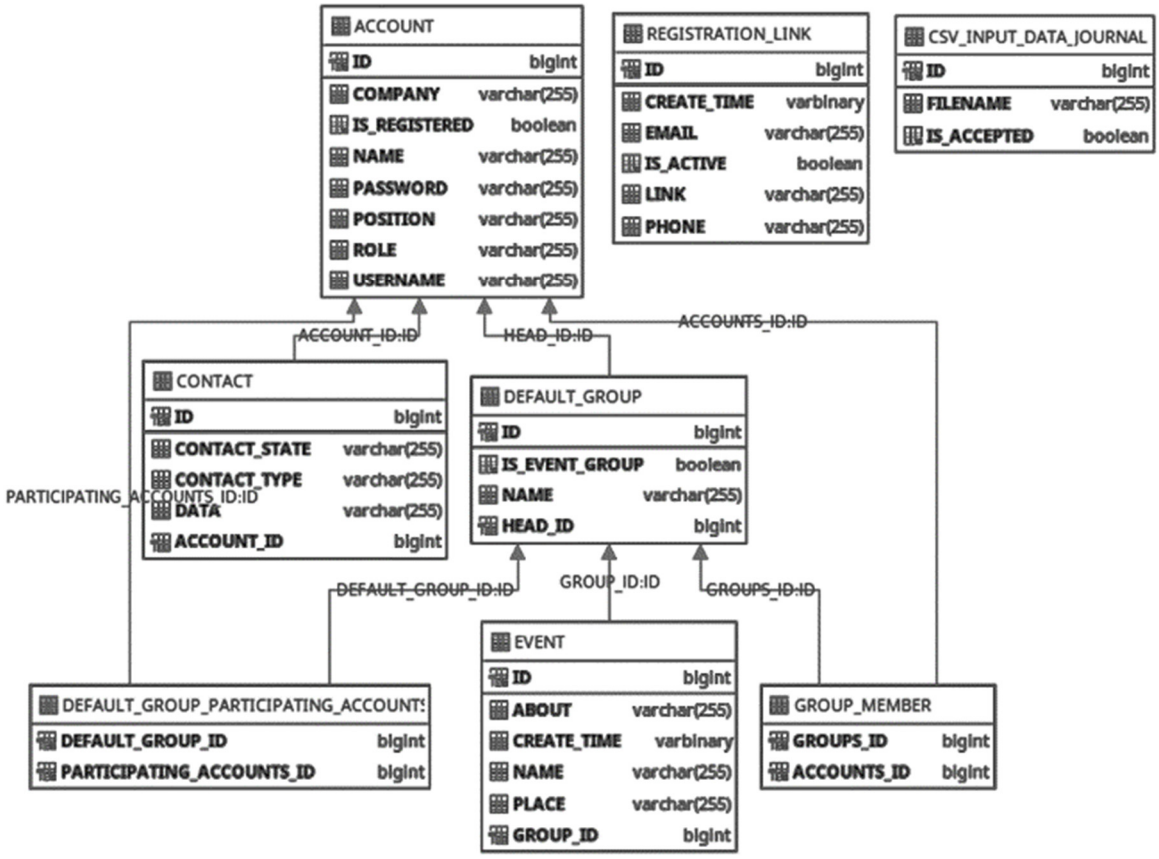

Рисунок 2.3 – Схема баз данных сервиса учетных записей

В рассматриваемой схеме представлены основные базовые сущности разрабатываемого модуля. Необходимо более подробно рассмотреть каждую из них.

«Account» – сущность, хранящая в себе следующую основную информацию об учетной записи с учетом всех действующих законодательных актов, регламентирующий работу и запись данных пользователей:

– место работы;
– занимаемая должность;
– ФИО;
– пароль;

– имя пользователя;

– роль в системе.

«Contact» – таблица, описывающая контактную информацию;

«Registration_link» представляет набор записей о пользователях, которым были высланы приглашения для регистрации в системе.

«CSV_Import_Journal» – журнал импорта записей из сторонних почтовых клиентов в формате CSV.

Рассмотрение служебных таблиц, не задействованных напрямую в бизнес-процессах, опущено ввиду избыточности в рамках обзора схемы баз данных.

За работу с шаблонами писем отвечает сервис шаблонизации, схема баз данных которого показана на рисунке 2.4.

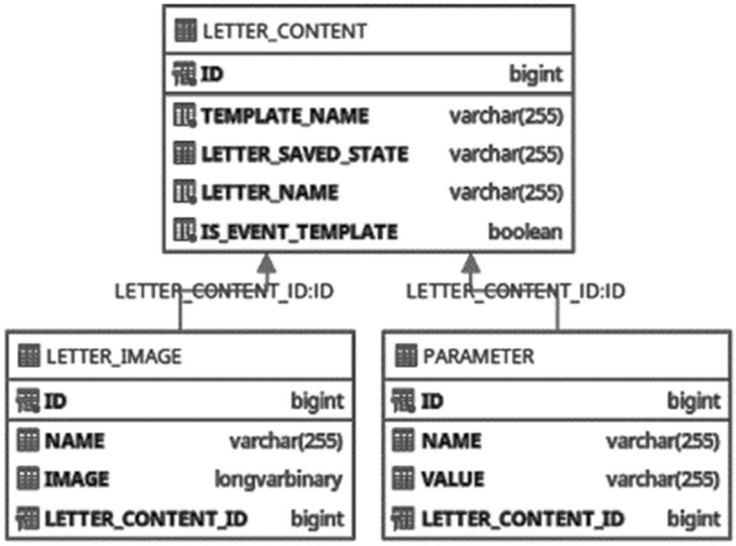

Рисунок 2.4 – Схема баз данных сервиса шаблонов

Сущности, представленные на рассматриваемой схеме, описывают шаблон письма, содержащий в себе название и даты последних манипуляций, а также сопроводительные данные в виде изображений и абстрактных параметров.

Далее рассмотрим схему баз данных сервиса событий (рисунок 2.5).

Рисунок 2.5 – Схема баз данных сервиса событий

Сущность, описывающая событие, содержит в себе название, полное и краткое описание, даты начала и окончания, место проведения и заголовочное изображение. Связывание участников события происходит через идентифицирующее поле «Id_Group», обеспечивающее отражение сущности группы сервиса управления учетными записями.

Рассмотрим схему баз данных сервиса уведомлений, продемонстрированную на рисунке 2.6.

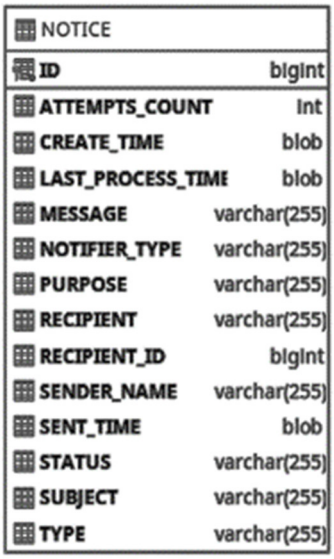

Рисунок 2.6 – Схема баз данных сервиса уведомлений

Данная таблица содержит в себе множество сервисных полей, идентифицирующих состояние отправления уведомления, тип сообщения и

используемый канал обмена информацией, текст и заголовок, а также перечень получателей.

Подобная структура обусловлена высоким уровнем абстракции уровня программного интерфейса, а также внутренними сложностями агрегации и многоканального взаимодействия с сервисами распространения информации.

В качестве результирующей схемы необходимо провести разработку общей схемы баз данных, представленной на рисунке 2.7. Пунктирные линии обозначают границы инкапсуляции рассмотренных выше компонентов системы.

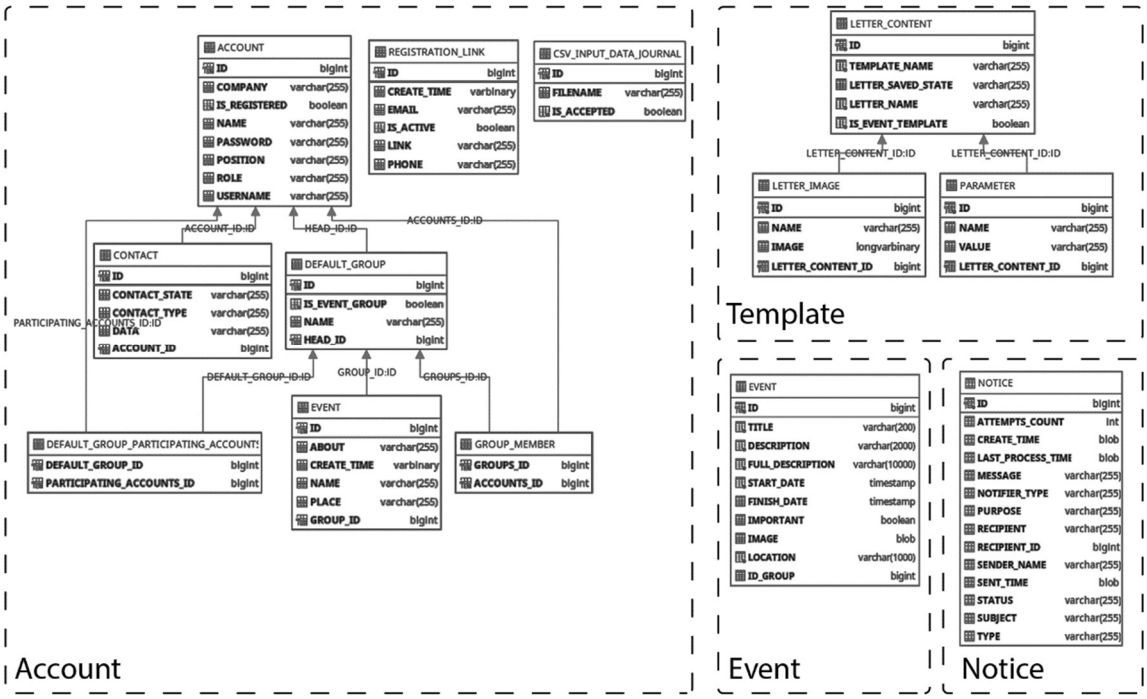

Рисунок 2.7 – Схема баз данных микросервисов

Представление схем баз данных всех микросервисов в целом позволяет проверить систему на наличия факта дублирования и избыточного использования информации.

На данном этапе проектирование схем баз данных считается законченным. Физическая реализация проводится путем использования встроенных механизмов выбранных программных средств проекта.

Выводы по второму разделу

В ходе проведенной работы по инфологическому обеспечению проекта работы произведено:

- обоснование и выбор программных средств реализации проекта;
- проектирование архитектуры приложения;
- проектирование баз данных.

Выполнение перечисленных работ позволяет приступить непосредственно к программной реализации проекта.

Подготовленное документационное сопровождение позволяет наиболее полно охватить и предусмотреть функциональные и архитектурные решения.

Разработанные схемы баз данных и диаграммы взаимодействия компонентов системы потенциально могут служить сопроводительной документацией к реализуемому программному продукту для обеспечения достаточного уровня комфорта сопровождения и расширения системы.

3 Программная реализация проекта

После проведенных этапов по аналитическому и инфологическому обеспечению проекта необходимо разработать программную реализацию рассмотренных ранее задач.

Рассмотрение подходов и методов реализации проводится послойно согласно концептуальной архитектурной схемы [21]. Особенности имплементации рассматриваются в рамках типовых примеров компонентов системы и их основных элементов. Подробный обзор программной реализации не имеет смысла ввиду однотипности решений и преобладании сопровождающей системной логики.

3.1 Программная реализация серверного приложения

Реализацию клиент-серверных приложений принято начинать с формирования основной бизнес-логики на стороне мейнфрейма. Рассмотрение будет проводится на примере микросервиса рассылки сообщений. На рисунке 3.1 показан фрагмент конфигурации проведения запланированных задач по рассылке информации.

```
@Slf4j
@Configuration
public class JobRunnerConfiguration implements SchedulingConfigurer {
    private final Job notifierJob;
    private final JobLauncher jobLauncher;
    private final JobProperties jobProperties;

    @Autowired
    public JobRunnerConfiguration(Job notifierJob, JobLauncher jobLauncher, JobProperties jo
        this.notifierJob = notifierJob;
        this.jobLauncher = jobLauncher;
        this.jobProperties = jobProperties;
    }


    @Bean
    Executor taskScheduleExecutor() {
        return Executors.newScheduledThreadPool( corePoolSize: 10, new CustomizableThreadFactory
    }

    @Override
    public void configureTasks(ScheduledTaskRegistrar scheduledTaskRegistrar) {
        Runnable task = () -> {
            try {
                JobParametersBuilder builder = new JobParametersBuilder();
                builder.addString( key: "jobName", notifierJob.getName());
                builder.addLong( key: "jobStartDate", System.currentTimeMillis());
                jobLauncher.run(notifierJob, builder.toJobParameters());
```

Рисунок 3.1 – Конфигурация планировщика задач

На данном этапе производится связывание служб планирования выполнения задач с службами выполнения сервисных функций и сущностями проекта [22].

Для работы с базовыми понятиями разработанной схемы баз данных проекта в фреймворке Spring необходимо провести связывание таблиц с сущностями Java проекта. Пример такой реализации показан на рисунке 3.2 [23].

```
@Data
@Entity
@EntityListeners(AuditingEntityListener.class)
@ToString
public class Notice {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "notice_seq")
    @SequenceGenerator(sequenceName = "notice_seq", allocationSize = 1, name = "notice_seq")
    private Long id;
    private String senderName;

    @Enumerated(EnumType.STRING)
    private NotifierTypeEnum notifierType;

    @Enumerated(EnumType.STRING)
    private NoticeStatusEnum status;
    private String subject;
    private String message;

    @CreatedDate
    @Temporal(TemporalType.TIMESTAMP)
    private Date createTime;

    @Temporal(TemporalType.TIMESTAMP)
    private Date sentTime;
```

Рисунок 3.2 – Связывание сущности с таблицей базы данных

Механизм скрепления сущности подразумевает полное описание таблицы базы данных с применением конфигурационных аннотаций объектного модуля фреймворка для установления ограничений в перечисляемых полях либо указания специфики их поведения [24].

Последующая работа с объектами проводится при помощи репозиториев, образец реализации которого показан на рисунке 3.3.

```
@CrossOrigin
@RepositoryRestResource
public interface NoticeRepository extends JpaRepository<Notice, Long> {
    List<Notice> findByIdIn(@Param("ids") List<Long> ids);
}
```

Рисунок 3.3 – Реализация репозитория

Встроенные реализации типовых моделей репозиториев в Spring позволяют быстро реализовать доступ к основным функциям манипулирования физическими сущностями базы данных, а также обеспечивает интерфейс формирования, структурирования и расширения способов фильтрации выдачи запросов.

Для реализации доступа к механизмам манипулирования данными средствами интернет необходимо реализовать сервисы – промежуточный программный слой, реализующий основную бизнес-логику приложения и интерфейс взаимодействия с репозиториями [25]. Имплементация такого компонента показана на рисунке 3.4.

```
@Service
public class NoticeService {
    @Autowired
    private NoticeRepository noticeRepository;
    @Autowired
    private RecipientRepository recipientRepository;

    public List<Notice> getAll () { return noticeRepository.findAll(); }

    public Notice getById (Long id) { return noticeRepository.findOne(id); }

    public Notice add (Notice notice) {
        Iterable<Recipient> recipients = recipientRepository.save(notice.getRecipients());
        notice.setRecipients((List<Recipient>) recipients);
        return noticeRepository.save(notice);
    }
```

Рисунок 3.4 – Программная реализация сервиса

Сетевое взаимодействие между компонентами системы производится посредством обмена HTTP сообщениями с использованием архитектурного стиля передачи состояния представления взаимодействия компонентов распределенного приложения.

Подобный стиль активно поддерживается Spring и используется с учетом ограничений, позволяющих вести динамическую навигацию между ресурсами сервиса при помощи вложенных гиперссылок [26].

Основным приемом реализации сетевого программного интерфейса приложения в Spring является создание контроллеров, связывающих входящие сетевые запросы, производя их обработку и предварительный анализ, и

основную бизнес-логику. Демонстрация имплементации такого подхода показан на рисунке 3.5.

```
@CrossOrigin
@RestController
public class NoticeRestController {
    @Autowired
    private NoticeService noticeService;

    @GetMapping("/notice")
    public List<Notice> getAll () { return noticeService.getAll(); }

    @GetMapping("/notice/one/{id}")
    public Notice getById (@PathVariable("id") Long id) { return noticeService.getById(id); }

    @PostMapping("/notice")
    public Notice add (@RequestBody Notice notice) { return noticeService.add(notice); }

    @PutMapping("/notice")
    public Notice update (@RequestBody Notice notice) { return noticeService.update(notice); }

    @DeleteMapping("/notice/{id}")
    public ResponseEntity delete (@PathVariable("id") Long id) {
        noticeService.delete(id);
        return ResponseEntity.ok().build();
    }
```

Рисунок 3.5 – Программная реализация контроллера

После проведенной работы разработанный микросервис готов к дальнейшему сетевому взаимодействию и композиции с остальными компонентами системы, в том числе и с клиентским слоем.

3.2 Программная реализация клиентского приложения

Выбранная платформа для программной реализации клиентского приложения Angular использует компонентный подход.

Рассмотрение примера производится на основе приложения площадки публикации и регистрации событий.

Компонент – часть структуры программы, реализующая часто используемые фрагменты логики или интерфейса, обладающая гибкостью и изменяемостью. Композиция подобных элементов составляет дерево объектов системы, используя которое движок отображение содержимого Angular производит процессинг и контроль состояния приложения [27].

Реализация компонента приложения отражена на рисунке 3.6

```
@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.css']
})
export class CardComponent implements OnInit {

  @Input() private type: string;
  @Input() private title: string;
  @Input() private subTitle: string;
  @Input() private imageBase64: string;
  @Input() private description: string;
  @Input() private cardHeader: string;
  @Input() private detailComponentUrl: string;
  @Input() private detailComponentUrlEventId: number;
  @Input() private detailComponentUrlGroupId: number;
  @Input() private showButton: boolean;
  @Input() private buttonCaption: string;

  @Output() private btnclick = new EventEmitter();

  constructor() {
  }

  ngOnInit() {
  }
```

Рисунок 3.6 – Компонент карточки

Для базовой работы с серверным приложением необходимо реализовать модели сущностей, используемых при обмене сетевыми запросами [28]. Пример имплементации показан на рисунке 3.7.

```
export class Event {
  id: number;
  title: string;
  description: string;
  fullDescription: string;
  location: string;
  important: boolean;
  image: string;
  startDate: Date;
  finishDate: Date;
  idGroup: number;
  _links: {
    event: {
      href: string;
    },
    self: {
      href: string;
    }
  };
```

Рисунок 3.7 – Модель сущности события

Такой подход позволяет обеспечить консистентность данных путем обеспечения единообразия связывания таблиц баз данных и сущностей программной реализации.

В качестве представления компонента используется документ гипертекстовой разметки. Встроенные возможности проекции содержимого контроллера при помощи интерполяции позволяет наглядно разместить контент внутри шаблона, провести простейшие преобразования и выполнить выражения, отображающие текущее состояние компонента. Реализация графического интерфейса объекта показана на рисунке 3.8.

```
<div class="card-title-wrapper">
      <span>
        {{cardHeader}}
      </span>
</div>
<div *ngIf="showButton" class="btn-wrapper">
  <button class="header-button show-warning" (click)="onButtonClick()">{{buttonCaption}}</button>
</div>
/div>
div class="flex-row">
 <div *ngIf="imageBase64" class="extended-card-block card-image">
   <img [src]="imageBase64 | sanitizeImageResource">
 </div>
 <div [ngClass]="{'extended-card-block': imageBase64}" class="card-description description">
   <div>
```

Рисунок 3.8 – Представление компонента

Как и Spring, Angular использует сервисы для компоновки часто используемого сервисного кода, работающего с сетевыми запросами, с целью дальнейшего многократного применения в других компонентах приложения при помощи механизма внедрения зависимостей и инверсии контроля [29]. Пример реализации такого подхода отражен на рисунке 3.9.

```
@Injectable()
export class AuthService {

  constructor(private http: HttpClient, private storage: GlobalStorageService) {
  }

  public login(username: string, password: string): Observable<boolean> {
    this.removeAuthToken();
    const credentials = btoa( rawString: username + ':' + password);
    const httpOptions = {
      headers: new HttpHeaders({
        'Authorization': 'Basic ' + credentials
      })
    };
    return this.http.get( url: environment.accountUrl + '/auth/', httpOptions)
      .map(response => {
        if (response) {
          this.setAuthToken(credentials);
          return true;
        } else {
          return false;
        }
      });
  }
```

Рисунок 3.9 – Сервис авторизации

Также встроенные в Angular средства перехвата запросов позволяют автоматически добавлять заголовки конфигурации сообщения, содержащие данные для доступа к приватному контенту [30]. Демонстрация реализации авто дополнения заголовков сообщения показана на рисунке 3.10.

```
@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  constructor(private authService: AuthService) {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const token: string = this.authService.getAuthToken();
      if (token) {
        req = req.clone( update: {
          setHeaders: {
            Authorization: 'Basic ' + token
          }
        });
      }
    return next.handle(req);
  }
}
```

Рисунок 3.10 – Программная реализация перехватчика запросов

Подобный подход позволяет перед попаданием ответа в сервис провести анализ ошибок и провести их форматирование для корректного отображения пользователю.

Схожим образом формируются и остальные компоненты клиентского приложения. После проведенной разработки серверного приложения необходимо провести живое тестирование проекта в середе веб-браузера.

3.3 Тестирование разработанной программной реализации

На этапе тестирования открывается возможность побыть пользователем системы, оценить качество выполненных работ, удобства интерфейса и проверить ключевые функциональные блоки приложения на наличие ошибок или непредсказуемого поведения [31].

Начать тестирование лучше с приложения администрирования системой. Основное окно работы показано на рисунке 3.11.

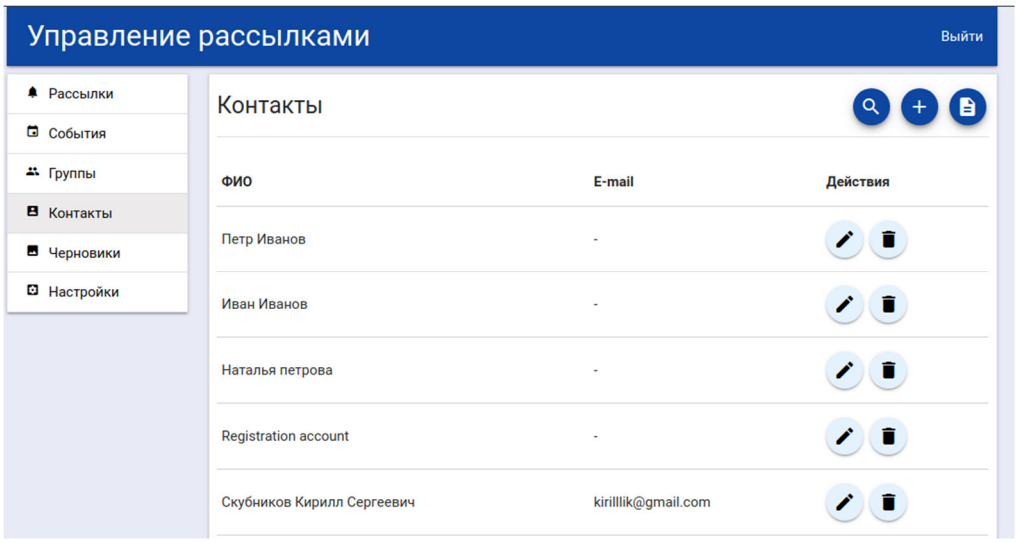

Рисунок 3.11 – Основной интерфейс панели администрирования

Разработанный интерфейс приложения администрирования системы обладает минималистичным дизайном, что позволяет сосредоточить пользователя на выполнении бизнес-задач и обеспечить оптимальное удобство работы. Каждый компонент работы с базовыми сущностями системы обладает функциями фильтрации, поиска и основными операциями манипуляции.

Пример компонента добавления сущности, а именно события, продемонстрирован на рисунке 3.12.

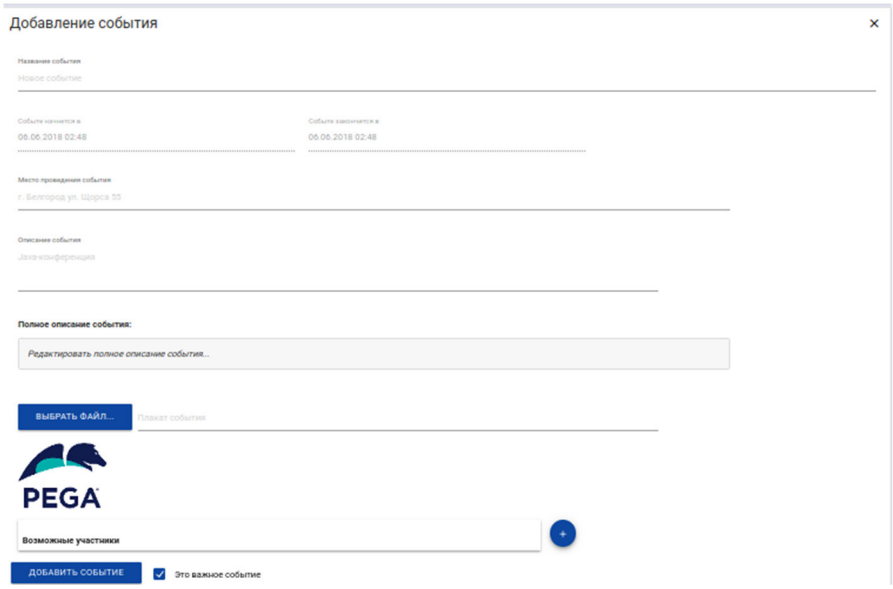

Рисунок 3.12 – Компонент добавления события

На данном этапе имеется возможность заполнить всю необходимую информацию о предстоящем мероприятии, пригласить участников и прикрепить изображение – обложку, отображаемую на главной странице портала публикации. После добавления лиц, приглашенных на участия, проводится рассылка индивидуальных писем (рисунок 3.13).

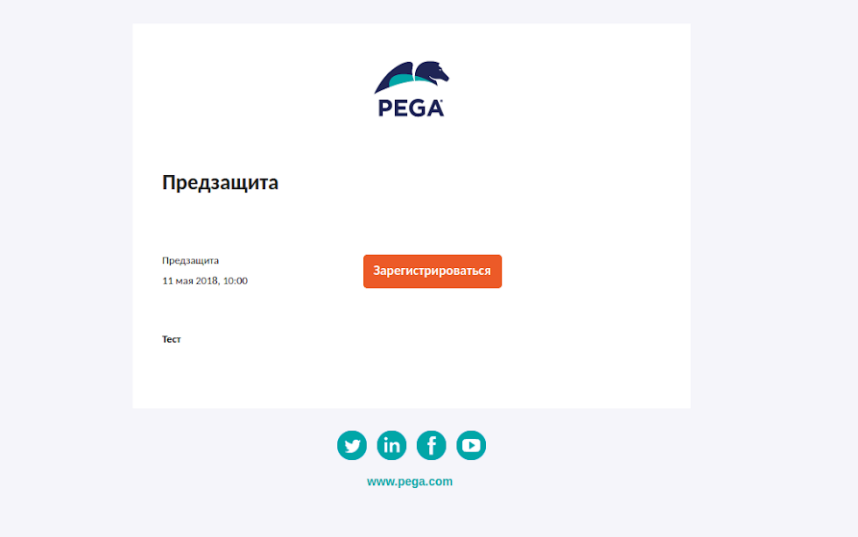

Рисунок 3.13 – Пригласительное письмо

Ссылка, указанная в письме, ведет непосредственно на портал публикации информации о событиях (рисунок 3.14).

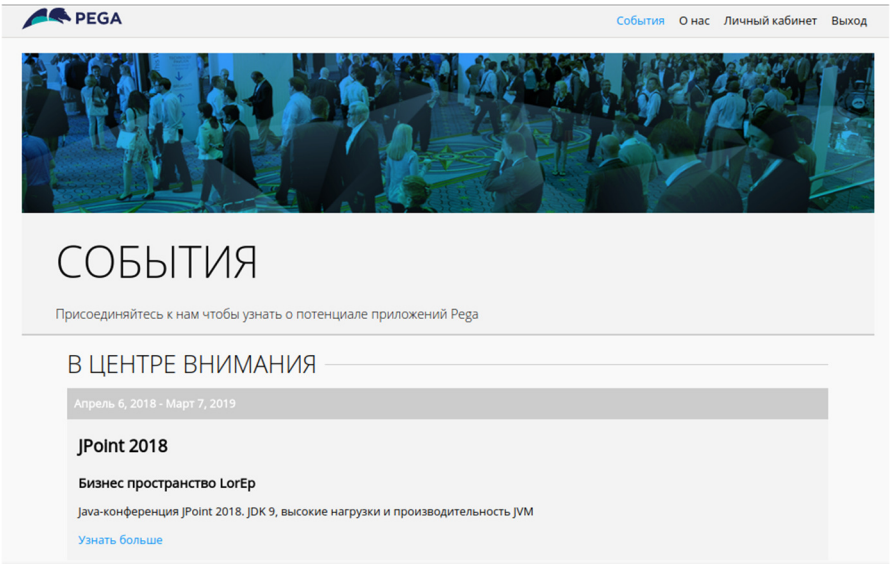

Рисунок 3.14 – Основной интерфейс портала событий

Разработанное приложение портала мероприятий оформлено с учетом корпоративных правил и цветовой схемы «Pegasystems inc.». Основная страница отображает все события, актуальность которых соответствует текущей дате на устройстве пользователя [32].

Просмотр подробной информации и последующая регистрация происходит в компоненте, показанном на рисунке 3.15.

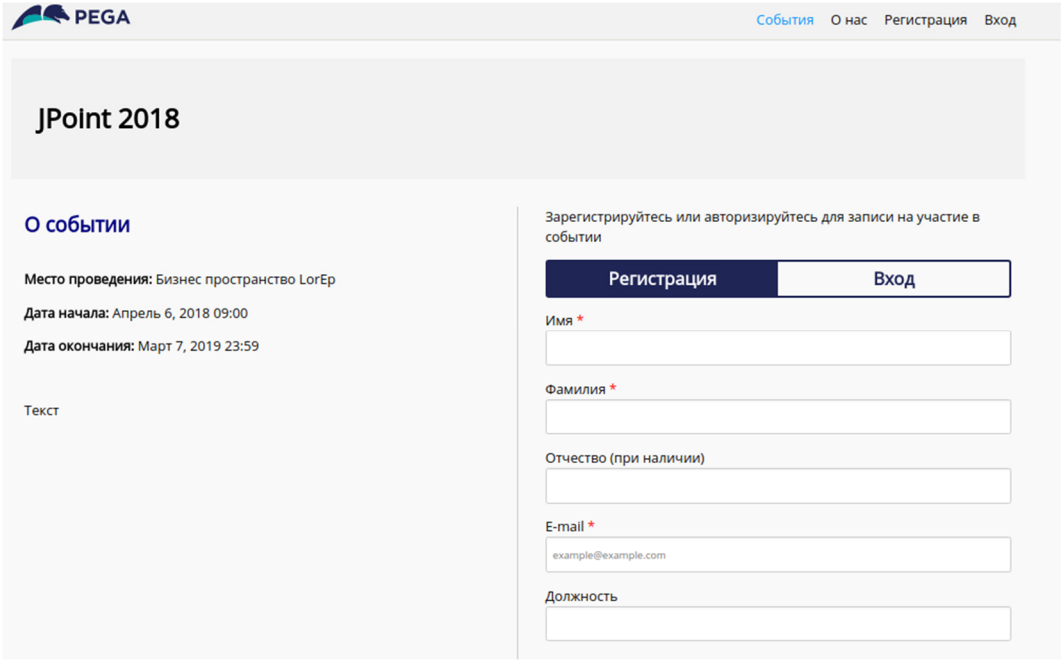

Рисунок 3.15 – Компонент отображения подробной информации о событии

При помощи представленных форм пользователь может зарегистрироваться на участие событии и одновременно завести аккаунт в системе. Также имеется возможность авторизации существующего пользователя на странице мероприятия для мгновенной регистрации. Все поля строго подвержены автоматической модерации и валидации. Такой подход позволяет типизировать вводимые пользователем данные и предотвратить деструктуризацию баз данных.

Пользователь имеет возможность посмотреть события, на которые он зарегистрирован либо был приглашен в личном кабинете (рисунок 3.16)

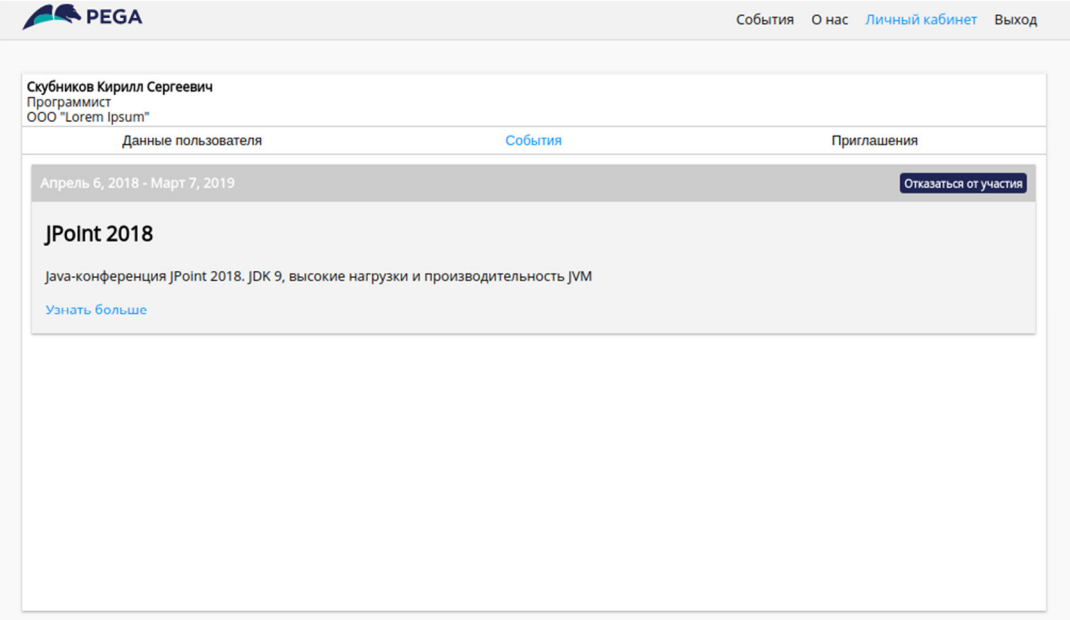

Рисунок 3.16 – Личный кабинет пользователя

В данном разделе системы предусмотрена возможность моментального принятия приглашения и отмены участия. Пользователь может изменить данные учетной записи в соответствующем разделе личного кабинета, что позволяет обеспечить актуализацию информации.

После проведенных манипуляций критических ошибок не было обнаружено и тестирование можно считать успешным.

Выводы по разделу 3

В ходе проведенной работы были разработаны и протестированы все модули системы. Реализация проводилась с учетом всех правил и пожеланий заказчика. Также были решены все поставленные задачи, связанные с программной имплементацией проекта.

По завершению разработки программный продукт передан заказчику с целью внедрения и интеграционного тестирования.

ЗАКЛЮЧЕНИЕ

В ходе реализации выпускной квалификационной работы была проведена автоматизация ключевых бизнес-процессов действующей организации.

Анализ текущего порядка ведения дел обозначил потенциальные участки внедрения разрабатываемого модуля, а составление результативной модели позволяет наглядно увидеть существенную оптимизацию использования человеческих ресурсов. Данный факт позволил эффективно перераспределить задачи между участниками процесса и упразднить ведение рутинных функциональных блоков.

Обзор существующих решений позволил подтвердить актуальность разработки, поскольку имеющиеся средства являются узконаправленными, а доступные механизмы решения поставленной задачи – монополизированными. Также в результате были обнаружено и учтено наличие ограничений и ряда поставщиков каналов передачи информации средствами Интернет.

Грамотное инфологическое обеспечение проекта позволило выполнить разработку распределенного модуля в кратчайшие сроки. Сформированное техническое задание отражает все пожелания заказчика с соблюдением общепринятых стандартов.

Проведенное проектирование архитектуры разрабатываемого программной реализации позволило реализовать сложные и актуальные шаблоны проектирования распределенных систем. Применение прогрессивных подходов структурной организации приложения открывает широкие возможности для дальнейшего масштабирования продукта.

Ввиду специфики разрабатываемый программный продукт работает с инкапсулированными и логически раздробленными фрагментами большой базы данных. Проектирование схем позволило составить наглядное представление комплексной схемы подобного класса.

Разработанный программный продукт отвечает всем современным требованиям к системам подобного класса. Внедрение такого решения позволит улучшить эффективность и скорость ведения маркетинговых компаний с целью увеличения оборота компании и клиентской базы. Также ожидается стабильный рост представительства компании, спровоцированный рядом внесенных оптимизаций в структуру ведения бизнес-процессов.

После внедрения программной реализации в бизнес-процессы компании – заказчика ожидается рост продуктивности как коллективных структур, прежде задействованных в оптимизируемом участке, так и остальных организационных структур в целом.

В итоге все поставленные задачи были решены своевременно и в полном объеме, а цели достигнуты. Разработанное программное обеспечение готовится к внедрению в компанию заказчика с целью оптимизации рассматриваемых на протяжении работы процессов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Развитие интернета в регионах России [Электронный ресурс] / Yandex – Москва: 2016. – Режим доступа: https://yandex.ru/company/researches/2016/ya_internet_regions_2016, свободный.
2. Официальное определение термина Интернет [Электронный ресурс] / Издание Medium – Россия: 2017. – Режим доступа: https://medium.com/bazanov-org/интернет-официальное-определение-термина-e8374398892, свободный
3. Кравец А. Д., Петрова И. Ю., Кравец А. Г. Агрегация информации о перспективных технологиях на основе автоматической генерации интеллектуальных агентов мультиагентных систем //Прикаспийский журнал: управление и высокие технологии. – 2015. – №. 4. – С. 141-148.
4. Ляндау Ю. В., Стасевич Д. И. Автоматизация бизнес-процессов //Russian Journal of Management. – 2014. – Т. 2. – №. 6. – С. 290-302.
5. Саати Т. Л., Кернс К. Аналитическое планирование: организация систем. – Радио и связь, 2013. – С. 224.
6. Методология функционального моделирования IDEF0 [Электронный ресурс] / Новосибирский государственный университет – Россия: 2014. – Режим доступа: https://nsu.ru/smk/files/idef.pdf, свободный.
7. Репин В., Елиферов В. Процессный подход к управлению. Моделирование бизнес-процессов. – Litres, 2017.
8. Ганелина Н., Мамонова В., Мамонова Н. Моделирование бизнес-процессов. – Litres, 2018.
9. Елугачев П. А., Елугачев М. А. Подготовка технического задания в концепции информационного моделирования дорог //САПР и ГИС автомобильных дорог. – 2015. – №. 2 (5).
10. Пыжикова Н. И., Титова Е. В. Комплексная задача организации эффективного бизнес-процесса //Успехи современной науки. – 2016. – №. 1. – С. 21-24.

11. Beasley J. E. OR-Library: distributing test problems by electronic mail //Journal of the operational research society. – 2014. – Т. 41. – №. 11. – С. 1069-1072.

12. Dumais S. et al. Stuff I've seen: a system for personal information retrieval and re-use //ACM SIGIR Forum. – ACM, 2016. – Т. 49. – №. 2. – С. 28-35.

13. Закон Ф. Об информации, информационных технологиях и о защите информации //№ 149-ФЗ. РФ – Режим доступа: http://www.internet-law.ru/law/inflaw/inf.htm. (дата обр. 6.05.18 г.). – 2016.

14. Артамонов Ю. С., Востокин С. В. Разработка распределенных приложений сбора и анализа данных на базе микросервисной архитектуры //Известия Самарского научного центра Российской академии наук. – 2016. – Т. 18. – №. 4-4.

15. Малахов К. С. Микросервисная архитектура как основа для построения сервисориентированных информационных систем //Материалы. – С. 32-33.

16. Зяблов Д. В., Кот А. А. ПРИМЕНЕНИЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ПРИ РАЗРАБОТКЕ КОРПОРАТИВНЫХ ВЕБ-ПРИЛОЖЕНИЙ //Электронный научный журнал. – 2017. – С. 17.

17. Алексанян Г. К., Тарасов А. Д., Клевец К. В. Анализ возможности применения языка программирования Java в задачах электроимпедансной томографии //Новая наука: От идеи к результату. – 2015. – №. 7-2. – С. 101-103.

18. Уолтс К. Spring в действии/Уолтс К //ДМК Пресс. – 2015. – Т. 754.

19. Холмс С. Getting MEAN with Mongo, Express, Angular, and Node. – Manning Publications Co., 2015 – С. 78-100.

20. Krochmalski J. IntelliJ IDEA Essentials. – Packt Publishing Ltd, 2014.

21. Feng X., Shen J., Fan Y. REST: An alternative to RPC for Web services architecture //Future Information Networks, 2009. ICFIN 2009. First International Conference on. – IEEE, 2009. – С. 7-10.

22. Копылов М. В., Кравец О. Я. Модель трехзвенной архитектуры «клиент-сервер» //Современные проблемы информатизации в проектировании и телекоммуникациях. – 2016. – №. 1. – С. 1.

23. Cogoluegnes A. et al. Spring batch in action. – Manning Publications Co., 2014.

24. Pollack M. et al. Spring Data: modern data access for enterprise Java. – " O'Reilly Media, Inc.", 2015.

25. King G., Bauer C. Hibernate in action //Greenwich, CT: Manning. – 2015. – С. 2-7.

26. Fowler M. Inversion of control containers and the dependency injection pattern. – 2014.

27. Liskin O., Singer L., Schneider K. Teaching old services new tricks: adding HATEOAS support as an afterthought //Proceedings of the Second International Workshop on RESTful Design. – ACM, 2015. – С. 3-10.

28. McIlroy M. D. et al. Mass-produced software components //Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany. – 2016. – С. 88-98.

29. Darwin P. B., Kozlowski P. AngularJS web application development. – Packt Publ., 2014.

30. Jain N., Bhansali A., Mehta D. AngularJS: A modern MVC framework in JavaScript //International Journal of Global Research in Computer Science (UGC Approved Journal). – 2015. – Т. 5. – №. 12. – С. 17-23.

31. Jadhav M. A., Sawant B. R., Deshmukh A. Single page application using angularjs //International Journal of Computer Science and Information Technologies. – 2015. – Т. 6. – №. 3. – С. 2876-2879.

32. Janzen D., Saiedian H. Test-driven development concepts, taxonomy, and future direction //Computer. – 2014. – Т. 38. – №. 9. – С. 43-50.

ПРИЛОЖЕНИЕ А

Декомпозиция побочных бизнес-процессов компании после интеграции разрабатываемого программного продукта

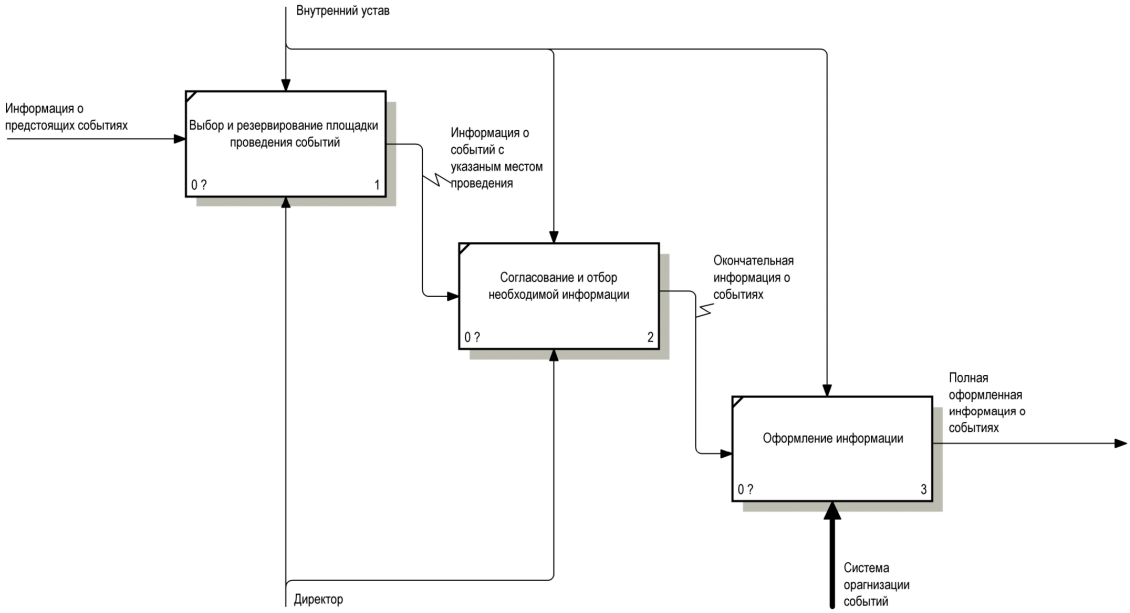

Рисунок А.1 – Диаграмма декомпозиции процесса оформления и рассылки электронных приглашений

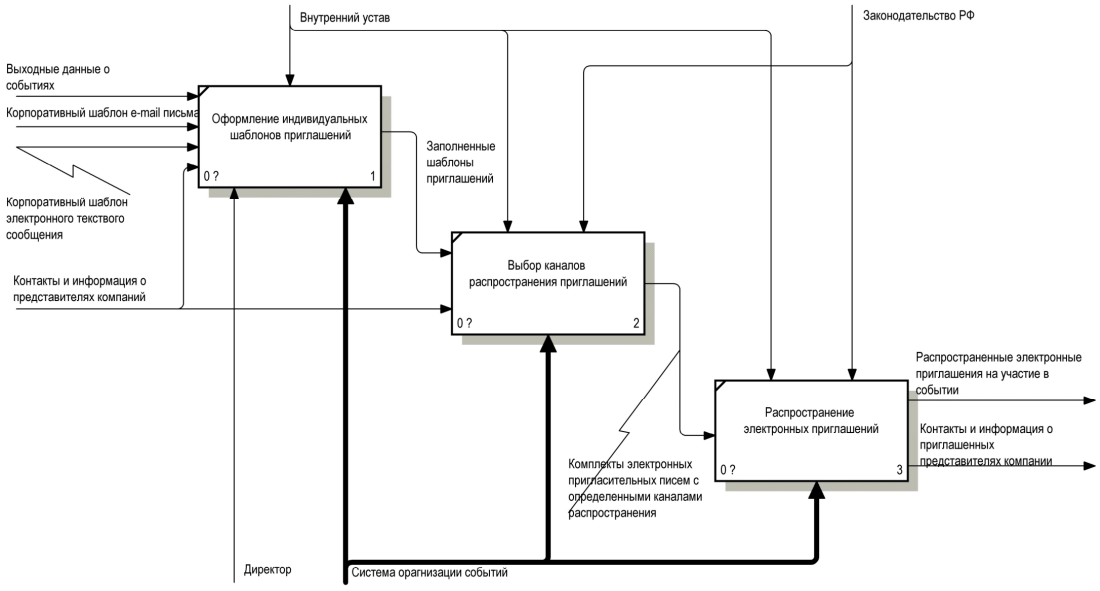

Рисунок А.2 –Диаграмма декомпозиции процесса рассылки оповещений и благодарностей

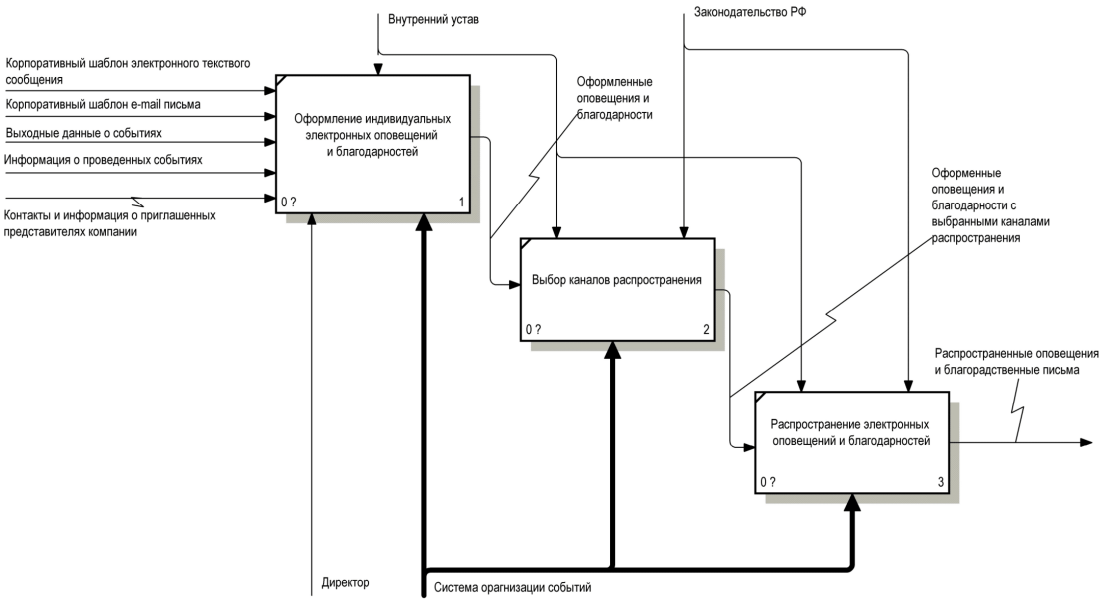

Рисунок А.3 – Диаграмма декомпозиции процесса согласования и оформления информации о событии

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
1. 110700026.09.03.02.349.ПЗВКР	Пояснительная записка	51 с.
	Графические документы	
2. 110700026.09.03.02.349.ДМВКР	Демонстрационные материалы (презентация)	13 сл.
	Демонстрационные материалы (пл. ф. А4)	65 лист.
	Другие документы	
3.	Документы на компакт-диске	1 CD

					110700026.09.03.02.349.ПЗВКР			
Изм.	Лист.	Номер докум.	Подп.	Дата				
Разработал					Тема ВКР Разработка модуля автоматизации организации событий средствами интернет Ведомость ВКР	Лит.	Лист	Листов
Проверил								
Н.контр.		Нестерова Е.В.				НИУ «БелГУ» гр.07001409		
Утвердил		Иващук О.А.						

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«30» мая 2018 г.

_____ Скубников Кирилл Сергеевич

(подпись) *(Ф.И.О.)*