МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.Т. ТРУБИЛИНА»

Факультет прикладной информатики

Кафедра компьютерных технологий и систем

Обухов Владимир Павлович

|  |  |
| --- | --- |
| ДОПУСКАЕТСЯ К ЗАЩИТЕ | ДОПУСКАЕТСЯ К ЗАЩИТЕ |
| Руководитель образовательной программы | Зав. кафедрой |
| *канд. техн. наук, доцент* | *канд. техн. наук, доцент* |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лукьяненко Т. В. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_Лукьяненко Т. В. |
| «\_\_\_\_» \_\_\_\_\_\_\_\_\_2024 г. | «\_\_\_\_» \_\_\_\_\_\_\_\_\_2024 г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

РАЗРАБОТКА ИНТЕРНЕТ-ВИТРИНЫ СЕТИ МАГАЗИНОВ ТАБАЧНЫХ ИЗДЕЛИЙ ДЛЯ ИП СИМОНОВ А.Н., Г. КРАСНОДАР

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность «Создание, модификация и сопровождение

информационных систем, администрирование баз данных»

Руководитель:

канд. физ.-мат. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лаптев С.В.

**Краснодар**

**2024**

**Министерство сельского хозяйства Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ ИМЕНИ И.Т. ТРУБИЛИНА»**

ФАКУЛЬТЕТ ПРИКЛАДНОЙ ИНФОРМАТИКИ

Кафедра компьютерных технологий и систем

УТВЕРЖДАЮ

зав. кафедрой

компьютерных технологий  
и систем КубГАУ

канд. техн. наук, доцент  
\_\_\_\_\_\_\_\_\_\_ Т. В. Лукьяненко

«13» 11 2023 г.

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

Направление 09.03.02 Информационные системы и технологии (бакалавриат)

студенту учебной группы: ИТ2002

Обухову Владимиру Павловичу

(фамилия, имя, отчество)

1. Тема работы «Разработка интернет-витрины сети магазинов табачных изделий для ИП Симонов А.Н., г. Краснодар»

2. Цель работы – Разработать интернет-витрину сети магазинов табачных изделий для ИП Симонов А.Н.

3. Исходные данные и методические указания

Методические рекомендации по подготовке к государственной итоговой аттестации. / сост. С. В. Лаптев, В.Н. Лаптев, Т.В. Лукьяненко – Краснодар : КубГАУ, 2023. – 79 с.

4. Содержание выпускной квалификационной работы: анализ предметной области, разработка концептуальной модели, выбор СУБД для реализации базы данных, разработка физической модели БД, интернет-витрины с использованием языка программирования C#, языка разметки HTML в среде Microsoft Visual Studio 2022 и СУБД SQLite; расчет экономической эффективности от внедрения проекта.

5. Перечень основной рекомендуемой литературы:

1. Базы данных : учебник / Е. В. Фешина, В. В. Ткаченко. – Краснодар: КубГАУ,   
   2020. – 172 с.

Наименование рекомендуемой литературы:

1. Методики и модели проектирования и разработки информационных систем : монография / Т. А. Крамаренко, Т. В. Лукьяненко. – Краснодар : КубГАУ, 2018. –   
   173 с.

Задание рассмотрено на заседании кафедры (протокол № 4 от «13» ноября 2023 г.)

Руководитель: канд. физ.-мат. наук, доцент Лаптев Сергей Владимирович

ученое звание, ученая степень, фамилия, имя, отчество (подпись)

Задание к исполнению принял: «13» ноября 2023 г.

**Реферат**

Выпускная квалификационная работа содержит 60 страниц, 42 рисунка, 9 таблиц, 10 используемых источников.

Ключевые слова: API, REST, BLAZOR, ИНТЕРНЕТ-ВИТРИНА, WASM, C#, ENTITY FRAMEWORK, .NET, JWT, BEARER.

Цель работы: разработка интернет-витрины сети магазинов табачных изделий для ИП Симонов А.Н.

Объект исследования – сеть магазинов табачных изделий «KaifHouse» (ИП Симонов А.Н.).

Предмет исследования – процесс разработки и реализации бизнес-проекта интернет-витрины в сегменте B2C.

Результат работы: после проведения анализа предметной области была найдена потребность и закрыта с помощью создания веб-приложения. Информационная система была построена по общим принципам проектирования веб-приложений.

Оглавление

[ВВЕДЕНИЕ 5](#_Toc198716989)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc198716990)

[1.1 Характеристика организации ООО «ДОМБИЛЕТ» 7](#_Toc198716991)

[1.2 Описание основных информационных потоков 7](#_Toc198716992)

[1.3 Описание функциональной модели 10](#_Toc198716993)

[1.4 Техническая сущность задачи 12](#_Toc198716994)

[2 ПРОЕКТНЫЙ РАЗДЕЛ 15](#_Toc198716995)

[2.1 Обзор средств разработки веб-приложения 15](#_Toc198716996)

[2.2 Обоснование построения архитектуры веб-приложения 15](#_Toc198716997)

[2.3 Проектирование физическое модели данных 15](#_Toc198716998)

[3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 16](#_Toc198716999)

[3.1 16](#_Toc198717000)

[4 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ 17](#_Toc198717001)

[Расчет трудоемкости разработки 17](#_Toc198717002)

[ЗАКЛЮЧЕНИЕ 19](#_Toc198717003)

[СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 20](#_Toc198717004)

# ВВЕДЕНИЕ

Выход современных компаний на цифровые платформы становится ключевым фактором конкурентоспособности, позволяя сократить операционные издержки, расширить аудиторию и оптимизировать бизнес-процессы. Особенно это актуально для сферы продажи билетов на мероприятия, где скорость, удобство и доступность играют решающую роль в выборе клиентов. Переход в онлайн-пространство не только минимизирует расходы на аренду помещений и персонал, но и открывает возможности для автоматизации продаж, персонализации предложений и круглосуточного обслуживания.

Рост популярности онлайн-покупок среди пользователей обусловлен такими преимуществами, как экономия времени, прозрачность цен, мгновенный доступ к актуальной информации о событиях и возможность сравнения вариантов. Однако продажа билетов имеет свою специфику: необходимость интеграции с системами бронирования, соблюдение законодательных требований (например, возврат средств, защита персональных данных, возрастные ограничения для отдельных мероприятий), а также обеспечение безопасных транзакций. Эти аспекты требуют тщательной проработки при создании онлайн-платформы.

Актуальность темы выпускной квалификационной работы обусловлена потребностью консьерж-сервиса ООО «Домбилет» в цифровизации своих услуг. Разработка интернет-витрины позволит компании снизить затраты на содержание офисов, увеличить охват аудитории за счёт онлайн-продвижения, а также улучшить качество обслуживания клиентов благодаря автоматизации процессов бронирования и оплаты.

Целью проекта является исследование и создание эффективной интернет-витрины для продажи билетов на мероприятия, адаптированной под специфику ООО «Домбилет».

Для достижения цели поставлены следующие задачи:

* Функциональное моделирование предметной области: анализ текущей деятельности компании; выбор методов концептуального моделирования; формирование требований к интернет-витрине; сравнительный анализ существующих решений; проектирование оптимизированного бизнес-процесса «как должно быть».
* Логическое проектирование платформы: выбор технологий для моделирования системы; разработка логической структуры интернет-витрины; проектирование базы данных; определение требований к аппаратно-программному обеспечению.
* Физическое проектирование: выбор архитектуры и ПО для реализации; разработка физической модели данных; описание функционала веб-интерфейса; оценка экономической эффективности внедрения.

Структура работы включает:

* Аналитический раздел — изучение деятельности ООО «Домбилет», анализ рынка онлайн-продажи билетов и нормативно-правовых требований.
* Проектный раздел — обоснование выбора технологий, проектирование информационной и программной архитектуры платформы.
* Технологический раздел — описание реализации веб-интерфейса, интеграции с платёжными системами и модулями бронирования.
* Экономический раздел — расчёт затрат на разработку, оценка эффективности внедрения и потенциальной окупаемости проекта.

Практическая значимость работы заключается в создании адаптированного под нужды ООО «Домбилет» решения, которое повысит конкурентоспособность компании, увеличит объём продаж и оптимизирует управление мероприятиями. Результатом проекта станет функциональная интернет-витрина, обеспечивающая удобный выбор событий.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Характеристика организации ООО «ДОМБИЛЕТ»

Основной деятельностью рассматриваемого предприятия является как розничная, так и оптовая торговля билетами на различные развлекательно-досуговые мероприятия. Торговля осуществляется через оффлайн-точку и при помощи сети сайтов ООО «ДОМБИЛЕТ»

Предприятие основано в 2020 году в городе Краснодар, изначально реализовывало билеты через офлайн точку, со временем увеличивая масштабы.

В настоящее время консьерж-сервис ООО «ДОМБИЛЕТ» реализует билеты на следующие виды мероприятий: футбольные, хоккейные, волейбольные матчи, F1, концерты, выставки, экскурсии, цирковые и театральные представления.

Работа по реализации проекта ведётся на основании внутренних регламентов и взаимодействия с партнёрами, обеспечивающими поставку данных о мероприятиях. Общая структура компании предполагает четкое распределение обязанностей между подразделениями, что позволяет эффективно управлять процессами онлайн-продаж.

Организационная структура ООО «ДОМБИЛЕТ» включает следующие ключевые отделы: отдел продаж, отдел контента, отдел разработки, юридический отдел и бухгалтерию. Каждый из этих подразделений выполняет специфические функции, обеспечивая бесперебойную работу платформы. Отдел продаж играет ключевую роль в деятельности ООО «ДОМБИЛЕТ», обеспечивая взаимодействие с клиентами на всех этапах покупки билетов. Основные функции отдела включают обработку входящих заказов, консультирование клиентов по вопросам выбора мероприятий, оформления билетов и способов оплаты. Сотрудники отдела продаж также отвечают за оперативное решение возникающих вопросов, таких как возврат или обмен билетов, а также информируют клиентов о текущих акциях и специальных предложениях. Кроме того, отдел занимается анализом клиентских запросов, что позволяет выявлять потребности аудитории и передавать данные другим подразделениям для улучшения сервиса.

Отдел контента отвечает за информационное наполнение и продвижение платформы. Его задачи включают создание и публикацию актуального контента о предстоящих мероприятиях, таких как описание событий, даты, места проведения и стоимость билетов. Контент-менеджеры работают над оформлением карточек мероприятий, чтобы информация была максимально понятной и привлекательной для пользователей. Важной частью работы отдела является SEO-оптимизация: специалисты подбирают ключевые слова, оптимизируют тексты и метаданные, чтобы повысить видимость сайта в поисковых системах. Также отдел контента взаимодействует с отделом продаж для создания рекламных материалов и рассылок, направленных на привлечение новых клиентов.

Отдел разработки обеспечивает техническую поддержку и развитие платформы ООО «ДОМБИЛЕТ». Основные функции включают разработку и внедрение новых функциональных возможностей. Программисты и технические специалисты отдела следят за стабильностью работы сайта, устраняют возможные ошибки и обеспечивают защиту данных пользователей. Кроме того, отдел разработки проводит регулярное тестирование платформы, чтобы гарантировать её бесперебойную работу даже при высоких нагрузках, например, во время старта продаж билетов на популярные мероприятия.

Юридический отдел отвечает за правовое сопровождение деятельности компании. Его специалисты следят за тем, чтобы все процессы онлайн-продаж соответствовали действующему законодательству, включая законы о защите прав потребителей, правила обработки персональных данных и требования к дистанционной торговле. Кроме того, юридический отдел занимается урегулированием возможных споров с клиентами или партнёрами, представляя интересы компании.

Бухгалтерия, как неотъемлемая часть компании, отвечает за ведение финансового учёта и подготовку отчётности. Основные задачи включают расчёт и начисление заработной платы сотрудникам, учёт доходов от продаж билетов и контроль расходов на развитие платформы. Бухгалтеры обеспечивают своевременное перечисление налогов и сборов, а также готовят финансовые документы для предоставления в налоговую инспекцию и другие контролирующие органы.

Организационная структура компании построена по линейному принципу, где основные управленческие решения принимает руководитель компании. В штате насчитывается 24 сотрудников, включая специалистов по продажам, контент-менеджеров, SEO-оптимизаторов, разработчиков, юристов и бухгалтеров. Правильное распределение задач между отделами способствует повышению эффективности работы платформы, независимо от масштаба проекта.

Анализ деятельности компании включает оценку эффективности онлайн-продаж и технического состояния платформы. Это необходимо для определения потребностей в доработке и улучшении сервиса. Схема организационной структуры ООО «ДОМБИЛЕТ» представлена на рисунке 1.

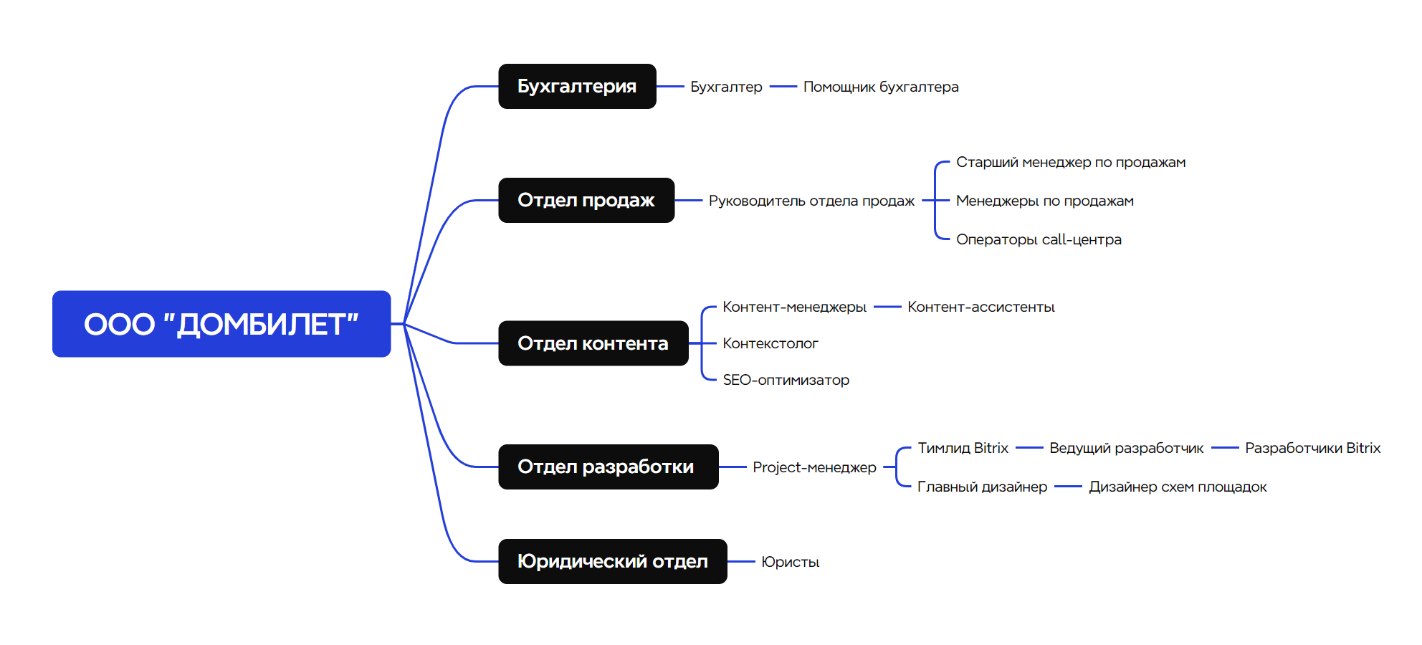


Рисунок 1 – Организационная структура ООО «ДОМБИЛЕТ»

## Описание основных информационных потоков

Информационные потоки – это перемещение информации между сотрудниками и подразделениями предприятия. Они необходимы для осуществления процессов и принятия решений. Информационные потоки между структурными подразделениями предприятия, между отдельными сотрудниками подразделений, а также с другими организациями и частными лицами, характеризуют работу организации и качество ее, поэтому к изучению информационных потоков должно быть привлечено особое внимание.

Основные информационные потоки в консьерж-сервисе по продаже билетов включают следующие процессы:

* Получение информации от партнеров и организаторов:

Данные о доступных билетах, датах и условиях мероприятий, квотах на места, изменениях в программах событий, а также сведения о VIP-доступе или дополнительных услугах (например, экскурсии, трансферы, парковочные места).

* Внутренние операционные потоки:

Согласование заказов между сотрудниками, управление бронированием, контроль доступности мест, синхронизация данных с системами партнеров, планирование расписаний мероприятий.

* Данные о продажах и бронировании:

Информация о проданных билетах, типах мероприятий (концерты, театры, спортивные события), категориях клиентов (корпоративные, VIP, частные лица), выбранных способах оплаты и дополнительных услугах.

* Взаимодействие с клиентами:

Обработка запросов на подбор мероприятий, консультации по программам, помощь в оформлении групповых заказов или возврате билетов. Сбор обратной связи: отзывы о качестве обслуживания, замечания по организации событий, запросы на персональные услуги.

* Аналитика и отчетность:

Мониторинг спроса на разные категории событий, прогнозирование пиковых нагрузок, оценка эффективности рекламных кампаний и партнерских программ, формирование индивидуальных предложений для постоянных клиентов.

Эффективное управление этими потоками позволяет:

* Оперативно обновлять данные о доступности билетов, избегая двойного бронирования.
* Персонализировать сервис, учитывая предпочтения клиентов (например, предпочтение определенных мест или жанров мероприятий).
* Оптимизировать взаимодействие с организаторами, сокращая время на согласование деталей.
* Повысить лояльность клиентов за счет оперативного решения вопросов (возвраты, замена билетов).
* Увеличить прибыль за счет перекрестных продаж дополнительных услуг и анализа востребованных событий.

Акцент делается на скорости обработки запросов, точности информации и интеграции с внешними системами (платежные сервисы, мобильные приложения клиентов, платформы организаторов), что особенно важно в условиях высокой конкуренции на рынке билетных услуг.

## Описание функциональной модели

Целью моделирования является упорядочение информации о компании и ее бизнес-процессах в виде наглядной графической структуры, удобной для аналитической обработки данных. Модель должна отображать организацию бизнес-процессов, детали их реализации и порядок документооборота.

Программа Ramus предназначена для описания, анализа и построения моделей бизнес-процессов. Она поддерживает создание организационных диаграмм, диаграмм с плавательными дорожками, а также интегрированных функциональных моделей на основе методологий IDEF0, IDEF3 и DFD. В Ramus можно задавать стоимость, время выполнения и частоту для каждого процесса. Эти параметры впоследствии суммируются по всем функциональным блокам, что помогает определить общую стоимость затрат, выявить слабые места в технологических цепочках и определить центры избыточных расходов.

Кроме того, Ramus интегрируется с инструментом ERwin для согласования функциональных и информационных моделей. Функциональная модель используется для описания текущих бизнес-процессов компании и представления идеального сценария работы. Методология IDEF0 предполагает создание иерархической системы диаграмм, представляющих отдельные элементы системы. В рамках IDEF0 система рассматривается как набор взаимодействующих функций или операций.

Такая ориентация исключительно на функции является основополагающей, поскольку анализ проводится независимо от объектов, с которыми эти функции работают. Это обеспечивает более точное моделирование логики и взаимосвязей процессов в организации. В IDEF0 модель представляет собой текстовое и графическое описание системы, предназначенное для ответа на заранее поставленные вопросы.

Моделирование системы в IDEF0 начинается с определения контекста — самого общего уровня описания системы в целом. После построения контекстной диаграммы выполняется функциональная декомпозиция: система делится на подсистемы, каждая из которых детализируется отдельной диаграммой. Этот процесс продолжается до достижения необходимой степени детализации, и каждый фрагмент системы отображается на собственной диаграмме декомпозиции. В Ramus модель интерпретируется как совокупность операций, каждая из которых оперирует определённым набором данных.

Операции (Activity) обозначают процессы, функции или задачи, выполняемые за определённый период и имеющие конкретные результаты. Они изображаются в виде прямоугольников и должны иметь чёткие названия, выраженные отглагольными существительными, отражающими действие. Стрелки (Arrow) иллюстрируют взаимодействия между операциями и представляют информацию, выраженную существительными. В IDEF0 выделяют четыре типа стрелок:

Вход (Input) — материалы или данные, которые используются или трансформируются операцией для получения результата (выхода);

Управление (Control) — правила, стратегии, процедуры или стандарты, которыми руководствуется операция;

Выход (Output) — материалы или данные, производимые операцией;

Механизм (Mechanism) — ресурсы, выполняющие операцию, такие как персонал компании.

Правильное расположение стрелок на контекстной диаграмме критически важно, так как они описывают взаимодействие системы с внешней средой. Неправильное их размещение может сделать модель непригодной для использования заказчиком.

Моделирование бизнес-процессов организации включает два этапа: структурный и детальный. Структурное моделирование выполняется в нотации IDEF0 с использованием Ramus. Сначала необходимо разобраться в функционировании ключевых процессов компании, определить приоритеты её эффективности и на этой основе создать контекстную диаграмму как основу модели. Она позволяет сформировать общее представление о работе предприятия, понять суть процессов и перейти к следующему этапу на основе полученных выводов.

Модель данных представляет собой абстракцию, которая, применённая к конкретным данным, позволяет пользователям и разработчикам интерпретировать их как информацию — данные, дополненные взаимосвязями. Модель бизнес-процессов в Ramus строится в виде контекстной диаграммы, отражающей все связи, как внутренние, так и внешние.

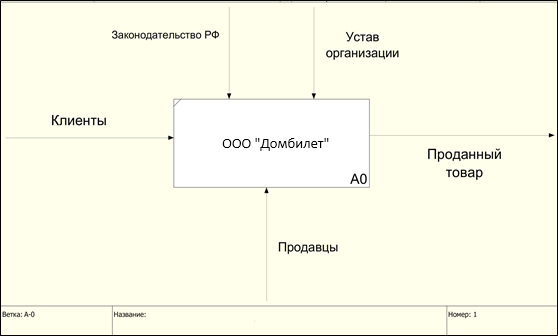
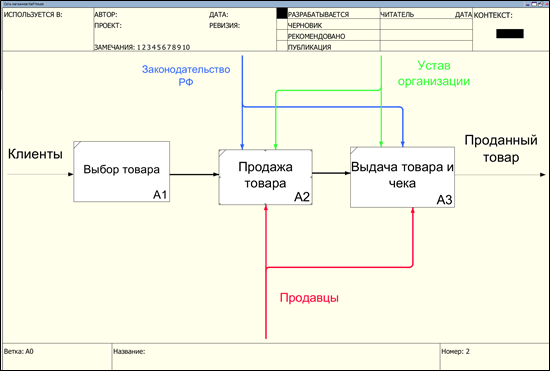
На рисунке 2 представлена модель ООО «ДОМБИЛЕТ»:

Рисунок 2 – модель ООО «ДОМБИЛЕТ»

Выходной поток один – это «Проданный товар». Управляющих потока два – это «Законодательство РФ» и «Устав организации». «Законодательство РФ» влияет на деятельности сети магазинов. «Устав организации» - правила корпоративной этики определенной сети магазинов. Процесс «Продажи товара» не может протекать без клиентов.

На втором уровне декомпозиции процесс «Продажа товаров» делится на три подпроцесса: «Выбор товара», «Продажа товара», «Выдача товара и чека» (рисунок 3).

Выбор товара зависит от желаний и предпочтений клиента. После просмотра он обращается к менеджеру для продажи билетов и последующей их выдаче.

Рисунок 3 – Декомпозиция модели ООО «ДОМБИЛЕТ»

## Техническая сущность задачи

Разработка интернет-витрины заключается в создании и запуске веб-приложения, которое представляет компанию и ее услуги в цифровой среде. Процесс включает выбор доменного имени, разработку дизайна, программирование функциональных возможностей, тестирование и размещение приложения в интернете. Главная цель – привлечение потенциальных клиентов и предоставление им удобного, информативного ресурса для изучения продукции или услуг компании.

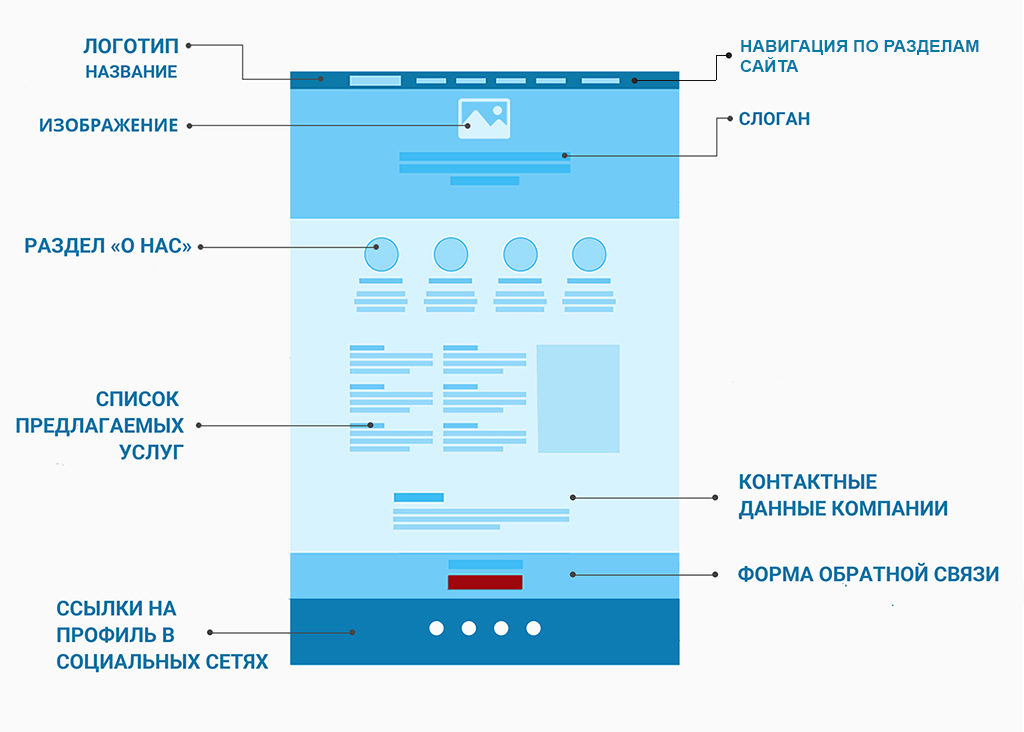
Интернет-витрина представляет собой веб-приложение с детализированным каталогом товаров, включающим описания и изображения. Обязательными элементами таких платформ являются структурированный каталог с категориями и подкатегориями, отображающий доступные товары. Пример интерфейса интернет-витрины приведен на рисунке 4.

Рисунок 4 –пример интерфейса интернет-витрины

Визуальное оформление каталога может варьироваться: от древовидной структуры до выпадающих или многоуровневых меню. Важной частью является система регистрации пользователей, которая позволяет каждому клиенту создать персональную «корзину» для добавления выбранных товаров и оформления заказов. Система также анализирует поведение пользователя в каталоге, что помогает в дальнейшем адаптировать ассортимент и рекомендации, оптимизируя взаимодействие с клиентом.

Такой подход сокращает временные и ресурсные затраты, повышает удовлетворенность клиентов и партнеров, а также укрепляет рыночные позиции компании.

Веб-приложение должно обеспечивать выполнение следующих задач:

* размещение и редактирование информации о событиях;
* добавление товаров в корзину;
* организация прямой коммуникации с клиентами через заказы и форму обратной связи;
* обеспечение удобной и интуитивной навигации.

Веб-приложение представляет собой сложную систему, состоящую из взаимосвязанных модулей, каждый из которых отвечает за определенные функции. В функционал входит просмотр каталога товаров и возможность бронирования билетов на мероприятия, например, спортивные матчи. Все процессы автоматизированы благодаря специально разработанному веб-приложению, что упрощает взаимодействие с пользователями и повышает эффективность работы платформы.

При создании приложения будут применены современные технологии и инструменты, такие как Python, FastAPI, HTML, CSS, JavaScript, Swagger. Взаимодействие между модулями будет осуществляться через API и RESTfull API, что обеспечит быструю обработку запросов и надежную работу всей системы.

# ПРОЕКТНЫЙ РАЗДЕЛ

## Обзор средств разработки веб-приложения

Веб-приложение — это программное обеспечение, которое функционирует в веб-браузере, предоставляя пользователям доступ к различным функциям и сервисам через интернет. Основное преимущество веб-приложений заключается в их универсальной доступности: они работают на любом устройстве с подключением к интернету, будь то компьютер, планшет или смартфон, без необходимости установки дополнительного программного обеспечения. Кроме того, веб-приложения легко обновляются, что позволяет быстро внедрять новые функции и исправлять ошибки, а также масштабируются, что делает их подходящими как для индивидуальных пользователей, так и для крупных организаций. Они обеспечивают удобство использования, кроссплатформенность и минимизируют затраты на развертывание.

Средства разработки веб-приложений играют ключевую роль в создании современных, надежных и функциональных решений. Они предоставляют разработчикам инструменты, библиотеки и фреймворки, которые упрощают процесс создания, тестирования и поддержки приложений. Выбор подходящего инструмента зависит от целей проекта, уровня подготовки команды и требований к производительности, масштабируемости и удобству разработки. Ниже представлен расширенный обзор популярных средств разработки веб-приложений с более детальным описанием их возможностей и особенностей.

Обзор средств разработки веб-приложений

1. HTML, CSS и JavaScript. Эти три технологии составляют основу веб-разработки и используются практически во всех веб-приложениях:

* HTML (HyperText Markup Language): отвечает за структуру и содержимое веб-страницы. С помощью HTML создаются элементы, такие как заголовки, параграфы, списки, таблицы и формы. Последняя версия, HTML5, добавила поддержку мультимедиа, семантических тегов (например, <article>, <section>) и API для работы с графикой и геолокацией.
* CSS (Cascading Style Sheets): используется для стилизации и оформления веб-страниц. CSS позволяет задавать цвета, шрифты, расположение элементов и адаптивный дизайн для разных устройств. Современные возможности, такие как Flexbox и Grid, упрощают создание сложных макетов, а CSS-препроцессоры (например, SASS или LESS) ускоряют разработку за счет переменных и миксинов.
* JavaScript: добавляет интерактивность и динамическое поведение. С его помощью создаются анимации, обработка событий (например, кликов или ввода данных), асинхронные запросы к серверу (AJAX) и сложные пользовательские интерфейсы. Современные стандарты ES6+ включают такие функции, как стрелочные функции, деструктуризация и модули, что делает код более читабельным и эффективным.

Эти технологии являются универсальными и необходимы для работы с любым фреймворком или библиотекой, так как они составляют основу веб-разработки.

1. Angular — это мощный фреймворк, разработанный Google, предназначенный для создания одностраничных приложений (SPA). Он использует TypeScript, что обеспечивает строгую типизацию и упрощает поддержку крупных проектов. Основные особенности:

* Компонентная архитектура: приложение разбивается на независимые компоненты, каждый из которых отвечает за свою часть интерфейса и логики.
* Двусторонняя привязка данных: изменения в данных автоматически отражаются в интерфейсе, и наоборот, что упрощает синхронизацию.
* Встроенные инструменты: Angular CLI (интерфейс командной строки) автоматизирует создание проектов, добавление компонентов и тестирование. Также фреймворк включает модули для маршрутизации, работы с формами и HTTP-запросами.
* Поддержка крупных приложений: благодаря строгой структуре и инструментам для управления зависимостями Angular идеально подходит для сложных корпоративных проектов.

Angular требует определенного времени на изучение из-за своей сложности, но его мощь и гибкость делают его популярным выбором для больших команд.

1. FastAPI — современный Python-фреймворк для создания высокопроизводительных API. Он построен на асинхронной библиотеке Starlette и использует аннотации типов Python для упрощения разработки. Особенности:

* Асинхронная поддержка: FastAPI использует возможности Python asyncio, что обеспечивает высокую производительность при обработке запросов, особенно для приложений реального времени.
* Автоматическая документация: благодаря поддержке OpenAPI и Swagger, FastAPI автоматически генерирует интерактивную документацию API, что упрощает тестирование и интеграцию.
* Простота и скорость разработки: интуитивный синтаксис и поддержка аннотаций типов позволяют быстро создавать надежные API.
* Широкая применимость: подходит для создания RESTful API, микросервисов и приложений, требующих высокой скорости обработки запросов.

FastAPI особенно популярен в проектах, где требуется высокая производительность и автоматическая документация, таких как микросервисы, IoT-приложения и системы машинного обучения.

1. React — это JavaScript-библиотека, созданная Facebook, которая используется для разработки динамичных и масштабируемых пользовательских интерфейсов. Она фокусируется на создании компонентов, которые можно переиспользовать в разных частях приложения. Особенности React:

* Виртуальный DOM: React использует виртуальную копию DOM, что ускоряет обновление интерфейса, так как изменения применяются только к необходимым элементам.
* Односторонняя передача данных: данные передаются от родительских компонентов к дочерним, что делает код более предсказуемым и легким для отладки.
* JSX: синтаксис, похожий на HTML, который позволяет писать интерфейсы прямо в JavaScript-коде, упрощая разработку.
* Экосистема: React легко интегрируется с библиотеками, такими как Redux или Zustand для управления состоянием, и инструментами, такими как Next.js для серверного рендеринга.

React популярен благодаря своей гибкости и большому сообществу, что делает его подходящим как для небольших, так и для крупных проектов.

1. Vue.js — это прогрессивный JavaScript-фреймворк, который сочетает простоту и мощь. Он идеально подходит для создания одностраничных приложений и интерактивных интерфейсов. Особенности:

* Простота интеграции: Vue можно использовать как для полного приложения, так и для отдельных компонентов на существующих страницах.
* Реактивность: данные автоматически синхронизируются с интерфейсом благодаря реактивной системе.
* Компактный синтаксис: синтаксис Vue интуитивно понятен, что делает его доступным для начинающих разработчиков.
* Экосистема: включает Vue Router для маршрутизации, Vuex для управления состоянием и Nuxt.js для серверного рендеринга.

Vue.js ценится за баланс между простотой и функциональностью, что делает его популярным среди разработчиков, ищущих легкий в освоении инструмент.

1. Node.js — это серверная среда выполнения JavaScript, которая позволяет создавать масштабируемые серверные приложения. Она основана на движке V8 от Google Chrome и ориентирована на асинхронные операции. Особенности:

* Асинхронная модель: Node.js использует событийно-ориентированную модель, что обеспечивает высокую производительность при обработке множества запросов.
* NPM (Node Package Manager): крупнейший репозиторий пакетов, который предоставляет доступ к тысячам библиотек для упрощения разработки.
* Создание API: Node.js часто используется с фреймворками, такими как Express, для создания RESTful API, которые обеспечивают взаимодействие между клиентской и серверной частями приложения.
* Кроссплатформенность: позволяет запускать серверные приложения на разных операционных системах.

Node.js идеально подходит для приложений реального времени, таких как чаты или стриминговые сервисы, благодаря своей скорости и масштабируемости.

1. Django — это высокоуровневый фреймворк на Python, предназначенный для быстрой разработки безопасных и масштабируемых веб-приложений. Он следует принципу DRY (Don’t Repeat Yourself) и предоставляет множество готовых решений. Особенности:

* ORM (Object-Relational Mapping): упрощает работу с базами данных, позволяя разработчикам использовать Python вместо SQL.
* Админ-панель: Django автоматически генерирует интерфейс администратора для управления содержимым.
* Безопасность: включает встроенные механизмы защиты от распространенных уязвимостей, таких как SQL-инъекции и XSS-атаки.
* Экосистема: поддерживает множество сторонних пакетов для добавления функциональности, например, Django REST Framework для создания API.

Django подходит для быстрого прототипирования и разработки сложных серверных приложений.

1. Laravel — это PHP-фреймворк, который упрощает создание современных веб-приложений благодаря элегантному синтаксису и мощным инструментам. Особенности:

* Eloquent ORM: упрощает взаимодействие с базами данных, предоставляя интуитивный синтаксис для запросов.
* Blade: шаблонизатор для создания динамических интерфейсов, который легко интегрируется с фронтенд-технологиями.
* Artisan: инструмент командной строки для автоматизации задач, таких как создание моделей, контроллеров и миграций.
* Безопасность и масштабируемость: Laravel предлагает встроенные функции для аутентификации, авторизации и защиты от атак.

Для реализации данного веб-приложения использовалась современная и удобная среда разработки Visual Studio Code (VS Code). Данная IDE отличается легкостью, высокой скоростью работы и широкой поддержкой расширений, что делает её отличным выбором для разработки как небольших, так и крупных проектов. Среди преимуществ VS Code можно выделить: кроссплатформенность (работает на Windows, Linux и macOS), встроенную поддержку систем контроля версий (например, Git), гибкую настройку интерфейса, огромное количество расширений для различных языков и технологий, а также удобные инструменты для отладки и автодополнения кода. Благодаря этим возможностям, разработка в VS Code становится максимально продуктивной и комфортной.

В качестве основного языка программирования был выбран Python. Python — это высокоуровневый, интерпретируемый язык программирования, который славится своей простотой, лаконичностью и читаемостью кода. Он идеально подходит для быстрой разработки веб-приложений благодаря большому количеству готовых библиотек и фреймворков, а также активному сообществу. Python поддерживает различные парадигмы программирования, включая объектно-ориентированное, процедурное и функциональное программирование, что позволяет гибко строить архитектуру приложения.

В качестве backend-фреймворка использовался FastAPI. FastAPI — это современный, быстрый (high-performance) веб-фреймворк для создания API на Python. Он позволяет создавать асинхронные приложения, что обеспечивает высокую производительность и масштабируемость. Среди ключевых преимуществ FastAPI: автоматическая генерация документации (Swagger и Redoc), поддержка асинхронного программирования, строгая типизация с помощью Pydantic, простота интеграции с различными базами данных и внешними сервисами. FastAPI отлично подходит для построения RESTful API и микросервисной архитектуры.

В проекте также активно используются основные библиотеки Python, такие как:

* Pydantic — для валидации и сериализации данных;
* Motor — асинхронный драйвер для работы с MongoDB;
* APScheduler — для планирования фоновых задач;
* Jinja2 — для шаблонизации HTML-страниц;
* Requests — для работы с HTTP-запросами;
* Selenium и webdriver\_manager — для автоматизации браузера и парсинга данных.

Для реализации клиентской части (frontend) используются стандартные веб-технологии: JavaScript, HTML и CSS. HTML отвечает за структуру страниц, CSS — за оформление и стилизацию, а JavaScript — за интерактивность и динамическое взаимодействие с пользователем. В проекте также применяются собственные JS-скрипты для работы с интерфейсом, а шаблоны HTML генерируются с помощью Jinja2.

Почему выбран именно такой стек технологий:

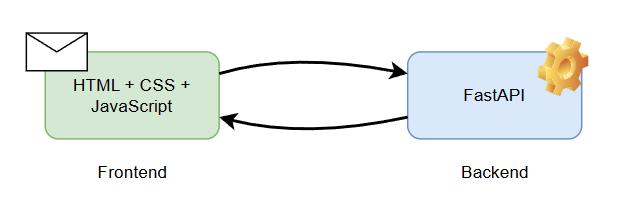
* Гибкость и скорость разработки. Python и FastAPI позволяют быстро создавать и масштабировать серверную часть приложения, а благодаря большому количеству библиотек можно легко интегрировать дополнительные функции.
* Высокая производительность. FastAPI обеспечивает отличную производительность за счет асинхронной обработки запросов, что особенно важно для современных веб-приложений с большим количеством пользователей.
* Простота поддержки и расширяемость. Код на Python легко поддерживать и расширять, а модульная структура проекта позволяет добавлять новые функции без значительных изменений в архитектуре.
* Современный и удобный интерфейс. Использование HTML, CSS и JavaScript позволяет создавать адаптивные и интерактивные пользовательские интерфейсы, которые хорошо работают на различных устройствах.
* Кроссплатформенность. Все используемые инструменты и библиотеки поддерживаются на разных операционных системах, что облегчает развертывание и сопровождение приложения.
* Активное сообщество и поддержка. Все выбранные технологии имеют большое сообщество, множество обучающих материалов и готовых решений, что ускоряет процесс разработки и решения возникающих задач.
* Таким образом, выбранный стек (VS Code + Python + FastAPI + современные библиотеки + JS/HTML/CSS) идеально подходит для реализации современного, масштабируемого и удобного веб-приложения, отвечающего всем требованиям по производительности, безопасности и удобству использования.

## Обоснование построения архитектуры веб-приложения

В современном программировании существует множество архитектурных подходов, однако для реализации данного веб-приложения была выбрана классическая клиент-серверная архитектура. Такой подход является одним из самых распространённых для построения распределённых систем и предполагает разделение приложения на две ключевые части:

* Клиент — это пользовательская часть системы, реализованная в виде веб-интерфейса с использованием HTML, CSS и JavaScript. Клиент отвечает за взаимодействие с пользователем: отображает данные, собирает вводимую информацию и отправляет запросы на сервер. В нашем случае клиентом выступает веб-браузер, через который пользователь получает доступ к функционалу приложения.
* Сервер — это внутренняя часть системы, реализованная на Python с использованием фреймворка FastAPI. Сервер принимает запросы от клиента, обрабатывает их, выполняет бизнес-логику, взаимодействует с базой данных и формирует ответы, которые возвращаются клиенту. Серверная часть отвечает за хранение и обработку данных, а также за реализацию всех основных процессов приложения.

На рисунке 5 показана архитектура проектируемой системы. Клиент-серверная архитектура разработки приложений предполагает, что пользователи обращаются к программному обеспечению через клиентский веб-интерфейс приложения.

Рисунок 5 – Клиент-серверная архитектура для проектируемой системы

Взаимодействие между клиентом и сервером осуществляется по протоколу HTTP с использованием REST API. Такой подход обеспечивает стандартизированный и удобный обмен данными между различными компонентами системы, а также позволяет легко интегрировать сторонние сервисы или расширять функционал приложения.

Для построения серверной части была выбрана концепция чистой архитектуры (Clean Architecture), предложенная Робертом Мартином. Чистая архитектура подразумевает разделение приложения на независимые слои, где каждый слой зависит только от внутренних слоёв и не зависит от внешних (рисунок 6). В нашем проекте это реализовано следующим образом:

* Внутренние слои отвечают за бизнес-логику, обработку данных и взаимодействие с базой данных (например, через асинхронный драйвер Motor для MongoDB).
* Внешние слои реализуют взаимодействие с пользователем (через шаблоны Jinja2 и статические файлы), а также обработку HTTP-запросов (через FastAPI).

Преимущества выбранной архитектуры для данного проекта:

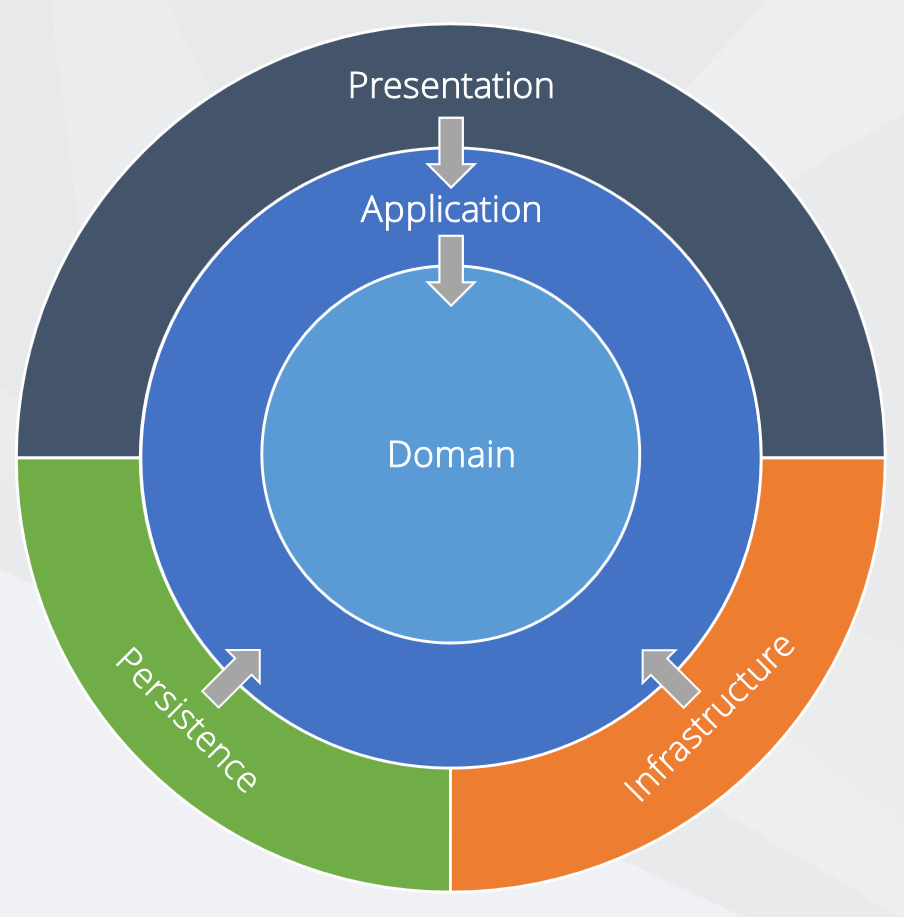
* Расширяемость: Модульная структура и слабая связность между компонентами позволяют легко добавлять новый функционал или изменять существующий без риска затронуть другие части системы.
* Гибкость: Благодаря разделению на слои, можно быстро адаптировать приложение под новые требования или интегрировать сторонние сервисы.
* Тестируемость: Каждый слой можно тестировать независимо, что ускоряет процесс разработки и повышает надёжность приложения.
* Удобство сопровождения: Чистая архитектура делает проект понятным и легко поддерживаемым, что особенно важно при работе в команде или при передаче проекта другим разработчикам.

Рисунок 6 – Чистая архитектура

Выбор клиент-серверной архитектуры в сочетании с чистой архитектурой и современным стеком технологий (Python, FastAPI, HTML, CSS, JavaScript) идеально подходит для реализации масштабируемого, надёжного и легко поддерживаемого веб-приложения, способного быстро адаптироваться к изменяющимся требованиям и обеспечивать высокий уровень пользовательского опыта.

## Проектирование логической модели данных

В процессе проектирования модели данных для веб-приложения по продаже билетов на спортивные мероприятия необходимо выделить основные информационные объекты (сущности), соответствующие требованиям нормализации, и определить связи между ними. В отличие от реляционных СУБД, в MongoDB данные хранятся в виде коллекций документов (JSON-подобных структур), что позволяет гибко моделировать связи и избегать избыточности данных.

Определение сущностей и их атрибутов:

Для идентификации сущностей были проанализированы бизнес-процессы системы. Каждая сущность отражает отдельный объект предметной области, обладающий уникальными характеристиками. Ниже приведён перечень основных коллекций (аналог таблиц) и их атрибутов, используемых в проекте:

Коллекция matches (Матчи)

Атрибуты:

\_id — уникальный идентификатор (ObjectId, ключевое поле)

id — числовой идентификатор матча (строка/число)

teams — название команд (строка)

date — дата проведения (строка)

time — время проведения (строка)

tournament — название турнира (строка)

image — изображение матча (строка, путь к файлу)

sectors — массив секторов (массив объектов)

is\_active — активен ли матч (логический)

slug — уникальный слаг (строка)

official\_url — ссылка на официальный сайт (строка)

Коллекция orders (Заказы/Билеты)

Атрибуты:

\_id — уникальный идентификатор (ObjectId, ключевое поле)

match\_id — идентификатор матча (строка, внешний ключ на matches.slug)

name — имя покупателя (строка)

email — email покупателя (строка)

phone — телефон покупателя (строка)

sector — сектор (строка)

row — ряд (число)

seat — место (число)

price — стоимость (число)

timestamp — дата и время заказа (datetime)

fan\_id — идентификатор болельщика (строка, опционально)

comment — комментарий (строка, опционально)

Коллекция admins (Администраторы)

Атрибуты:

\_id — уникальный идентификатор (ObjectId, ключевое поле)

username — имя пользователя (строка)

password\_encrypted — зашифрованный пароль (строка)

role — роль (строка: "admin" или "superadmin")

auth\_token — токен авторизации (строка)

Коллекция admin\_logs (Логи действий админов)

Атрибуты:

\_id — уникальный идентификатор (ObjectId, ключевое поле)

admin\_id — идентификатор администратора (ObjectId, внешний ключ на admins.\_id)

action — действие (строка)

details — детали действия (объект)

timestamp — дата и время (datetime)

Коллекция contact\_requests (Заявки на обратную связь)

Атрибуты:

\_id — уникальный идентификатор (ObjectId, ключевое поле)

name — имя пользователя (строка)

phone — телефон (строка)

email — email (строка)

timestamp — дата и время (datetime)

Проектирование физической модели данных

Цель этапа физического проектирования — описание конкретной реализации базы данных, размещаемой во внешней памяти компьютера. Это включает описание структуры хранения данных и эффективных методов доступа к данным. При логическом проектировании отвечают на вопрос — что надо сделать, а при физическом — выбирается способ, как это сделать.

В данном проекте для хранения информации используется база данных MongoDB, а взаимодействие с ней реализовано с помощью асинхронной библиотеки motor для Python. Такой подход позволяет эффективно работать с большими объёмами данных и поддерживать высокую производительность веб-приложения.

**Подключение к базе данных**

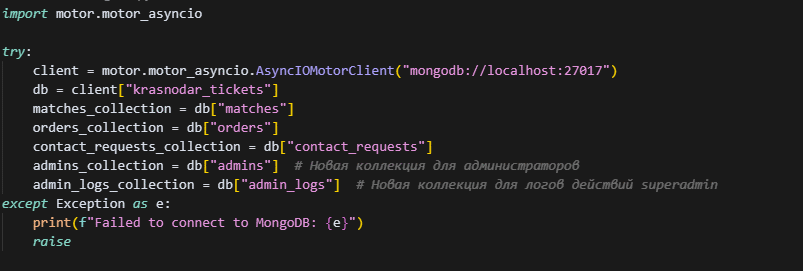
В файле app/database/mongodb.py реализовано подключение к MongoDB и определены основные коллекции (Рисунок )

Рисунок – Подключение к базе данных

Это позволяет централизованно управлять всеми коллекциями и использовать их в различных сервисах приложения.

**Моделирование данных**

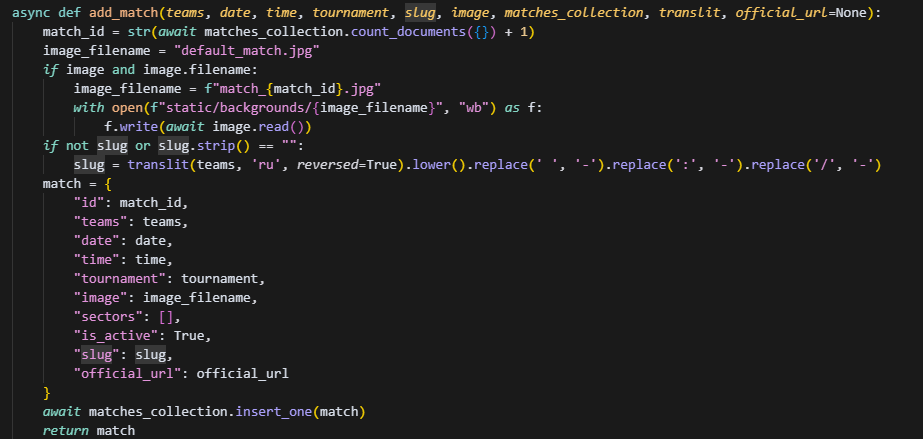
В MongoDB структура документов может быть гибкой, однако для поддержания целостности данных в проекте используются определённые схемы (структуры словарей Python). Например, структура матча и заказа. (Рисунок )

Рисунок – Пример структуры матча

Каждый матч содержит информацию о командах, дате, времени, турнире, изображении, секторах стадиона, активности, уникальном идентификаторе (slug) и ссылке на официальный источник.

**Пример структуры заказа**

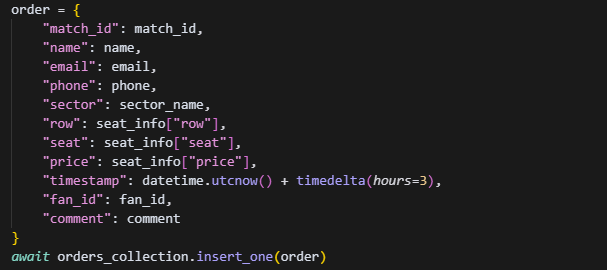
Оформление заказа реализовано в файле app/services/order\_service.py (Рисунок )

Рисунок – Пример структуры заказа

Заказ содержит информацию о выбранном матче, пользователе, месте, цене, времени оформления, а также дополнительные поля, такие как Fan ID и комментарий.

**Операции с данными**

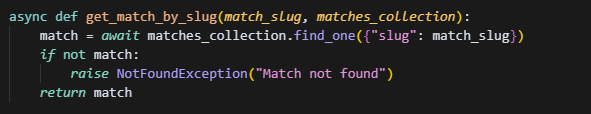
Для работы с данными реализованы сервисы, которые обеспечивают создание, поиск, обновление и удаление информации. Например, получение матча по slug (Рисунок )

Рисунок – Получение матча по slug

Аналогично реализованы функции для редактирования и удаления матчей, проверки доступности мест, обработки заказов и других операций.

**Преимущества выбранного подхода**

Гибкость: MongoDB позволяет легко изменять структуру документов без необходимости сложных миграций.

Масштабируемость: Асинхронная работа с базой данных обеспечивает высокую производительность при большом количестве одновременных запросов.

Централизованное управление: Все коллекции и операции с ними определены в отдельных модулях, что облегчает поддержку и развитие проекта.

После выполнения всех этапов физического проектирования структура базы данных полностью соответствует требованиям проекта. Использование MongoDB и асинхронного Python-кода позволяет эффективно управлять схемой данных и поддерживать синхронизацию между моделью кода и реальной базой данных. Такой подход обеспечивает надёжность, масштабируемость и удобство дальнейшего развития системы.

# ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 

# ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

## Расчет трудоемкости разработки

В проекте разрабатывается современная система онлайн-продажи билетов, реализуемых в сети магазинов ООО «Домбилет». В отличие от простой интернет-витрины, данная система обладает сложной архитектурой, поддерживает работу с большим количеством пользователей, интеграцию с внешними сервисами, а также реализует гибкую систему управления мероприятиями, секторами стадиона и заказами.

В связи с повышенной сложностью и масштабом проекта, для его реализации была сформирована команда из двух разработчиков, каждый из которых специализировался на определённых областях, но при этом оба обладали навыками как backend-, так и frontend-разработки. Такой подход позволил обеспечить высокое качество кода, быстрое внедрение новых функций и эффективное решение возникающих задач.

Разработка веб-приложения включала в себя следующие этапы:

* Анализ требований. Детальный анализ бизнес-требований, определение целей и задач проекта, сбор и формализация пользовательских сценариев, создание подробного плана реализации, оценка рисков и определение сроков выполнения работ. Особое внимание уделялось вопросам безопасности, масштабируемости и удобства для конечных пользователей.
* Проектирование архитектуры. Выбор многоуровневой архитектуры, определение основных компонентов системы (API, сервисы, базы данных, интерфейс администратора, клиентский интерфейс), проектирование взаимодействия между слоями, выбор оптимальной базы данных (MongoDB) для хранения информации о матчах, секторах, билетах и заказах. Были проработаны вопросы интеграции с внешними сервисами (например, для рассылки уведомлений и обработки платежей).
* Разработка back-end. Реализация сложной бизнес-логики, включая управление матчами, секторами, местами, обработку заказов, аутентификацию и авторизацию пользователей и администраторов, ведение логов действий, интеграцию с внешними API. Были реализованы асинхронные сервисы для обеспечения высокой производительности и масштабируемости. Особое внимание уделялось обработке ошибок, валидации данных и обеспечению безопасности пользовательских данных.
* Разработка front-end. Создание современного, адаптивного и интуитивно понятного пользовательского интерфейса с использованием современных технологий. Были разработаны отдельные интерфейсы для клиентов и администраторов, реализована интеграция с backend API, обеспечена поддержка клиентской аутентификации и авторизации, реализованы динамические элементы, анимации и переходы для улучшения пользовательского опыта. Особое внимание уделялось кроссбраузерности и мобильной адаптивности.
* Тестирование и отладка. Написание и выполнение модульных, интеграционных и end-to-end тестов для проверки корректности работы отдельных компонентов и системы в целом. Проводилось тестирование на различных устройствах и в разных браузерах, осуществлялась нагрузочная проверка для выявления и устранения узких мест в производительности.
* Развертывание приложения. Подготовка и автоматизация процесса сборки и деплоя приложения, настройка серверной инфраструктуры, развертывание приложения в облачной среде, настройка резервного копирования, мониторинга и журналирования для отслеживания производительности, ошибок и инцидентов безопасности. Были реализованы механизмы автоматического масштабирования и обновления приложения без простоя для пользователей.

Таким образом, проект представляет собой сложную, масштабируемую и надёжную систему, разработанную с учётом современных требований к безопасности, производительности и удобству использования. Командная работа двух специалистов позволила реализовать все этапы жизненного цикла разработки, обеспечить высокое качество продукта и его соответствие ожиданиям заказчика и конечных пользователей.

Список работ, необходимых для разработки, указан в таблице

Таблица 8 – Плановый список работ

|  |  |
| --- | --- |
| Вид работы | Период дней |
| Анализ требований | 3 дня |
| Проектирование архитектуры | 10 дней |
| Разработка back-end | 20 дней |
| Разработка front-end | 15 дней |
| Тестирование и отладка | 5 дней |
| Развертывание приложения | 2 дня |

## Определение плановой себестоимости проведения работ

Согласно статистике, представленной на веб-сайте «Хабр Карьера», заработная плата backend- и frontend-разработчиков начинается от 47 тыс. рублей в месяц и может достигать 413 тыс. рублей в месяц в зависимости от квалификации специалиста (график представлен на рисунке 42). Для реализации нашего проекта по созданию системы онлайн-продажи билетов для ООО «ДОМБИЛЕТ» была сформирована команда из двух специалистов: один backend-разработчик и один frontend-разработчик.

Средняя зарплата backend-разработчика уровня junior составляет около 98 тыс. рублей в месяц, а frontend-разработчика — порядка 87 тыс. рублей в месяц. Для расчёта стоимости работ примем среднюю зарплату по команде:

(98000 + 87000) / 2 = 92500 рублей в месяц на одного сотрудника.

Добавим к этой сумме надбавку 15% за будущее сопровождение и поддержку системы, а также учтём налогообложение и страховые взносы, которые в совокупности составляют 43%. В результате итоговая стоимость одного месяца работы одного сотрудника составит примерно 153000 рублей, а для двух сотрудников — 306000 рублей в месяц.

Исходя из трудозатрат, рассчитанных по этапам разработки (см. таблицу списка работ), общая стоимость разработки и внедрения системы онлайн-продажи билетов для ФК «Краснодар» составит 366 000 рублей. Эта сумма включает в себя не только оплату труда разработчиков, но и расходы на сопровождение, тестирование и развертывание системы.Также стоит рассмотреть расходы на внедрение системы. Список расходов представлен в таблице 9.

Таблица 9 – Список расходов на внедрение системы

|  |  |
| --- | --- |
| Статья расходов | Величина, руб. |
| Приобретение компьютерной техники | 70 000 |
| Установка и монтаж оборудования | 6 500 |
| Разработка дизайна | 20 000 |

Итоговая стоимость разработки и внедрения интернет-витрины сети магазинов «Kaif House» составляет 261 500 рублей.

* 1. Экономическая эффективность

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ