МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

апФедеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

Допустить к защите

Заведующий кафедрой

канд. физ.-мат. наук, доц.,

\_\_\_\_\_\_\_\_\_\_\_\_В.В.Подколзин

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(БАКАЛАВРСКАЯ РАБОТА)**

**РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ**

**ДЛЯ ВЫЧИСЛЕНИЯ ОПРЕДЕЛЕНИЯ ЭФФЕКТИВНОСТИ ПРОИЗВОДСТВА**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.П.Козел

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность (профиль) \_\_\_\_\_\_\_\_«Информационные технологии» \_\_\_\_\_\_\_

Научный руководитель

канд. физ.-мат. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.В.Гаркуша

(подпись)

Нормоконтролер

ст.преп\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В.Харченко

(подпись)

Краснодар

2023

# **РЕФЕРАТ**

Выпускная квалификационная работа 45 стр., 6 ч., 18 рис., 10 источников, 4 приложения.

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, ВЫЧИСЛЕНИЕ ЭФФЕКТИВНОСТИ ПРОИЗВОДСТВА, КОЭФФИЦИЕНТ ЭФФЕКТИВНОСТИ, ОЦЕНКА СОТРУДНКИКОВ, KPI, PYTHON, DJANGO.

Цель работы: разработка клиент-серверного приложения для вычесления и определения эффективности производства на конкретном предприятии.

В ходе работы были изучены и изложены некоторые понятия из теории экономической, финансвой и производственной эффективности, принципы определения эффективности на производстве, способы реализации клиент-серверного приложения.

Научная новизна работы заключается в том, что предлагается усовершенствованный вариант определения эффективности в компании.

В результате работы было разработано приложение для руводителя предприятия, которое позволяет оценить сотрудников по системе KPI, вычислить эффективность предприятия за год по ключевым показателям.

По окончанию разработки приложения, программа была успешно внедрена на производство и используется на данный момент.

Введение

Всего несколько десятилетий назад общество было захвачено идеей эры передовых технологий. Сегодня это уже не идея, это реальность. В настоящее время Интернет является самой быстроразвивающейся средой обмена информацией в истории человечества. Трудно назвать какую-либо сферу деятельности, которая не имела бы своего полного и всесторонне отраженного в Интернете. Однако все это было бы трудно представить без веб-приложений. В 1995 году Netscape представил клиентский скриптовый язык под названием JavaScript, и, начиная с этого момента, можно проследить историю веб-приложений. Тем не менее, само понятие “веб-приложение” было введено только в 1999 году. Как правило, веб-приложения эволюционировали от статических веб-страниц до динамических веб-сайтов, которые обеспечивают индивидуальный пользовательский интерфейс на основе информации, известной или предоставленной пользователем. Веб-приложения используются для самых разных целей, например, для систем назначений, служб мгновенного обмена сообщениями, торговли, развлечений и даже демократического принятия решений в некоторых странах.

Эта выпускная работа посвящена внедрению веб-приложения для управления спортивными турнирами. Приложение позволяет пользователям создавать индивидуальные турниры, используя привлекательный и удобный интерфейс в веб-браузере. В то же время приложение дает возможность легко регистрироваться на турниры и производить платежи. Он также может быть использован для управления всеми турнирами и зарегистрированными участниками. Данные и функциональность, доступные пользователям, основаны на различных правах доступа. Все данные, относящиеся к приложение хранится в базе данных, которая расположена на стороне сервера. Сервер получает и обрабатывает все данные, которые отправляются из веб-браузера, и отправляет обратно все запрошенные данные.

1. Введение

* Обзор темы дипломной работы
* Цель и задачи работы
* Описание области применения web-приложения для управления публичными мероприятиями
* Обзор теоретических основ, связанных с темой дипломной работы, таких как управление проектами, управление событиями, технологии web-разработки и т.д.

1. Обзор существующих решений

* Обзор существующих web-приложений для управления публичными мероприятиями
* Сравнительный анализ существующих решений
* Выявление преимуществ и недостатков существующих решений
* Обзор теоретических основ, связанных с сравнительным анализом, таких как критерии выбора, методы сравнения, оценка эффективности и т.д.

1. Архитектура приложения

* Обзор выбранных технологий и фреймворков для разработки приложения
* Описание архитектуры приложения (фронтенд, бэкенд, база данных)
* Описание процесса разработки и интеграции компонентов приложения
* Обзор теоретических основ, связанных с архитектурой приложений, таких как моделирование, паттерны проектирования, принципы SOLID и т.д.

1. Описание функционала приложения

* Описание основных функций приложения, таких как создание и управление событиями, продажа билетов, управление посетителями и т.д.
* Описание процессов, которые реализованы в приложении, например, процесс создания события, процесс продажи билетов, процесс проверки билетов на мероприятии и т.д.
* Обзор теоретических основ, связанных с функциональностью приложения, таких как проектирование интерфейсов, UX/UI дизайн, методы разработки функциональности и т.д.

1. Тестирование и оптимизация приложения

* Описание процесса тестирования приложения
* Процесс оптимизации приложения для повышения производительности и улучшения пользовательского опыта
* Обзор теоретических основ, связанных с тестированием и оптимизацией приложений, таких как методы тестирования, профилирование кода, оптимизация баз данных и т.д.

1. Заключение

* Резюме проделанной работы
* Описание полученных результатов и достижений
* Описание возможных направлений для дальнейшего развития приложения
* Обзор теоретических основ, связанных с развитием и поддержкой web-приложений, таких как методы разработки, управление изменениями, поддержка пользователей и т.д.

НОВЫЙ ДИПЛОМ

**Введение**

В современном мире публичные мероприятия играют важную роль в общественной жизни и являются неотъемлемой частью многих сфер деятельности, таких как культура, спорт, бизнес и туризм. Однако, организация мероприятий может стать сложной задачей, требующей от организаторов много времени и усилий. В данном контексте, разработка web-приложения для управления публичными мероприятиями может значительно упростить процесс планирования, организации и проведения мероприятий.

Цель данной дипломной работы - разработать web-приложение для управления публичными мероприятиями, которое позволит организаторам быстро и эффективно создавать, редактировать и управлять мероприятиями, а также улучшить взаимодействие с участниками мероприятий и повысить качество их проведения.

В работе будут рассмотрены основные этапы разработки web-приложения, включая анализ требований, проектирование архитектуры, выбор технологий и фреймворков, реализацию функциональности и тестирование приложения. Также будет проведен анализ существующих аналогов и оценка конкурентоспособности разработанного приложения.

В итоге, разработка web-приложения для управления публичными мероприятиями может стать важным инструментом для организаторов мероприятий, позволяющим им эффективно управлять их проведением и повышать уровень взаимодействия с участниками.

Web-приложение для управления публичными мероприятиями может быть применено в различных областях деятельности, где проводятся мероприятия, включая культуру, спорт, бизнес и туризм.

В культурной сфере, web-приложение может быть использовано для управления организацией различных культурных мероприятий, таких как выставки, концерты, театральные постановки и фестивали. Организаторы могут использовать приложение для создания расписания, продажи билетов, управления маркетинговыми кампаниями, а также для общения с участниками мероприятий.

В сфере спорта, web-приложение может быть использовано для управления проведением спортивных событий, таких как соревнования, турниры и марафоны. Организаторы могут использовать приложение для создания расписания, управления регистрацией участников, проведения жеребьевки, а также для общения с участниками и зрителями мероприятий.

В бизнесе, web-приложение может быть использовано для управления корпоративными мероприятиями, такими как конференции, семинары и тренинги. Организаторы могут использовать приложение для создания расписания, управления регистрацией участников, проведения опросов и голосований, а также для обмена информацией с участниками мероприятий.

В туризме, web-приложение может быть использовано для управления организацией туристических мероприятий, таких как экскурсии, путешествия и круизы. Организаторы могут использовать приложение для создания расписания, управления бронированием и оплатой услуг, а также для обмена информацией с туристами.

Таким образом, web-приложение для управления публичными мероприятиями может быть применено в различных областях деятельности, где проводятся мероприятия, и помочь организаторам эффективно управлять их проведением.

В итоге, разработка web-приложения для управления публичными мероприятиями имеет большой потенциал для упрощения и улучшения процесса организации мероприятий, а также для улучшения взаимодействия с участниками. В данной дипломной работе будет рассмотрено проектирование и разработка такого приложения, а также его конкурентоспособность на рынке. Надеемся, что результаты работы будут полезны для организаторов мероприятий и помогут им повысить качество проведения мероприятий в различных сферах деятельности.

**Обзор темы дипломной работы**

Конечный продукт дипломной работы - веб-приложение для управления публичными мероприятиями. В современном мире проведение мероприятий является важным инструментом для привлечения внимания к бренду или продукту, а также для общения с целевой аудиторией. Организаторы мероприятий сталкиваются с большим количеством задач, связанных с планированием, рекламой, продажей билетов, управлением участниками и другими аспектами. Веб-приложение для управления мероприятиями помогает упростить и оптимизировать процесс организации мероприятий, что позволяет сэкономить время и ресурсы. В данной работе будет проведен обзор существующих веб-приложений для управления мероприятиями, оценена их функциональность, удобство использования, преимущества и недостатки. Будут определены основные требования к веб-приложению для управления мероприятиями, такие как создание и редактирование мероприятий, продажа билетов, управление участниками, аналитика и статистика и другие. Также будет разработана архитектура веб-приложения, которая включает диаграммы классов, схему базы данных и другие элементы. Будет описан процесс разработки веб-приложения, включая выбор технологий, реализацию функциональности, тестирование и отладку. Будет проведена оценка эффективности и потенциала веб-приложения для управления мероприятиями, а также обсуждены возможные дальнейшие улучшения и расширения на основе полученных результатов.

Кроме того, в работе будет проведен анализ требований к интерфейсу веб-приложения, который должен быть удобным и интуитивно понятным для пользователей. Будут определены основные функциональные требования к интерфейсу, такие как возможность создания и редактирования мероприятий, продажа билетов, управление участниками и другие. В работе также будет уделено внимание аспектам безопасности, таким как защита от несанкционированного доступа и защита персональных данных пользователей. Будет проведен анализ существующих методов и технологий для обеспечения безопасности веб-приложений и определены наиболее подходящие для данного проекта. В работе будет описан процесс тестирования веб-приложения, включая функциональное тестирование, тестирование производительности и тестирование безопасности. Будут представлены результаты тестирования и оценена работоспособность веб-приложения. В работе будет представлено руководство пользователя для веб-приложения, которое описывает основные функции и возможности приложения, а также инструкции по использованию. Будет проведен анализ рынка веб-приложений для управления мероприятиями, определены конкурентные преимущества разработанного приложения и оценен его потенциал на рынке. В работе будет представлен план дальнейшего развития и улучшения веб-приложения, включая добавление новых функций и улучшение существующих. В заключении будут сделаны выводы о проделанной работе, оценены полученные результаты и сформулированы рекомендации для дальнейшей разработки и использования веб-приложения для управления мероприятиями.

**Описание области применения**

Web-приложение для управления публичными мероприятиями может быть использовано в различных областях, связанных с организацией и проведением мероприятий. В первую очередь, это может быть полезно для организаторов конференций, семинаров, выставок и других мероприятий, которые требуют подробной организации и координации. С помощью веб-приложения они смогут легко и быстро создавать страницы мероприятий, продавать билеты, управлять участниками и проводить анализ эффективности. Это также может быть полезно для организаторов культурных мероприятий, таких как концерты, фестивали и театральные представления.

Веб-приложение может также быть использовано для управления спортивными событиями, такими как соревнования, турниры и марафоны. Организаторы смогут легко управлять регистрацией участников, продавать билеты и отслеживать результаты соревнований. Это может быть полезно как для организаторов профессиональных спортивных мероприятий, так и для любителей спорта, которые хотят организовать свой собственный турнир.

Web-приложение может быть использовано также для управления бизнес-мероприятиями, такими как семинары, тренинги и корпоративные мероприятия. Организаторы смогут легко управлять регистрацией участников, продавать билеты и создавать программы мероприятий.

Web-приложение может быть использовано также для управления образовательными мероприятиями, такими как курсы, лекции и вебинары. Организаторы смогут легко управлять регистрацией участников, продавать билеты и создавать программы мероприятий.

Web-приложение может быть использовано также для управления благотворительными мероприятиями, такими как аукционы, гала-вечера и другие мероприятия, направленные на сбор средств для благотворительных организаций. Организаторы смогут легко управлять продажей билетов и сбором пожертвований.

Web-приложение может быть использовано также для управления религиозными мероприятиями, такими как церковные службы, праздники и другие мероприятия, связанные с религиозными обрядами. Организаторы смогут легко управлять регистрацией участников и создавать программы мероприятий.

Web-приложение может быть использовано также для управления различными мероприятиями, связанными с туризмом, такими как экскурсии, путешествия и круизы. Организаторы смогут легко управлять продажей билетов и создавать программы мероприятий.

Web-приложение может быть использовано также для управления мероприятиями, связанными с развлечениями, такими как концерты, фестивали, парки развлечений и другие мероприятия. Организаторы смогут легко управлять продажей билетов и создавать программы мероприятий.

Web-приложение может быть использовано также для управления мероприятиями, связанными с государственными и политическими событиями, такими как выборы, политические митинги и другие мероприятия. Организаторы смогут легко управлять регистрацией участников и создавать программы мероприятий.

Web-приложение может быть использовано также для управления мероприятиями, связанными с научными конференциями, симпозиумами и другими мероприятиями, связанными с исследованиями и разработками. Организаторы смогут легко управлять регистрацией участников и создавать программы мероприятий.

Web-приложение может быть использовано также для управления мероприятиями, связанными с музыкой, театром и другими видами искусства. Организаторы смогут легко управлять продажей билетов и создавать программы мероприятий.

**Обзор теоретических основ, связанных с темой дипломной работы, таких как управление проектами, управление событиями, технологии web-разработки и т.д.**

Область применения web-приложений для управления публичными мероприятиями является весьма широкой и включает в себя организацию и управление различными мероприятиями, такими как конференции, выставки, фестивали, спортивные соревнования и др. В связи с этим, существует множество теоретических основ, связанных с управлением проектами и событиями, технологиями web-разработки и другими аспектами, которые оказывают влияние на эффективность и функциональность web-приложений для управления мероприятиями.

Одним из основных аспектов, связанных с управлением проектами, является планирование и контроль бюджета, времени и ресурсов. Важно разработать эффективный план проекта, который позволит управлять ресурсами и соблюдать сроки выполнения задач. При этом необходимо учитывать риски и предусмотреть механизмы их минимизации.

Управление событиями также имеет свои особенности. Для успешной организации мероприятий необходимо правильно выбрать место проведения, определить формат мероприятия, разработать программу, обеспечить безопасность и комфорт участников и посетителей. Важно также учитывать взаимодействие с партнерами, спонсорами и другими заинтересованными сторонами.

Технологии web-разработки также играют важную роль в создании web-приложений для управления мероприятиями. Одним из ключевых аспектов является использование современных фреймворков и библиотек, таких как React, Angular, Vue.js и других, которые позволяют упростить и ускорить процесс разработки и обеспечить высокую производительность и удобство использования приложения.

Кроме того, важным аспектом является обеспечение безопасности приложения и защита данных пользователей. Для этого необходимо использовать современные методы шифрования и механизмы аутентификации и авторизации.

Также важно учитывать психологические и социологические аспекты управления мероприятиями. Например, необходимо учитывать психологические особенности посетителей и участников, чтобы обеспечить максимальный комфорт и удобство при посещении мероприятия. Кроме того, важным аспектом является учет социологических тенденций и трендов, чтобы создать мероприятие, которое будет интересно и актуально для целевой аудитории.

В общем, разработка web-приложения для управления публичными мероприятиями требует учета множества аспектов и факторов, связанных с управлением проектами и событиями, технологиями web-разработки, безопасностью, психологическими и социологическими особенностями. Важно создать приложение, которое будет эффективно и удобно для пользователей, а также обеспечивать безопасность и защиту данных. Для этого необходимо учитывать современные тенденции и использовать современные технологии разработки.

Разработка веб-приложения для управления публичными мероприятиями является сложной и многогранной задачей, которая требует учета множества теоретических основ и научных подходов. Одним из ключевых аспектов в этом контексте является управление проектами, которое включает в себя планирование, контроль и управление ресурсами.

Одним из методов управления проектами, применяемых в создании web-приложений, является методология Agile. Она позволяет создавать приложения быстро и гибко, учитывая изменения требований и потребностей пользователей. В рамках этой методологии используются такие понятия, как sprint (цикл разработки), backlog (список задач) и user story (описание требований пользователя).

Кроме того, важным аспектом является управление рисками. Использование методов анализа рисков позволяет оценить вероятность возникновения проблем и разработать стратегии их предотвращения или устранения. Один из таких методов - анализ SWOT, который позволяет выявить сильные и слабые стороны проекта, а также возможности и угрозы, связанные с его реализацией.

Для разработки web-приложения необходимо учитывать множество технических аспектов, таких как выбор языка программирования, архитектура приложения, взаимодействие с базами данных и т.д. Важным этапом является тестирование, которое позволяет обнаружить и устранить ошибки и проблемы в работе приложения.

Одним из научных экспериментов, связанных с разработкой web-приложений, является анализ пользовательского поведения. С помощью таких методов, как эксперименты A/B тестирования, можно выявить предпочтения и потребности пользователей и оптимизировать приложение с учетом этих факторов.

Важным аспектом при разработке web-приложения является безопасность. Для защиты приложения от взлома используются такие методы, как шифрование данных, аутентификация и авторизация пользователей, защита от SQL-инъекций и другие.

Кроме того, для управления публичными мероприятиями необходимо учитывать такие аспекты, как маркетинговые стратегии, управление бюджетом и взаимодействие с партнерами и спонсорами.

Для учета всех этих аспектов в создании web-приложения можно использовать различные инструменты и технологии, такие как Git (система контроля версий), Docker (контейнеризация), Jenkins (система непрерывной интеграции и развертывания) и другие.

Таким образом, разработка веб-приложения для управления публичными мероприятиями является сложной задачей, требующей учета множества научных подходов и технологий. Однако, правильно спроектированное и разработанное приложение может значительно упростить процесс проведения мероприятий и повысить их эффективность.

В заключение, можно отметить, что разработка веб-приложения для управления публичными мероприятиями является важной задачей, которая требует учета множества технических, организационных и маркетинговых аспектов. Для ее успешной реализации необходимо использование современных методов управления проектами, а также научных подходов, таких как анализ рисков и пользовательского поведения. Однако, правильно спроектированное и разработанное приложение может значительно упростить процесс проведения мероприятий и повысить их эффективность.

**Обзор популярных решений**

Одним из наиболее популярных web-приложений для управления мероприятиями является Eventbrite. Это приложение позволяет создавать и продвигать мероприятия, продавать билеты и управлять участниками. Одним из основных преимуществ Eventbrite является его простота в использовании и интеграция с другими приложениями и сервисами. Однако, приложение не всегда удобно для настройки более сложных мероприятий и может иметь высокую комиссию за продажу билетов.

Еще одним популярным web-приложением является Cvent. Оно предназначено для управления различными мероприятиями, включая конференции, выставки и семинары, и предоставляет широкий спектр инструментов для планирования, организации и анализа мероприятий. Основным преимуществом Cvent является его гибкость и возможность настройки под различные потребности. Однако, приложение может быть дорогостоящим для небольших мероприятий.

Splash – это web-приложение, которое позволяет создавать красивые и эффективные страницы для мероприятий, включая информацию о мероприятии, продажу билетов и управление участниками. Splash предоставляет широкий набор инструментов для настройки страницы мероприятия и его продвижения, таких как интеграция с социальными сетями и возможность создания видео-приглашений. Однако, приложение может иметь ограниченные функциональные возможности для более сложных мероприятий.

Большинство web-приложений для управления мероприятиями также предоставляют мобильные приложения для удобства пользователей. Например, приложение Attendify позволяет создавать мобильные приложения для мероприятий с возможностью создания программы, общения участников и других инструментов. Однако, создание мобильного приложения может быть дополнительной затратой и требовать дополнительного времени на настройку.

Еще одним примером web-приложения для управления мероприятиями является RegOnline. Оно предоставляет инструменты для создания страниц мероприятий, продажи билетов, управления участниками и анализа данных. Одним из преимуществ RegOnline является его обширный набор инструментов для анализа данных и отчетности. Однако, приложение может быть сложным в использовании для новичков и иметь ограниченную интеграцию с другими приложениями.

Bizzabo – это web-приложение, которое предоставляет инструменты для создания страниц мероприятий, продажи билетов, управления участниками и создания мобильных приложений для мероприятий. Одним из преимуществ Bizzabo является его интеграция с социальными сетями и возможность создания персонализированных приглашений и уведомлений для участников. Однако, приложение может быть дорогостоящим для небольших мероприятий.

Помимо вышеуказанных приложений, существует множество других web-приложений для управления мероприятиями, таких как Eventzilla, Ticketbud, amiando и другие. Каждое из них имеет свои особенности, преимущества и недостатки, и выбор приложения зависит от конкретных потребностей пользователей.

Важным аспектом при выборе web-приложения для управления мероприятиями является его интеграция с другими приложениями и сервисами. Например, интеграция с системой электронной коммерции может позволить быстро и безопасно продавать билеты, а интеграция с социальными сетями – продвигать мероприятие и привлекать новых участников.

Еще одним важным аспектом является возможность настройки web-приложения под конкретные потребности мероприятия. Например, приложение должно поддерживать различные типы билетов, настройки цен, создание опросов и получение обратной связи от участников.

Безопасность и защита данных также являются важными аспектами при выборе web-приложения для управления мероприятиями. Приложение должно обеспечивать безопасность персональных данных участников.

**Сравнительный анализ существующих решений**

Существует множество web-приложений для управления публичными мероприятиями, каждое из которых имеет свои преимущества и недостатки. В данном разделе мы проведем сравнительный анализ наиболее популярных решений: Eventbrite, Meetup, Cvent и Eventzilla.

Начнем с Eventbrite. Это один из самых популярных сервисов для создания и продвижения мероприятий. Eventbrite предоставляет широкий спектр инструментов для продажи билетов, регистрации участников, создания событийных страниц и многого другого. Среди его преимуществ можно отметить удобный интерфейс и интеграцию с многими популярными платежными системами. Однако Eventbrite может быть дорогим для небольших мероприятий, а также не предоставляет достаточно гибких функциональных возможностей.

Meetup - это социальная платформа для организации и поиска событий в реальном мире. Его основная цель - объединение людей с общими интересами в офлайн-среде. Meetup предоставляет инструменты для создания и продвижения мероприятий, а также позволяет участникам создавать собственные группы с обсуждением тематики мероприятий. Одним из основных преимуществ Meetup является возможность объединения людей с общими интересами, что может увеличить число участников мероприятий. Однако Meetup может быть не подходящим для бизнес-мероприятий и не предоставляет возможность продажи билетов.

Cvent - это интегрированная платформа для управления событиями, которая предоставляет множество инструментов для планирования, регистрации и продвижения мероприятий. Cvent имеет широкий спектр функциональных возможностей, включая создание анкет для участников, продажу билетов, создание событийных страниц и многое другое. Одним из главных преимуществ Cvent является его гибкость и возможность настройки инструментов под конкретные потребности мероприятий. Однако, Cvent может быть дорогим для небольших мероприятий.

Eventzilla - это web-приложение, которое предоставляет инструменты для управления событиями, включая создание событийных страниц, продажу билетов, регистрацию участников и многое другое. Eventzilla является более доступным по цене, чем многие другие решения, и имеет удобный интерфейс. Однако Eventzilla может быть не подходящим для крупных мероприятий и не имеет такого широкого спектра функциональных возможностей, как Cvent или Eventbrite.

В целом, при выборе web-приложения для управления публичными мероприятиями необходимо учитывать конкретные потребности мероприятия, бюджет и функциональные возможности приложения. Каждое из решений, описанных выше, имеет свои преимущества и недостатки, поэтому необходимо проанализировать их и выбрать подходящее для конкретной ситуации.

**Выявление преимуществ и недостатков существующих решений**

При выявлении преимуществ и недостатков существующих решений для управления публичными мероприятиями, следует проанализировать каждое из рассмотренных web-приложений. Ниже приведены основные преимущества и недостатки каждого из них:

1. Eventbrite:

- Преимущества: великолепная интеграция с социальными сетями, удобная платформа для продвижения мероприятий, отличные инструменты для управления билетами и продажами.

- Недостатки: ограниченные возможности для настройки дизайна страницы мероприятия, высокие комиссии на продажу билетов.

2. Cvent:

- Преимущества: широкие возможности для настройки страницы мероприятия, удобный интерфейс пользователя, великолепные инструменты для управления гостиничными номерами и бронированием.

- Недостатки: высокая стоимость использования, ограниченные возможности для настройки мобильного приложения.

3. Bizzabo:

- Преимущества: удобный интерфейс пользователя, широкие возможности для настройки страницы мероприятия, интеграция с социальными сетями и электронной почтой.

- Недостатки: ограниченные возможности для настройки мобильного приложения, высокая плата за использование.

4. Splash:

- Преимущества: широкие возможности для настройки страницы мероприятия, удобный интерфейс пользователя, отличные инструменты для управления билетами и продажами.

- Недостатки: ограниченные возможности для настройки мобильного приложения, высокие комиссии на продажу билетов.

Исходя из анализа, можно сделать вывод, что каждое из рассмотренных решений имеет свои преимущества и недостатки. Однако, для разработки нового веб-приложения для управления публичными мероприятиями, следует учитывать наиболее важные критерии для пользователей и выбрать наиболее подходящий функционал для приложения.

**Описание процесса разработки и интеграции компонентов приложения**

Описание процесса разработки и интеграции компонентов приложения начинается с выбора технологий, которые будут использоваться в проекте. В данном случае мы выбрали Vue.js для фронтенд-разработки, Django для бэкенд-разработки и Mongo DB для хранения данных.

Первым шагом в процессе разработки было создание дизайна и макета для приложения. Затем мы приступили к разработке фронтенд-части приложения с использованием Vue.js. Мы использовали компонентный подход для создания отдельных компонентов, которые могут быть использованы повторно в разных частях приложения.

Далее мы приступили к созданию бэкенд-части приложения с использованием Django. Мы создали модели данных, которые определяют структуру и связи между данными. Затем мы создали представления, которые обрабатывают запросы от фронтенда и возвращают данные. Мы также создали URL-адреса для каждого представления, чтобы они могли быть доступны из фронтенда.

После этого мы приступили к интеграции фронтенда и бэкенда. Мы использовали API, чтобы обмениваться данными между фронтендом и бэкендом. Мы также создали авторизацию и аутентификацию пользователей с помощью токенов, чтобы защитить доступ к приложению.

Наконец, мы интегрировали Mongo DB для хранения данных. Мы создали модели данных, которые соответствуют структуре данных, используемых в приложении. Мы также создали соединение с базой данных, чтобы данные могли быть сохранены и извлечены.

Общий процесс разработки и интеграции компонентов приложения включает в себя следующие шаги:

1. Выбор технологий.
2. Создание дизайна и макета для приложения.
3. Разработка фронтенд-части приложения.
4. Создание бэкенд-части приложения.
5. Интеграция фронтенда и бэкенда с использованием API.
6. Создание авторизации и аутентификации пользователей.
7. Интеграция Mongo DB для хранения данных.

После того, как мы завершили интеграцию всех компонентов приложения, мы провели тестирование приложения, чтобы убедиться, что все функции работают правильно и нет ошибок. Мы использовали различные методы тестирования, включая модульное тестирование, функциональное тестирование и интеграционное тестирование.

Модульное тестирование позволяет проверить, что каждый компонент приложения работает правильно в отдельности. Функциональное тестирование позволяет проверить, что приложение работает правильно в целом и выполняет все функции, которые должны быть доступны для пользователей. Интеграционное тестирование позволяет проверить, что все компоненты приложения работают правильно вместе.

После того, как мы завершили тестирование приложения, мы развернули его на сервере, чтобы пользователи могли получить к нему доступ. Мы использовали различные инструменты и сервисы, такие как Amazon Web Services и Heroku, чтобы развернуть приложение и обеспечить его стабильную работу.

В процессе разработки приложения мы также уделяли внимание безопасности и защите данных. Мы использовали различные методы для защиты приложения и данных, включая шифрование данных, защиту от атак типа SQL Injection и Cross-Site Scripting, а также обеспечение безопасности при обмене данными между фронтендом и бэкендом.

В целом, процесс разработки веб-приложения для управления публичными мероприятиями включает в себя множество шагов, начиная с выбора технологий и заканчивая развертыванием приложения на сервере. Каждый шаг является важным и требует внимательности и тщательного планирования, чтобы обеспечить качественную и стабильную работу приложения.

**Обзор теоретических основ, связанных с архитектурой приложений, таких как моделирование, паттерны проектирования, принципы SOLID и т.д.**

Обзор теоретических основ, связанных с архитектурой приложений, является важным шагом в процессе разработки веб-приложения для управления публичными мероприятиями. Моделирование, паттерны проектирования, принципы SOLID и другие теоретические основы помогают создавать более гибкие, масштабируемые и удобные для сопровождения приложения.

Моделирование позволяет разработчикам создавать абстрактные модели реальной системы, что позволяет лучше понимать ее компоненты и взаимодействия между ними. Это помогает создать более эффективную архитектуру приложения.

Паттерны проектирования - это повторяемые решения для часто встречающихся проблем в проектировании приложений. Они помогают создавать более гибкие, масштабируемые и удобные для сопровождения приложения. Например, паттерн MVC (Model-View-Controller) разделяет приложение на три слоя, что упрощает сопровождение и изменение приложения.

Принципы SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion) - это набор принципов проектирования программного обеспечения, которые помогают создавать более гибкие и удобные для сопровождения приложения. Например, принцип единственной ответственности (SRP) гласит, что каждый класс должен иметь только одну ответственность, что упрощает сопровождение и изменение приложения.

Другие теоретические основы, такие как DRY (Don't repeat yourself) и KISS (Keep it simple, stupid), также помогают создавать более гибкие, масштабируемые и удобные для сопровождения приложения. Например, принцип DRY гласит, что не следует дублировать код, а следует использовать повторно уже написанный код.

Кроме того, для эффективного проектирования приложения разработчики также могут использовать другие теоретические основы, такие как принципы компонентного программирования, шаблоны проектирования и т.д. Все эти теоретические основы помогают создавать более гибкие, масштабируемые и удобные для сопровождения приложения, что важно для успешной разработки и поддержки веб-приложения.

Для разработки веб-приложения для управления публичными мероприятиями также важно учитывать специфические требования, связанные с безопасностью, скоростью работы и масштабируемостью приложения. Например, при разработке приложения необходимо учитывать возможные уязвимости и применять соответствующие меры защиты.

Для обеспечения высокой скорости работы приложения можно использовать кэширование данных, оптимизацию запросов к базе данных и другие техники оптимизации.

Для обеспечения масштабируемости приложения можно использовать техники горизонтального и вертикального масштабирования, которые позволяют увеличивать производительность приложения с ростом нагрузки.

Еще одним важным аспектом проектирования веб-приложения является выбор подходящих технологий и инструментов. Например, для разработки веб-приложения можно использовать различные фреймворки, библиотеки и языки программирования.

При выборе технологий необходимо учитывать требования к приложению, а также опыт разработчиков и доступные ресурсы. Например, если приложение должно обрабатывать большое количество запросов, то может быть целесообразно выбрать язык программирования, который обеспечивает высокую производительность.

Также важным аспектом является выбор базы данных для хранения данных приложения. Необходимо учитывать требования к скорости работы и масштабируемости базы данных, а также ее совместимость с выбранными технологиями приложения.

**РЕАЛИЗАЦИЯ**

**Описание проекта**

Проект представляет собой веб-приложение, которое было разработано для облегчения процесса создания, управления, регистрации и оплаты танцевальных турниров. Разработанная платформа предоставляет часть всех необходимых функций. Тем не менее, в приложение можно легко добавить больше функций. На момент написания данной статьи были реализованы следующие интерфейсы были реализованы:

Интерфейс администратора для создания и управления учетными записями различных организаций, организующих турниры.

- Интерфейс администратора для чата поддержки.

- Интерфейс организации для просмотра и обновления контактной и деловой информации.

- Интерфейс организации для просмотра и обновления платежных реквизитов.

- Интерфейс организации для создания и управления созданными турнирами с

очень индивидуальными настройками.

- Интерфейс организации для создания полей форм и групп форм для заполнения участниками в процессе регистрации.

- Интерфейс организации для создания и просмотра созданных шаблонов электронной почты с возможностью вставки подстановочных знаков, которые впоследствии могут быть автоматически заменены на данные соответствующего участника турнира.

- Интерфейс организации для чата поддержки.

- Интерфейс участника для просмотра всех доступных турниров и с возможностью зарегистрироваться для участия в выбранном турнире.

Как видно из приведенного выше списка, приложение состоит из трех основных частей: интерфейсы администратора, организации и участника.

**Настройки среды**

Приложение было разработано на настольном компьютере под управлением операционной системы Windows 10.

Помимо ПК и ОС для реализации требовалась среда разработки, чтобы быть создана. Среда разработки состоит из различных программ и инструментов для работы с ними. Таким образом, было установлено следующее программное обеспечение:

* Node.js

Node.js является важной частью для процесса разработки данного проекта, поскольку он обеспечивает среду выполнения для выполнения кода JavaScript. Его можно загрузить с официального сайта (https://nodejs.org/en/download/). После установки, самый простой способ проверить, правильно ли он работает, - это выполнить команду node -v в терминале. Команда выводит установленную версию Node.js. Во время разработки использовалась версия 6.3.1.

* MongoDB

MongoDB является второй по важности частью среды разработки, поскольку

приложению требуется база данных для хранения и запроса различных данных. Она может быть установлена с официального сайта (https://www.mongodb.com/download-center). После установки сервер MongoDB может быть запущен с помощью команды mongod из терминала. В ходе разработки использовалась версия 3.2.3.

* WebStorm

WebStorm - это очень мощная интегрированная среда разработки (IDE) для разработки на JavaScript разработки. Она имеет встроенную поддержку проектов Node.js. А также завершение кода, мощные функции навигации, обнаружения ошибок на лету и рефакторинга, она имеет дружественный интерфейс для отладки приложений Node.js.

* Robomongo

Robomongo - это инструмент управления MongoDB с очень дружественным пользовательским интерфейсом и встроенной поддержкой оболочки MongoDB.

* SourceTree

SourceTree - это очень мощный графический интерфейс Git с отличным и простым интерфейсом.

**Структура и реализация проекта**

Структура проекта хорошо определена, и компоненты разделены соответственно их роли. Обзор структуры можно увидеть на следующем рисунке:

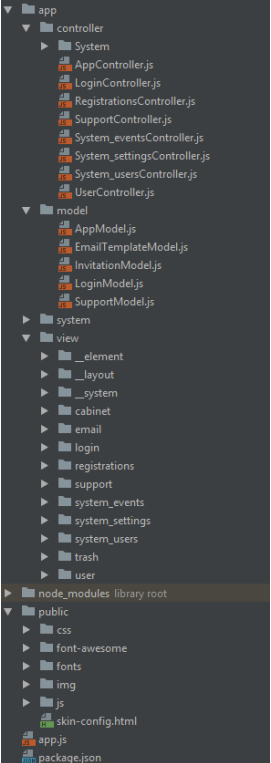


Рисунок n – обзор структуры проекта.

На рисунке N показана структура приложения. Точкой входа для приложения является файл "app.js". Он содержит всю инициализацию, конфигурацию, маршрутизацию и сервер. Бизнес-логика расположена в директории "controller". Все модели соответствующие коллекциям баз данных, находятся в каталоге "model". Каталог директория "view" содержит компоненты представления, которые показываются пользователю приложения. Все изображения, CSS, шрифты, картинки и JS-код, используемые в front-end, лежат в директории "public". Различные служебные функции находятся в директории "system".

А файл "package.json" содержит список всех зависимостей, которые используются в проекте.

Как было сказано выше, "app.js" - это основной и начальный файл приложения. Прежде всего, он считывает и инициализирует все модели в каталоге моделей. Затем он требует и инициализирует экземпляры всех контроллеров, которые находятся в папке controller. Затем создается экземпляр "express" и устанавливается соединение с базой данных с помощью модуля "mongoose". После этого для экземпляра "express" выполняются все необходимые конфигурации. Затем с помощью модуля "passport" создается простая система аутентификации сессии. Далее создаются обработчики для GET и POST запросов и отдельно для "/login" создаются с помощью экземпляра "express". И, наконец, сервер "express" запускается на порту, указанном в конфигурации.

Следующей важнейшей частью приложения является бизнес-логика. Контроллеры, которые определенные в каталоге "controller", отвечают за всю бизнес-логику, которая происходит в приложении. Каждый контроллер имеет одинаковую структуру и должен расширять главный контроллер, который называется "AppController". Он содержит логику для разбора и подготовки запроса, чтобы информация была легко доступна каждому из дочерних компонентов. Помимо логики инициализации и подготовки, "AppController" имеет общие методы, которые часто используются производными классами.

Как было рассмотрено выше, каталог моделей содержит модели, которые являются абстракцией над соответствующими коллекциями MongoDB. Каждая модель должна расширять класс, который называется "AppModel". Родительский класс содержит логику для инстанцирования модели Mongoose с соответствующей схемой. Более того, общие методы create, read, update, delete (CRUD) для работы с коллекцией также определены в "AppModel".

Представления являются последней частью приложения. Каждое представление представляет собой шаблон EJS с определенной HTML-разметкой и JavaScript-кодом для вставки данных в шаблон. Некоторые файлы EJS содержат небольшой компонент, созданный для конкретной страницы, а часть представлений определяет общий макет, который используется во всем приложении. Макет, тело и все необходимые EJS-шаблоны объединяются в окончательное представление перед отправкой клиенту. Таким образом, пользователь приложения видит полную веб-страницу HTML с необходимыми компонентами.

**Система запросов-ответов (request-response flow)**

Поток запрос-ответ в приложении, описанном в данной статье, основан на пользовательском фреймворке. Было решено создать собственный фреймворк поверх стандартного "экспресс" маршрутизатора, чтобы иметь лучший контроль и гибкость. Таким образом, каждый GET и POST запрос перехватывается соответствующим обработчиком, а затем URL разбирается, чтобы определить правильный контроллер и его метод для обработки запроса. URL должен быть в форме:

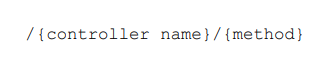


Рисунок n – URL шаблон

Как видно из рисунка 6, каждый URL запроса должен содержать имя контроллера и правильный метод. Следовательно, после разбора URL проверяется, есть ли в списке всех контроллеров запрашиваемый контроллер. Если контроллер отсутствует в списке, клиенту будет отправлен статус 404. С другой стороны, если контроллер присутствует, объекты запроса и ответа подготавливаются и добавляются к экземпляру контроллера, а затем вызывается метод запрашиваемый метод вызывается с параметрами GET или POST. Таким образом, внутри вызываемого метода есть доступ к объектам запроса, ответа и запрашиваемым параметрам. Наряду с объектами запроса и ответа, модель по умолчанию с таким же именем, что и контроллер, доступна в области видимости метода. Во время выполнения метода выполняется вся необходимая бизнес-логика, такая как чтение или запись в базу данных и обработка данных. В конце выполнения метода можно отправить ответ с одними данными или отрисовать необходимую страницу или конкретный элемент с данными, используя специализированной системы представления и отправить обратно HTML-разметку.

Таким образом, созданный фреймворк обеспечивает очень простой, гибкий и прозрачный процесс разработки. Очень удобно добавлять дополнительные контроллеры, модели и представления. Используя этот фреймворк, можно легко расширить приложение, добавив больше страниц, информации и функциональности.

**Контроллеры**

В этой подглаве подробно рассматривается и описывается роль каждого контроллера, представленного в самой большой части проекта, которой является организационный интерфейс.

* Контроллер кабинета

Контроллер кабинета отвечает за сохранение и извлечение информации об организации, такой как платежные реквизиты, контактная и деловая информация. Существует несколько POST и GET методов, представленных в этом контроллере. Прежде всего, базовый метод "index":

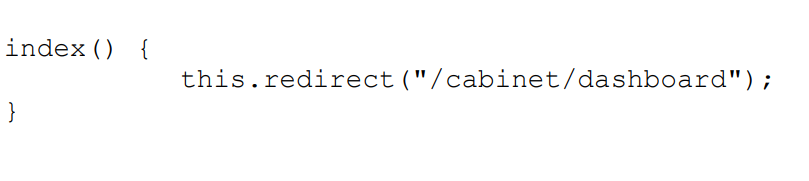


Рисунок n - "Индексный" метод

Как видно из листинга n, метод "index" перенаправляет на относительный URI "cabinet/dashboard". Метод перенаправления объявлен внутри "AppController" и доступен поскольку каждый контроллер расширяет "AppController". Метод вызывает функцию перенаправления объекта response. Перенаправленный URI вызывает метод "dashboard" того же самого контроллера, что продемонстрировано ниже:

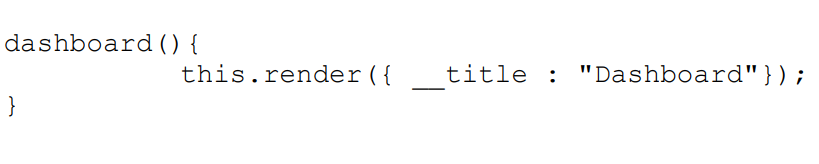


Рисунок n – метод “dashboard”

На рисунке 8 показана реализация метода "приборной панели" контроллера шкафа. Метод вызывает функцию "render" на экземпляре класса. Метод "Render", определенный в "AppController", подготавливает необходимые опции и рендерит представление, соответствующее методу "dashboard". Представление содержит интерфейс приборной панели с обзором статистики. Во-вторых, существует три метода для сохранения информации об организации: "saveContactInfo", "saveBusinessInfo" и "savePaymentCredentials". Все они имеют одинаковую структуру:

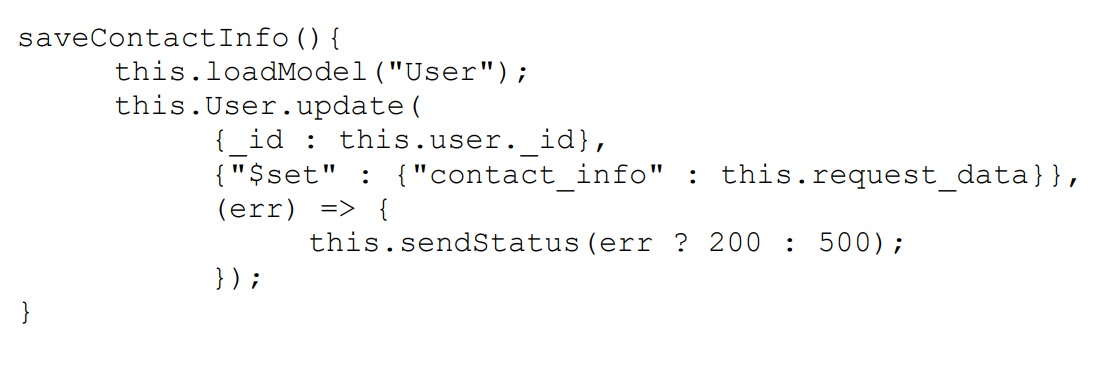


Рисунок n – метод “saveContactInfo”

Рисунок 9 демонстрирует фрагмент кода метода "saveContactInfo". Первая строка в метода вызывает метод "loadModel" из "AppController" для подготовки запрашиваемой модель и добавить ее к экземпляру класса. Вторая строка вызывает метод "update" загруженной модели "User". Первым аргументом является запрос, означающий, какие документы в коллекции должны быть обновлены. Второй аргумент содержит полезную нагрузку обновления.

Третий и последний аргумент - это функция обратного вызова, которая будет вызвана после того, как обновление будет ибо успешным, либо возвращающим ошибку. Обратный вызов проверяет, возвращена ли ошибка, и отправляет соответствующий статус клиенту. Метод "SendStatus" является абстракцией над методом "sendStatus" метода объекта ответа, созданного "express".

Наконец, для отображения сохраненных данных, используются два аналогичных метода: метод "info" для получения и отображения контактной и бизнес-информации и метод "paymentCredentials" для отображения представления с данными о платежных реквизитах. Оба метода содержат только одну строку:

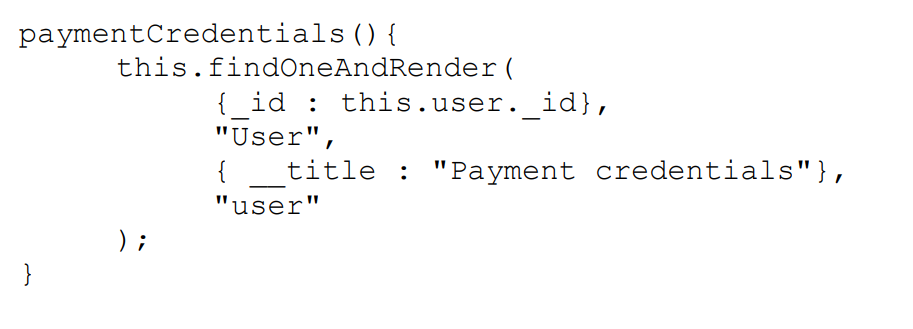


Рисунок n – метод “paymentCredentials”.

Пример кода, продемонстрированный в листинге 10, показывает реализацию метода "paymentCredentials". Как можно видеть, метод вызывает только метод "findOneAndRender", который определен в "AppController". Этот метод является абстракцией над запросом данных из базы данных и рендеринга конкретного представления с этими данными. Первый аргумент - это запрос к базе данных. Второй - имя модели, которая должна быть запрошена.

Следующий аргумент содержит дополнительные данные, которые будут переданы представлению. И имя представления передается в качестве последнего аргумента методу "findOneAndRender".

**Контроллер полей формы**

Контроллер полей формы имеет логику для отображения и сохранения полей формы, созданных пользователя, а также формирования и отображения групп созданных полей формы. Созданные поля форм и группы полей форм используются при создании турниров и шаблонов писем.

Логику контроллера строят следующие методы. Метод "Index" перенаправляет на метод "show", который отображает интерфейс для создания и отображения полей формы.

Создание полей формы поддерживается методом "save", который подготавливает и сохраняет нужное поле формы. И, конечно же, есть метод "remove" для удаления созданного поля формы. Большая часть логики в контроллере занята обработкой формы групп полей. Прежде всего, метод "groups" находит все ранее созданные группы форм и поля формы, а также создает представление для создания групп из созданных полей формы и отображения обзора найденных групп. С помощью метода "getGroup" можно получить представление с подробной информацией о запрашиваемой группе форм. Для создания и удаления групп необходимы следующие методы: "saveGroup" и "removeGroup" соответственно.

**Контроллер электронной почты**

Контроллер электронной почты отвечает за сохранение и отображение шаблонов электронной почты, которые могут быть созданы с помощью модульного редактора. Также возможно вставить в шаблон подстановочные знаки, которые были сформированные из полей созданной формы. Первый и основной метод - "индекс". Он перенаправляет на метод "myTemplates" для отображения представления, содержащего список всех шаблонов электронной почты, которые были созданы пользователем. Из этого представления можно либо удалить шаблон с помощью функции помощью метода "removeTemplate" или отредактировать шаблон. Если выбрана опция редактирования, страница будет перенаправлена на метод "templateEditor", который рендерит представление с помощью редактором, а с помощью метода "getTemplate" данные выбранного шаблона загружаются на клиент. Представление с редактором также содержит список подстановочных знаков, которые могут быть использованы при создании различных шаблонов. Для получения списка подстановочных знаков используется метод "getWildcards". И наконец, новый или отредактированный шаблон может быть сохранен в базе данных с помощью метода "saveTemplate".

**Контроллер входа в систему**

Контроллер входа в систему имеет только два метода. "Index", который отображает страницу входа в систему, и "logout" который выводит пользователя из системы и перенаправляет на страницу входа. Функции входа, сессии и выхода предоставляются с помощью пакетов NPM: "passport", "express-session" и "passport-local". На странице входа в систему после ввода имени пользователя и пароля выполняется POST-вызов конечной точки "login" выполняется POST-вызов конечной точки. Конечная точка создается с помощью "express" и "passport", никакой контроллер не принимает участия в авторизации. Обработчик, зарегистрированный на конечной точке "login", проверяет имя пользователя и пароль и в случае правильных учетных данных инстанцируется сессия входа.

Выход из системы осуществляется с помощью "паспортного" метода "logout", который присоединяется к объекту запроса при последующих запросах.

**Контроллер поддержки**

Контроллер поддержки отвечает за отображение страницы поддержки, сохранение и получение сообщений. Основной метод "index" находит все разговоры в базе данных и отображает страницу со списком бесед. На отрисованной странице вызывается метод "messages" с идентификатором первой беседы. Метод запрашивает в базе данных все сообщения в конкретной беседе и возвращает отрисованный список, который затем отображается в браузере. Последним методом, представленным в контроллере поддержки, является "saveNewMessage". Как можно догадаться из названия, он отвечает за сохранение сообщений в базе данных.

**Контроллер турнира**

Контроллер турниров содержит логику для поиска, редактирования, создания и удаления турниров. Базовый метод "index" перенаправляет на метод "manage", который создает представление для списка турниров с полями поиска. Когда страница открывается на клиенте, вызывается метод "get". вызывается. По умолчанию метод находит все турниры, созданные пользователем, и возвращает отрендеренный список клиенту. Этот же метод используется с разными параметрами при поиске определенных турниров с помощью поисковых полей. Пользователь может нажать на любой турнир в списке, чтобы получить подробную информацию. Информация запрашивается с помощью метода "detail". Он получает данные и затем отправляет клиенту визуализированное представление подробной информации о запрашиваемом турнире. Представление с подробной информацией имеет четыре кнопки для управления выбранным турниром. Можно переключить активный статус турнира с помощью метода "toggleActive", переключить видимость турнира с помощью метода "toggleVisible".

Турнир так же может быть удален с помощью метода "remove". Последняя кнопка отвечает за перенаправление на страницу редактора для редактирования конкретного турнира. Редактор используется как для создания, так и для редактирования турниров. Метод "Create" отвечает за отображение редактора турнира. Если передать id турнира в метод то в окне отображается информация о выбранном турнире. В противном случае вид отображается с пустыми полями. Последним, но не менее важным, является метод "save", который отвечает за сохранение созданного или отредактированного турнира.

**Модели**

Каждая модель расширяет основной класс "AppModel". Родительский класс имеет абстракции методов для каждой операции с базой данных, например, find, findOne, distinct, create, insert, update и т.д. Поэтому контроллеры выполняют запросы к базе данных только с использованием методов определенные в "AppModel". Например, метод, отвечающий за вставку документов в базу данных, показан ниже:

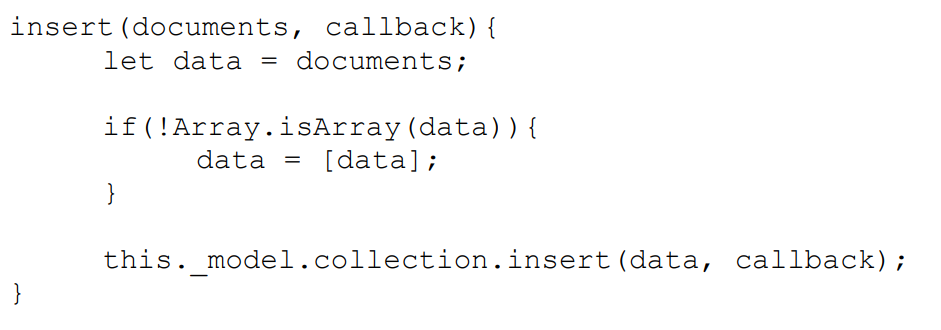


Рисунок n - метод "insert" класса основной модели.

Рисунок 11 демонстрирует одну из абстракций, представленных в классе "AppModel". Как видно, метод принимает два параметра: документ или массив документов, который необходимо вставить в базу данных, и обратный вызов, который будет вызван с результатом данных по завершении операции. Первая пара строк в методе проверяет, является ли предоставленная переменная переменная documents является одним документом или массивом, и если переменная не является массивом, преобразует ее в массив с одним элементом.

Последняя строка вызывает метод "insert" с предоставленными данными и обратный вызов на соответствующую коллекцию, используя модель mongoose модель. Модель Mongoose создается с помощью метода "\_initiate" в "AppModel". Тело метода инициализации выглядит следующим образом:

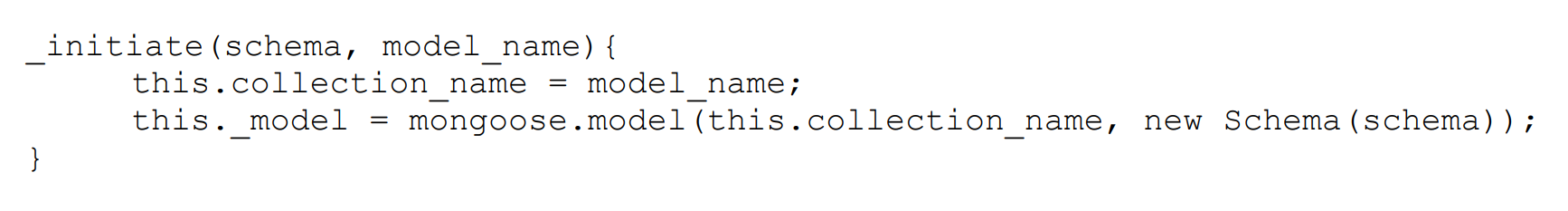


Рисунок n - Метод "\_initiate" класса основной модели.

На рисунке 12 показана реализация метода "\_initiate" основного класса модели. Метод принимает в качестве аргументов схему mongoose и имя модели.

Первая строка присваивает имя модели переменной класса с именем "collection\_name". Вторая и последняя строка отвечает за инициирование модели mongoose, используя предоставленную схему и имя модели имя. Инициированная модель хранится в свойстве "\_model" класса. Предоставленная схема может быть определена в двух местах. Во-первых, если модель не требует никаких специфических методов, то схема должна быть определена в файле "Schemas", который находится в директории каталоге "system". И в этом случае нет необходимости создавать отдельный класс для модели. С другой стороны, если модель требует некоторых методов, которые используются исключительно

которые используются исключительно в этой модели, они должны быть определены как отдельный класс, расширяющий класс "AppModel". класс. Если модель относится ко второй категории, то схема должна быть определена в свойстве свойстве "schema" класса. Схема для одной из моделей показана ниже:

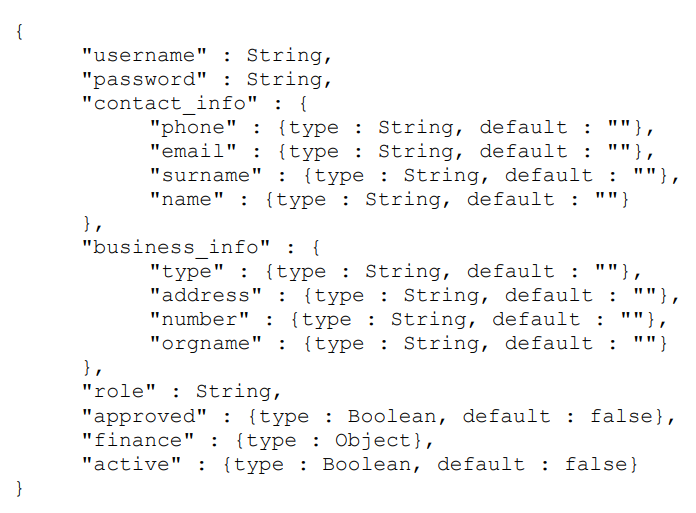


Рисунок n - Схема для модели пользователя.

Схема, показанная на рисунке 13, относится к пользовательской модели. Она описывает каждое поле, его тип и значение по умолчанию. Во-первых, у пользователя есть имя пользователя и пароль. Во-вторых, пользователь имеет роль, например, администратор, организация или клиент. Затем, пользователь может быть одобрен и активирован, поэтому необходимы статусы одобрен и активен. И, наконец, есть личная информация: контактная, деловая и платежная информация, которая управляется в кабинете контроллера.

**Представления**

**Представления** в проекте создаются с помощью шаблонизатора EJS. Существует несколько типов представлений в приложении, описанном в данной работе. Во-первых, общие представления, такие как "layout.ejs", "body.ejs", "javascript\_global.ejs" и т.д. Во-вторых, представления, связанные с конкретным контроллером и его методом, например, в каталоге под названием "cabinet" есть

есть представление под названием "info.ejs", которое является представлением по умолчанию для метода "info" в контроллере cabinet. И, наконец, есть отдельные элементы, которые используются в представлениях, связанных с контроллерами.

Представления имеют следующую общую структуру и поток. Некоторый контроллер отображает определенное представление. Каждое представление определяет макет, с которым это представление должно использоваться, используя следующего синтаксиса EJS:

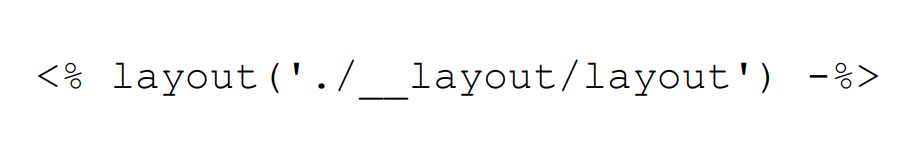


Рисунок n - Определение разметки EJS.

На рисунке 13 показан код для определения макета представления. Файл "layout.ejs", описанный выше, включает файл "styles\_global.ejs", который содержит все CSS, используемые в приложении глобально. В зависимости от статуса аутентификации файл "layout.ejs" включает в себя либо "body.ejs", либо "body\_no\_auth.ejs". Оба представления, в свою очередь, включают файл "javascript\_global.ejs", который определяет все библиотеки JavaScript, используемые в проекте, такие как "jquery", "toastr", "bootstrap" и др. А представление "body.ejs" также включает файл "left\_menu.ejs", который создает

навигационное меню.

Большая часть строк в представлениях занята необработанной HTML-разметкой, и только часть содержит синтаксис EJS синтаксис для рендеринга некоторых динамических данных. Пример кода для небольшой части представления чата поддержки чата можно увидеть ниже:

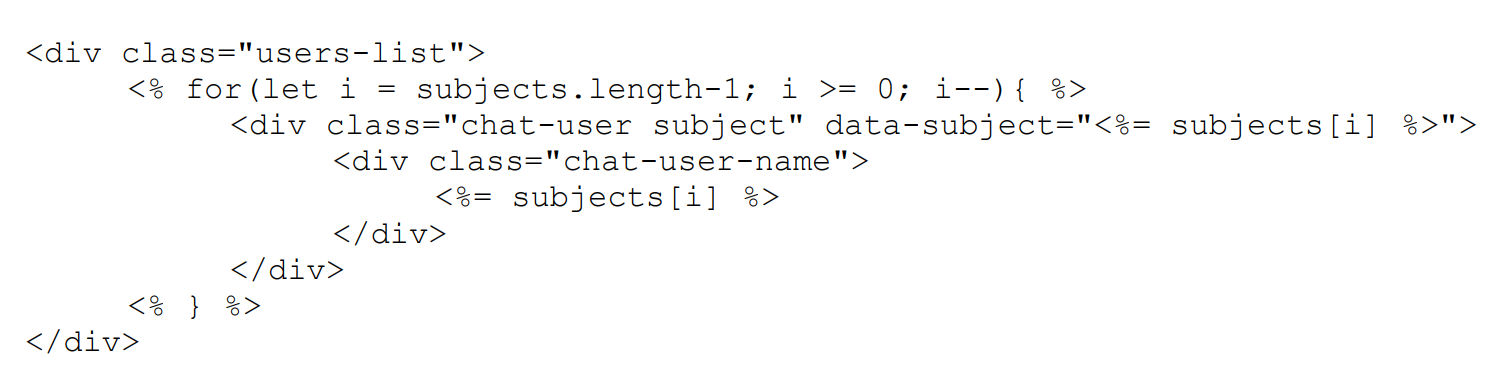


Рисунок n – Пример кода EJS для списка бесед.

Рисунок 14 демонстрирует способ вставки данных в HTML с использованием синтаксиса EJS. Пример кода, показанный на рисунке 14, создает список всех бесед в чате поддержки. Как видно, код EJS обернут в теги "<% %>", которые называются "скриплеты". Внутри него находится цикл перебора тем разговоров. Весь HTML, который находится внутри цикла, будет создаваться для каждой итерации. Внутри цикла значение каждой темы разговора вставляется в двух местах с помощью тегов "<%= %>", которые выводят значение в шаблон. Таким же образом в проекте создаётся весь динамический контент.