

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА
на тему
**«Сервис подбора и продажи билетов на
развлекательные мероприятия»**

Студент _____ Ежова Ксения Валерьевна
(подпись, дата) (ФИО полностью)

Руководитель _____ Иванчуков А. Г.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Ежова Ксения Валерьевна гр. 19ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Сервис подбора и продажи билетов на развлекательные мероприятия

Тема утверждена приказом ПГУ № _____ от " ____ " _____ 20 ____

2. Руководитель доцент кафедры «МОиПЭВМ» Иванчуков А.Г.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать сервис по подбору и продаже билетов на развлекательные мероприятия.

2) Среда выполнения клиентской части: ОС MacOS

Среда выполнения серверной части: ОС MacOS

Языковые средства разработки: JavaScript, C#, UML

Инструментальные средства разработки: Visual Studio, Visual Studio Code, MongoDB Atlas, Visual Paradigm, Argo UML, Draw.io, Project Libre

Технологии разработки: микросервисная архитектура, ООП, SPA

3) Основные функции web-приложения:

а) Отображение списка развлекательных мероприятий

б) Возможность выбирать места на развлекательные мероприятия

в) Возможность бронировать мероприятия

г) Добавление новых мероприятий

д) Отмена бронирования

4. Перечень подлежащих разработке
вопросов _____

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____

Л.Р. Фионова

Реферат

Пояснительная записка содержит 95 листов, 43 рисунка, 21 таблицу, 14 использованных источников и 7 приложений.

C#, МИКРОСЕРВИСЫ, REACT, WEB-ПРИЛОЖЕНИЕ, WEB-API, ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ, ИНФОРМАЦИОННЫЕ СИСТЕМЫ, РАЗВЛЕКАТЕЛЬНЫЕ МЕРОПРИЯТИЯ.

Объект разработки: Web-приложение, позволяющее просматривать и покупать билеты по проходящим мероприятиям, а также создавать собственные.

Цель работы: разработка программного обеспечения для просмотра, покупки и создания мероприятий.

Технологии разработки серверной части: C#, ASP.NET, Ocelot. Разработка велась в IDE – Visual Studio.

Технологии разработки клиентской части: JavaScript, React, Mobx. Разработка велась в IDE – Visual Studio Code.

Технологии разработки баз данных: MongoDB. Разработка велась в сервисе MongoDB Atlas.

Результаты работы: в процессе разработки проводилось изучение Web-API и создание микросервисов с помощью языка программирования C# и на основе их использования разработано Web-приложение по подбору и продаже билетов на развлекательные мероприятия.

					ПГУ 09.03.02 – 8БР191.7 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.	Ежова К.В.				Сервис подбора и продажи билетов на развлекательные мероприятия. Пояснительная записка	Лит.	Лист	Листов
Провер.	Иванчуков А.Г.						4	95
Реценз.						группа 19ВИ1		
Н. Контр.	Такташкин Д.В.							
Утверд.	Козлов А.Ю.							

Содержание

Введение	7
1 Анализ предметной области	9
1.1 Описание предметной области	9
1.2 Анализ программных продуктов-аналогов для подбора развлекательных мероприятий	10
1.2.1 Сайт «Кассир.Ру»	10
1.2.2 Сайт «Redkassa»	11
1.2.3 Сайт «Яндекс Афиша»	12
1.3 Сравнение аналоговых-систем для развлекательных мероприятий.....	13
1.4 Расчет бюджета разрабатываемого приложения	18
1.5 Анализ функциональных требований	21
2 Анализ технологий разработки	25
2.1 Язык моделирования UML	25
2.2 Язык программирования серверного приложения	25
2.3 Язык программирования клиентского приложения	27
2.4 Базы данных.....	27
3 Проектирование приложения	29
3.1 Проектирование базы данных	29
3.2 Проектирование серверной части	29
3.3 Проектирование клиентской части	37
4 Реализация	40
4.1 Реализация базы данных	40
4.2 Реализация серверной части	40
4.3 Реализация клиентской части	45
4.4 Реализация развертывания приложения	50
5 Тестирование информационной системы по подбору и продаже билетов на развлекательные мероприятия	52
5.1 Тестирование создания и редактирование развлекательного мероприятия	52

5.1.1	Классы эквивалентности.....	52
5.1.2	Функциональная диаграмма и таблица решений для информационной системы.....	54
5.2	Тестирование метрики валидации формы по созданию развлекательного мероприятия	57
5.3	Автоматизированное тестирование валидации формы по созданию мероприятия	59
	Заключение	61
	Список использованных источников	62
	Приложение А. Глоссарий.....	64
	Приложение Б. Исходный код клиентской части.....	66
	Приложение В. Исходный код серверной части	72
	Приложение Г. Описание баз данных.....	82
	Приложение Д. Диаграммы классов	86
	Приложение Е. Логические модели баз данных.....	89
	Приложение Ж. Тестирование приложения	92

Введение

В современном мире, с развитием Интернет-пространства, онлайн-бронирование стало неотъемлемой частью нашей жизни. Это относится не только к бронированию авиабилетов и отелей, но и к бронированию билетов на различные развлекательные мероприятия. С каждым годом все больше людей предпочитают бронировать билеты онлайн, вместо того чтобы стоять в очередях кассы.

До распространения Web-сайтов с бронированием, чтобы купить билет на развлекательное мероприятие, нужно было физически посетить кассу или агентство по продаже билетов. Это могло потребовать дополнительного времени и усилий, особенно если касса находилась в другом районе или городе. Кроме того, при покупке билетов лично в кассе не всегда было возможно выбрать места, которые были бы наиболее удобными для зрителя.

Сегодня ситуация изменилась благодаря развитию технологий. Теперь можно заказать билеты на мероприятия онлайн, используя сайты, мобильные приложения или социальные сети. В этом случае покупка билетов становится значительно более удобной и быстрой. Пользователи могут выбрать места на карте зала, а также сравнить различные предложения от разных поставщиков билетов, чтобы выбрать наиболее выгодное.

Кроме того, сегодня многие мероприятия также предлагают электронные билеты, которые можно распечатать или сохранить на мобильном устройстве. Это удобно, так как позволяет избежать необходимости физически получать билеты в кассе или на почте.

Таким образом, современные технологии значительно упрощают процесс покупки билетов на развлекательные мероприятия, делая его быстрым, удобным и доступным для большего числа людей. В этом контексте автоматизация онлайн-бронирования развлекательных мероприятий становится все более актуальной. Автоматизация процесса бронирования позволит пользователям

более удобно и быстро забронировать билеты на мероприятия в любое удобное для них время.

В данной работе рассмотрим возможности и преимущества автоматизации онлайн-бронирования развлекательных мероприятий, а также рассмотрим основные задачи, которые необходимо решить для создания системы автоматизации бронирования.

Для создания WEB-приложения, удобного пользователю, необходимо выяснить необходимые требования к разрабатываемой системе. Система должна быть интуитивно понятна, доступна на различных устройствах, а также в различных браузерах для охватывания большей категории людей.

Цель выпускной квалификационной работы – разработка информационной системы по подбору и продаже билетов на развлекательные мероприятия.

Объект исследования – информационная система в Интернет-пространстве.

Предмет исследования – подбор и продажа билетов на развлекательные мероприятия.

Задачи, поставленные для выполнения выпускной квалификационной работы:

- провести сравнительный анализ программных продуктов-аналогов;
- провести анализ требований;
- произвести проектирование клиентской и серверной частей для системы развлекательных мероприятий с использованием методологии UML;
- провести планирование бюджета, закладываемого на разработку информационной системы;
- реализовать информационную систему для подбора и продажи билетов на развлекательные мероприятия;
- протестировать клиентскую и серверную части на основе выбранного метода тестирования.

Выделенные требования к приложению позволят провести анализ.

1 Анализ предметной области

1.1 Описание предметной области

Информационная система по продаже и подбору билетов на развлекательные мероприятия – это комплекс программных средств, предназначенных для автоматизации процесса продажи билетов на различные развлекательные мероприятия, такие как концерты, театральные представления, киносеансы, спортивные события и так далее.

Основная цель системы – обеспечить быстрый и удобный доступ к информации о мероприятиях и продаже билетов, а также обеспечить автоматизацию процесса продажи билетов, что позволяет сократить время на ожидание в очереди в кассе или агентстве по продаже билетов.

Функциональность информационной системы по продаже и подбору билетов на развлекательные мероприятия может включать в себя:

- каталог мероприятий – система содержит информацию о различных мероприятиях, которые доступны для бронирования или покупки билетов;
- бронирование и покупка билетов – пользователи могут выбрать мероприятие и забронировать или купить билеты онлайн;
- электронные билеты – многие системы предоставляют возможность получения электронных билетов, которые можно распечатать или сохранить на мобильном устройстве;
- управление продажами – система может автоматически управлять процессом продажи билетов, обеспечивая точный учет проданных билетов и доступных мест в зале;
- избранные мероприятия – пользователь сможет откладывать бронирование выбранного мероприятия на будущее, добавляя в список избранных;

- аналитика – система может предоставлять аналитические данные о продажах билетов, а также о покупках, позволяющий в сформированном формате просмотреть данных;
- согласование мероприятий – каждый пользователь может предложить собственное мероприятие для добавления на сайт после согласования с администратором.

Для добавления новых мероприятий используется администрирование системы за счет разделения ролей пользователей. Так, для роли администратора будет доступен функционал создания, редактирования и удаления существующих мероприятий. Для пользователей данный функционал не будет доступен.

1.2 Анализ программных продуктов-аналогов для подбора развлекательных мероприятий

Существует множество программных продуктов-аналогов для подбора развлекательных мероприятий, которые могут помочь людям выбрать и забронировать билеты на мероприятия. Некоторые из наиболее популярных и широко используемых программных продуктов-аналогов опишем в данной работе.

1.2.1 Сайт «Кассир.Ру»

Кассир.Ру – web-приложение предоставляющая доступ к покупке билетов в театр, цирк, музеи, выставки, на концерт, фестивали или шоу в выбранном городе. Преимуществами данной системы является широкий спектр мероприятий по выбранному городу, система рекомендаций, удобство в выборе мест, а также новостная лента.

Главная страница со списком мероприятий представлена на рисунке 1.

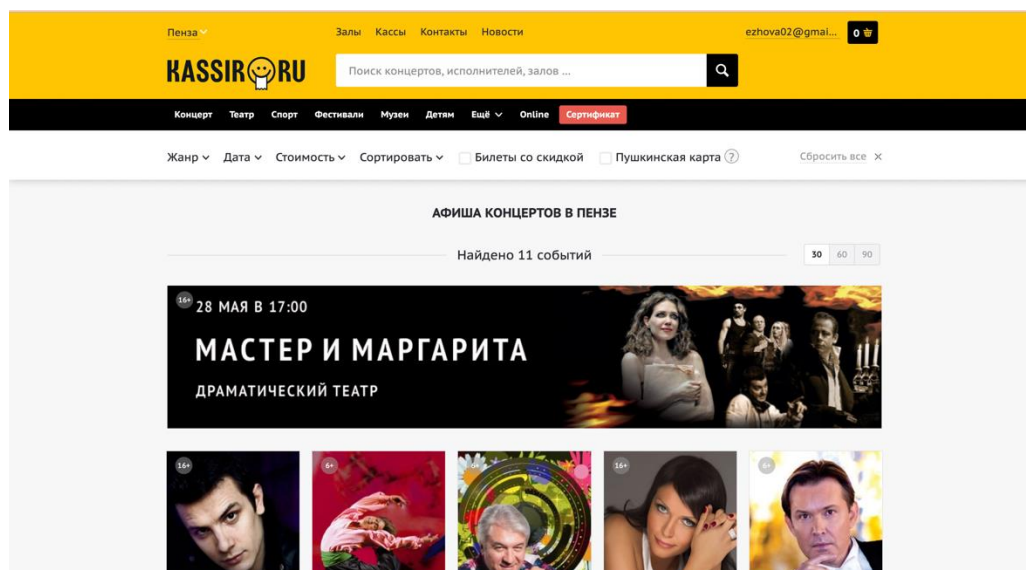


Рисунок 1 – Главная страница «Кассир.Ру»

Из недостатков можно выделить отсутствие подборок по интересам, отсутствие отображения места проведения на карте, добавление в избранное определенных мероприятий.

На сайте для backend части используется язык программирования PHP. Для frontend части используется jQuery. При использовании данной технологии зачастую идет повтор использования одного и того же кода в разных частях приложения. Данный подход устарел и снижает производительность сайта, ведя к долгой загрузке страниц.

1.2.2 Сайт «Redkassa»

Redkassa – web-приложение предоставляющая доступ к онлайн покупке билетов, а также в кассах в театр, на концерты, детские мероприятия и юмористические в ограниченном количестве городов. Преимуществами является доступ к карте внутри мероприятия, что облегчает пользователю нахождение проведения мероприятия. Недостатками сайта являются устаревший дизайн, присутствие сервисного сбора при покупке онлайн, отсутствие авторизации. Также сайт предоставляет билеты только в четырех городах России. Были найдены технические ошибки при использовании приложения.

Главная страница с концертами представлена на рисунке 2.

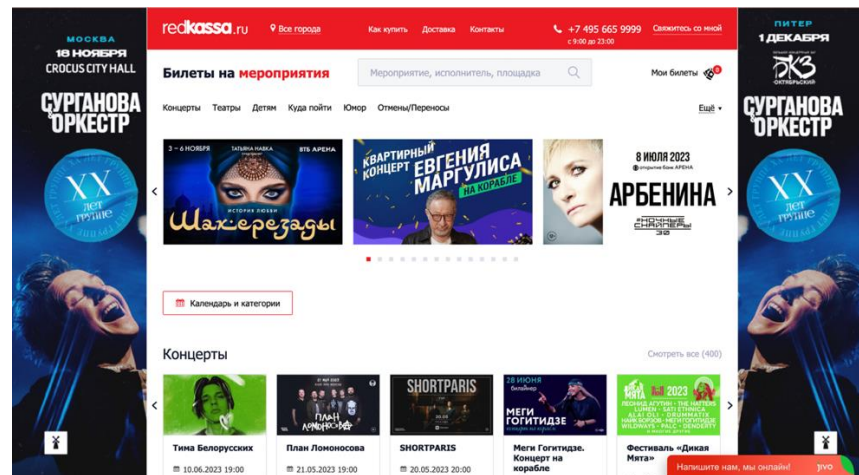


Рисунок 2 – Главная страница «Redkassa»

При выборе мест в зале, перед показом схемы есть выбор категории мест.

Для клиентской части был выбран Preact – библиотека схожая с React, отличием которой является меньший размер, но похожая функциональность. Web-сервер настроен на Nginx – это HTTP-сервер и обратный прокси-сервер, почтовый прокси-сервер, а также TCP/UDP прокси-сервер общего назначения.

1.2.3 Сайт «Яндекс Афиша»

Система по продаже и подбору мероприятий на события в большом списке городов. Доступны тематические подборки, представленные на рисунке 3. Данные подборки разделены по категориям и тематикам, представленным в выбранном городе.

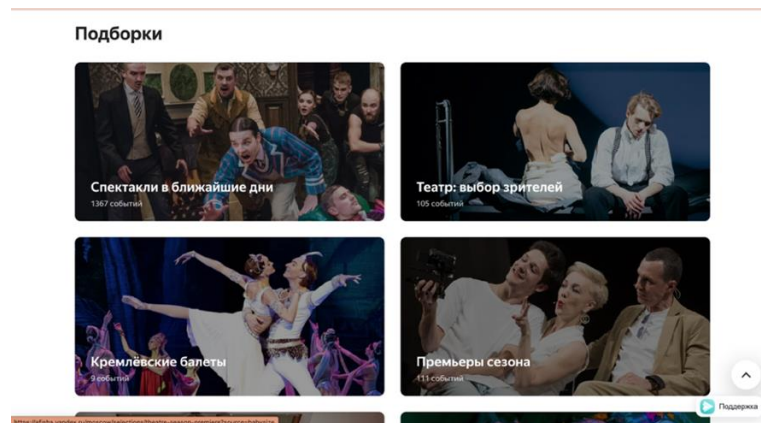


Рисунок 3 – Подборки по тематикам

Доступен тест, после прохождения которого персональная подборка по выбранному жанру и обстоятельствам.

В отличие от сайтов Кассир.Ру и Redkassa, при выборе билетов отображается полностью вся схема зала без деления на категории (партер/ложа/сцена). Это может вызвать неудобства при выборе на больших схемах залов.

В отличие от вышеперечисленных сервисов, данный сервис предоставляется не только в России, но и в Казахстане, Белоруссии. Но при этом, покупка билетов офлайн в кассах данная система не предоставляет.

Из минусов можно выделить отсутствие фильтрации по цене.

1.3 Сравнение аналоговых-систем для развлекательных мероприятий

При анализе различных сайтов по подбору развлекательных мероприятий, были выделены определенные характеристики, которые должны присутствовать в подобных системах.

Существует несколько критериев, которые можно использовать для сравнения сайтов по бронированию мероприятий. Некоторые из них включают в себя:

- доступность к бронированию;
- просмотр списка развлекательных мероприятий;
- разнообразие развлекательных мероприятий;
- расширение каталога мероприятий;
- аналитика данных.

Критерий «доступность к бронированию» для сайта оценивается по трёхбалльной шкале и отражает насколько легко на сайте доступна информация и оповещения о бронированиях развлекательных мероприятий. Описание оценок представлено в таблице 1.

Таблица 1 – Критерий оценки «доступность к бронированию»

Балл	Расшифровка характеристики оценки
0	на сайте сложно найти информацию о доступных для бронирования услугах, функции для бронирования неочевидны или отсутствуют, процесс бронирования запутанный, информация о бронировании не подтверждается на сайте и может не приходить на электронную почту.
1	на сайте можно найти информацию о доступных для бронирования услугах, функции для бронирования доступны, но могут быть не совсем ясны, процесс бронирования может быть запутанным, информация о бронировании может не подтверждаться на сайте, но будет отправлена по электронной почте.
2	на сайте легко найти информацию о доступных для бронирования услугах, все необходимые функции для бронирования явно выделены и легко доступны, процесс бронирования интуитивно понятен, информация о бронировании подтверждается на сайте или по электронной почте.

Критерий «просмотр списка развлекательных мероприятий» для сайта оценивается по трёхбалльной шкале и отображает возможность отображения списка мероприятия и работы с ним. Описание оценок представлено в таблице 2.

Таблица 2 – Критерий оценки «просмотр списка развлекательных мероприятий»

Балл	Расшифровка характеристики оценки
0	пользователи не могут просматривать список мероприятий на сайте, список мероприятий находится на другом ресурсе или не отображается вовсе.
1	пользователи могут просматривать список мероприятий на сайте, но отсутствует возможность фильтрации и сортировки, список доступен на сайте, но неудобен для использования.
2	пользователи могут просматривать список мероприятий на сайте, а также фильтровать и сортировать его по различным параметрам, список мероприятий на сайте легко доступен и удобен для использования.

Критерий «разнообразие мероприятий» для сайта оценивается на трехбалльной шкале и отражает насколько хорошо сайт обслуживает своих пользователей. Описание оценок представлено в таблице 3.

Таблица 3 – Критерий оценки «разнообразие мероприятий»

Балл	Расшифровка характеристики оценки
0	сайт предлагает ограниченный выбор мероприятий, возможно, только в определенных жанрах или тематиках
1	сайт предлагает некоторое разнообразие мероприятий, но может не иметь полного покрытия всех жанров и тематик
2	сайт предлагает широкий выбор мероприятий различных жанров и тематик, включая музыкальные концерты, спортивные события, театральные представления, кино и другие развлечения.

Критерий «расширение каталога мероприятий» для сайта оценивается по двухбалльной шкале и отражает возможность предлагать собственные мероприятия. Описание оценок представлено в таблице 4.

Таблица 4 – Критерий оценки «расширение каталога мероприятий»

Балл	Расшифровка характеристики оценки
0	сайт не предлагает возможность предлагать собственные мероприятия.
1	сайт предлагает возможность согласовывать собственные мероприятия пользователем через администратора сайта, которое в дальнейшем попадет в список всех мероприятий.

Критерий «аналитика данных» для сайта оценивается по двухбалльной шкале и отражает возможность пользователям просматривать общие сведения по совершенным покупкам, категориям, датам. Описание оценок представлено в таблице 5.

Таблица 5 – Критерий оценки «аналитика данных»

Балл	Расшифровка характеристики оценки
0	сайт не предлагает возможность просматривать аналитику и улучшать пользовательские подборки в зависимости от ранее совершенных действий пользователем.
1	сайт осуществляет систематический сбор и анализ данных о посетителях и их поведении на сайте. Анализ проводится с целью улучшения пользовательского опыта и повышения эффективности бизнес-процессов на сайте. Также пользователям доступна данная аналитика для просмотра с выводами в зависимости от их совершенных ранее действий.

Анализ на основе критериев проводился на следующих сайтах:

- Кассир.Ру;
- Redkassa;
- Яндекс Афиша.

После исследования функциональности и оценочных критериев, была составлена таблица результатов (таблица 6).

Таблица 6 – Результаты анализа

Критерий	Касса.Ру	Redkassa	Яндекс Афиша
Доступность к бронированию	2	1	2
Просмотр списка развлекательных мероприятий	2	2	2
Разнообразие мероприятий	2	1	2
Расширение каталога мероприятий	0	0	0
Аналитика данных	0	0	0

Результаты анализа в виде графика с накоплением представлены на рисунке 4.

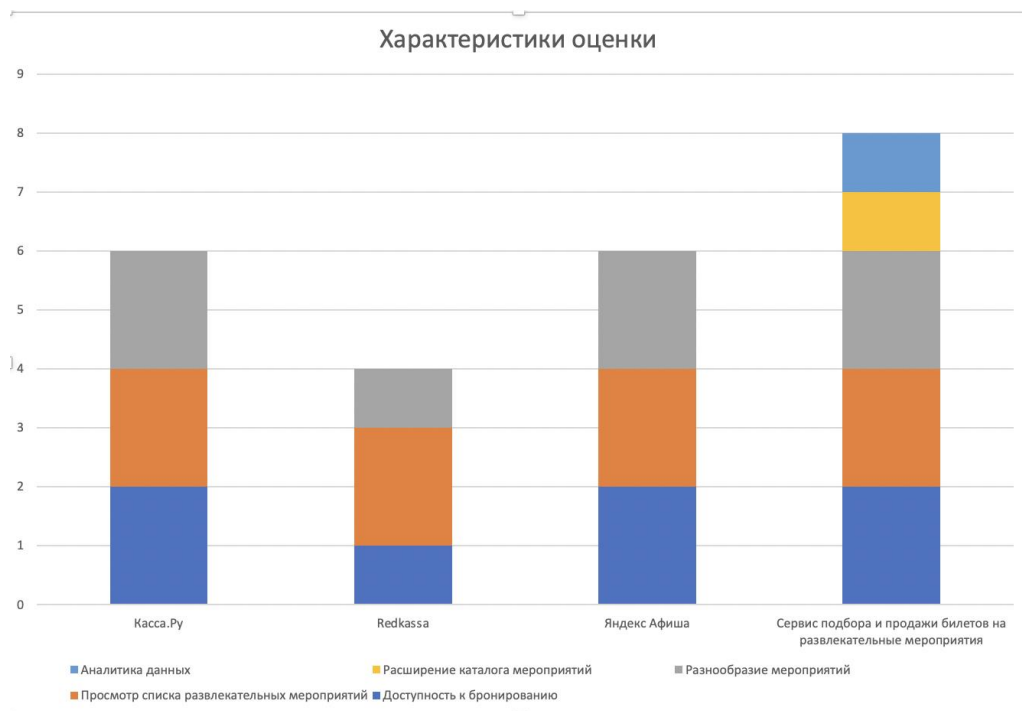


Рисунок 4 – График с накоплением на основе анализа

В представленных сервисах-аналогах используется работа с поставщиками мероприятий, либо они сами таковыми являются, интегрируясь с определенными концертными площадками. Существует множество мероприятий, которые проводятся на небольших площадках и узнать о них можно лишь напрямую зная о месте проведения, либо об организаторе. Это усложняет распространение таких мероприятий. Эту проблему поможет решить согласование каждым человеком, имеющий доступ к сервису для распространения и продажи собственных мероприятий, которые ранее были известны лишь ограниченному кругу людей.

Также для продавцов важно следить за покупками на их мероприятия. Для этого важно представление данных в удобном формате графиков, на которых наглядно можно увидеть какое из их мероприятий наиболее быстро продается и есть ли смысл увеличивать количество билетов на такие мероприятия. Для покупателей, аналитика так же имеет важность в отслеживании своих финансов и исследовании какие мероприятия для них наиболее продаваемые. Это позволит сервису выдавать более релевантные подборки.

В результате анализа было выяснено, что наиболее подходящей под требования аналоговой системой по продаже и подбору билетов на развлекательные мероприятия является Яндекс Афиша. Тем не менее каждая из этих систем имеет свою плюсы и недостатки, как в технической реализации, так и в функциональной. Так ни один из аналоговых сервисов не имеет функционала для пользователей, которые хотят продвигать собственные мероприятия. Также у аналоговых сервисов не представляется возможность анализировать собственные расходы на мероприятия в личном кабинете.

Наиболее проблемной системой является Redkassa, так как не предоставляет большой выбор городов для пользователей и рассчитана на ограниченную категорию людей. Дизайн системы устаревший, без доступности на разные категории пользователей.

1.4 Расчет бюджета разрабатываемого приложения

После анализа требований к приложению, функциональных частей, произведем расчет бюджета.

Для расчета бюджета на разработку web-приложения по развлекательным мероприятиям произведем декомпозицию с использованием методологии IDEF0. IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов [1].

Выделим следующие модули при моделировании ресурсов разработки:

- анализ предметной области;
- проектирование;
- реализация;
- тестирование;
- оформление пояснительной записки.

На выходе должны быть реализовано приложение, пояснительная записка и презентация.

Верхний уровень для IDEF0 по проекту «Выпускная квалификационная работа» представлена на рисунке 5.

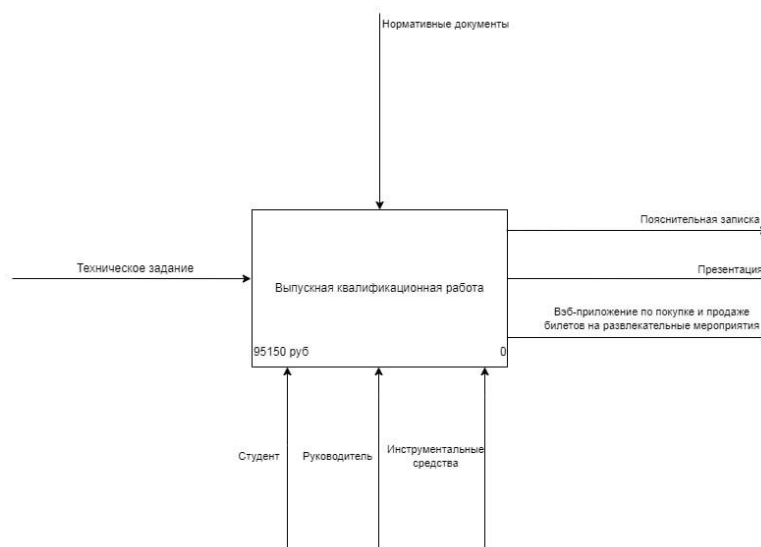


Рисунок 5 – Контекстная диаграмма, верхний уровень

Стрелкой управления являются «Нормативные документы», поступающие в верхнюю кромку модуля «Выпускная квалификационная работа».

Стрелкой входа является «Техническое задание», которая по итогу должна произвести на выход элементы «Пояснительная записка», «Презентация» и «Web-приложение по покупке и подбору билетов на развлекательные мероприятия».

Механизмами по данной методологии в описанном случае являются «Студент», «Руководитель» и «Инструментальные средства».

Было выделено на выпускную квалификационную работу 95 150 рублей.

Для описания модуля «Выпускная квалификационная работа» проведем декомпозицию, в которой выделим следующие модули:

- анализ предметной области – выделено 5950 рублей, который идет из технического задания;
- проектирование – выделено 9600 рублей, к выполнению которого приступают после анализа;
- реализация – выделено 52 800 рублей;
- тестирование – выделено 12 800 рублей;
- оформление пояснительной записки – выделено 14 000 рублей.

Декомпозиция представлена на рисунке 6.

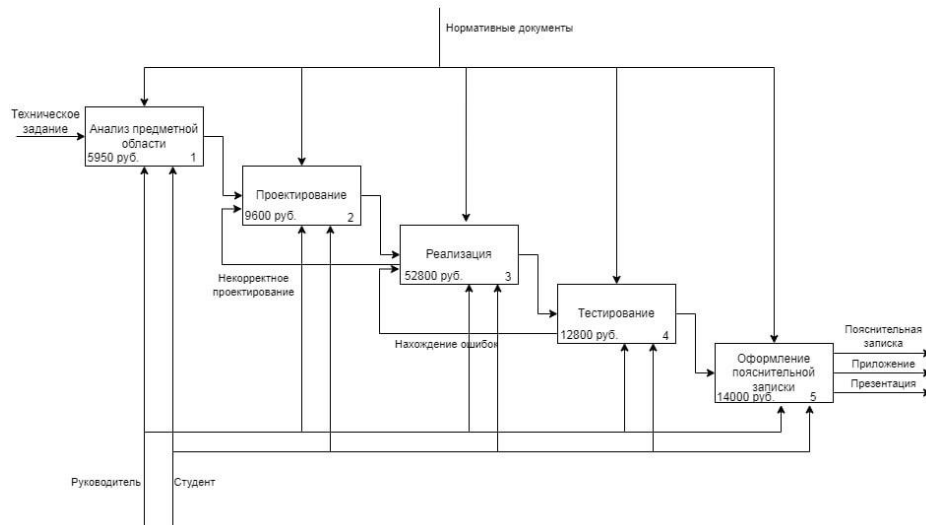


Рисунок 6 – Декомпозиция модуля «Выпускная квалификационная работа»

Для каждого модуля также выделен бюджет, затраченный на их реализацию. Подробнее опишем составленные траты с помощью программы ProjectLibre, позволяющий наглядно показать временные затраты, а также использованные ресурсы людей.

Опишем рабочие ресурсы, используемые для разработки приложения, а также выделим материальные ресурсы, использованные при работе (рисунок 7).

	⊕	Название	RBS	Тип	E-mail адрес	Ед. изм. материалов	Инициалы	Группировать	Максимальные единицы	Стандартная ставка	Ставка сверхурочных	Затраты на одно испол.	Начислять	Основной календарь
1	⊕	Project менеджер	Работа				P		100%	350 руб./час	0 руб./час	0 руб. Пропорционально	Пятидневка	
2	⊕	Frontend-разработчик	Работа				F		100%	300 руб./час	0 руб./час	0 руб. Пропорционально	Пятидневка	
3	⊕	Backend-разработчик	Работа				B		100%	300 руб./час	0 руб./час	0 руб. Пропорционально	Пятидневка	
4	⊕	Тестировщик	Работа				T		100%	250 руб./час	0 руб./час	0 руб. Пропорционально	Пятидневка	
5		Дизайнер	Работа				D		100%	250 руб./час	0 руб./час	0 руб. Пропорционально	Пятидневка	
6		Компьютер	Материал			шт	K			20000 руб.		0 руб. Пропорционально		
7		Принтер	Материал			шт	P			2000 руб.		0 руб. Пропорционально		
8		Бумага	Материал			шт	S			300 руб.		0 руб. Пропорционально		
9		Краски для принтера	Материал			шт	K			300 руб.		0 руб. Пропорционально		
10		Интернет	Материал			месц	I			700 руб.		0 руб. Пропорционально		

Рисунок 7 – Ресурсы для разработки

Каждый рабочий ресурс имеет почасовую оплату, составленную на основе средней зарплаты по выделенным категориям. Так, для Junior тестировщика средней зарплатой является 40000 рублей, что в пересчете на почасовую оплату при пятидневной рабочей неделе составляет 250 рублей в час.

Составим план по ранее описанным модулям опираясь на конкретные промежутки дат, а также на использованные в каждом модуле ресурсы (рисунок 8).

	⊕	Название	Продолжи...	Начало	Окончание	Стоимость	Предшествующие	Название ресурса
1	⊕	Анализ предметной области	3 дней?	06.04.23 8:00	10.04.23 17:00	5950 руб.		
2	⊕	Описание предметной области	0,375 дней?	06.04.23 8:00	06.04.23 11:00	1050 руб.		Project менеджер
3	⊕	Анализ программных требований	0,5 дней?	07.04.23 8:00	07.04.23 13:00	1400 руб.	2	Project менеджер
4	⊕	Сравнение аналоговых систем	0,25 дней?	08.04.23 8:00	10.04.23 10:00	700 руб.	3	Project менеджер
5	⊕	Расчет бюджета	1 день?	10.04.23 8:00	10.04.23 17:00	2800 руб.	4	Project менеджер
6	⊕	Проектирование	4 дней?	11.04.23 8:00	14.04.23 17:00	9600 руб.		
7	⊕	Проектирование клиентской части	1,5 дней?	11.04.23 8:00	12.04.23 13:00	3600 руб.	5	Frontend-разработчик
8	⊕	Проектирование серверной части	1,5 дней?	13.04.23 8:00	14.04.23 13:00	3600 руб.	7	Backend-разработчик
9	⊕	Проектирование хранения данных	1 день?	14.04.23 8:00	14.04.23 17:00	2400 руб.	8	Backend-разработчик
10	⊕	Реализация	22 дней?	15.04.23 8:00	16.05.23 17:00	52800 руб.		
11	⊕	Реализация серверной части	11 дней?	15.04.23 8:00	01.05.23 17:00	26400 руб.	9	Backend-разработчик
12	⊕	Реализация клиентской части	11 дней?	02.05.23 8:00	16.05.23 17:00	26400 руб.	11	Frontend-разработчик
13	⊕	Тестирование	6 дней?	17.05.23 8:00	24.05.23 17:00	12800 руб.		
14	⊕	Функциональное тестирование	2 дней?	17.05.23 8:00	18.05.23 17:00	4000 руб.	12	Тестировщик
15	⊕	Написание Unit-тестов	2 дней?	19.05.23 8:00	22.05.23 17:00	4000 руб.	14	Тестировщик
16	⊕	Правки	2 дней?	23.05.23 8:00	24.05.23 17:00	4800 руб.	15	Frontend-разработчик[50...
17	⊕	Оформление пояснительной записки	5 дней?	25.05.23 8:00	31.05.23 17:00	14000 руб.		
18	⊕	Написание отчета	5 дней?	25.05.23 8:00	31.05.23 17:00	14000 руб.	16	Project менеджер

Рисунок 8 – Структура с описанием модулей

Составим диаграмму Ганта на основе ранее описанного плана, иллюстрирующей временное планирование работы каждого ресурса (рисунок 9).

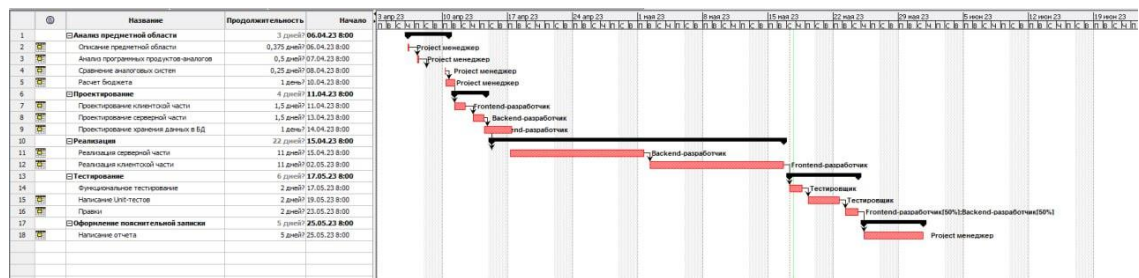


Рисунок 9 – Диаграмма Ганта

На основе проведенного анализа бюджета можно увидеть, что на проект потребуется 95 150 рублей, 313 часов работы, а также ресурсы Project менеджера, Frontend и Backend разработчиков и тестировщика.

1.5 Анализ функциональных требований

После проведения исследования по существующим web-приложениям выбранной предметной области, можно выделить функциональные требования, которые должны быть в данной системе.

Представим функциональные требования к приложению, используя диаграмму вариантов использования. Диаграмма вариантов использования – это вариант UML диаграмм, которая включает в себе актеров и их связь с вариантами использования [2].

Выделим актеров, которые будут участвовать в использовании создаваемого приложения:

- авторизованный пользователь – это человек, который ранее зарегистрировался, имеет собственный JWT-токен для использования приложения;
- не авторизованный пользователь – это человек, который не был зарегистрирован в системе, имеет возможность пользоваться лишь ограниченным функционалом;
- администратор – это человек, у которого есть собственный JWT-токен и роль, позволяющая создавать, редактировать, согласовывать и архивировать мероприятия.

Используя данных актеров, выделим варианты использования для диаграммы (рисунок 10).

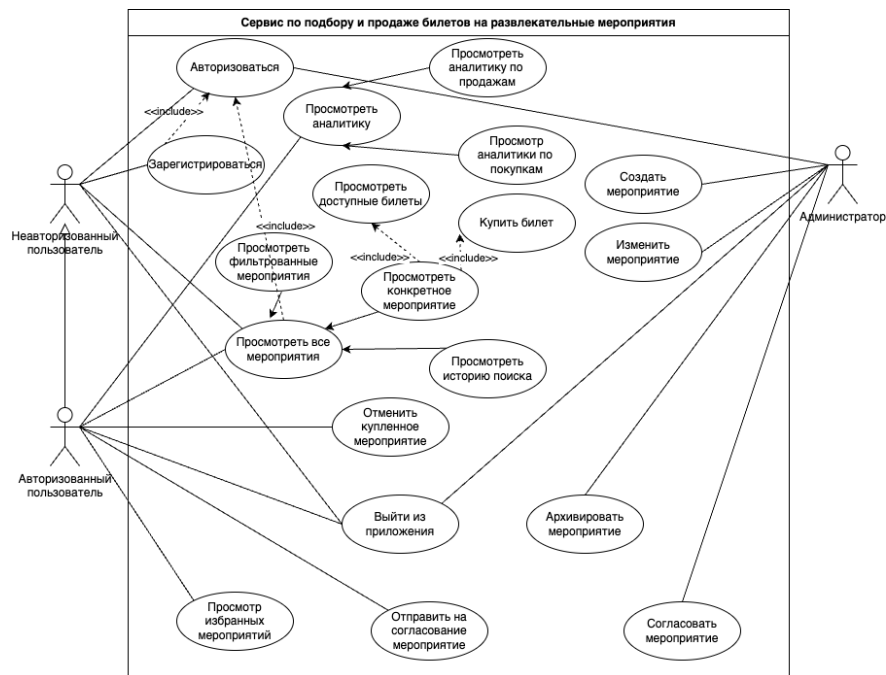


Рисунок 10 – Диаграмма вариантов использования

Актер «Авторизованный пользователь» является частным вариантом «Неавторизованного пользователя», в связи с чем наследуется от него.

Описание вариантов использования с действиями пользователя и последующей реакцией системы представлены в таблице 7.

Таблица 7 – Варианты использования

Действия пользователя	Реакция системы
Авторизоваться	Система перенаправит на главную страницу с мероприятиями, либо выдаст ошибку авторизации.
Зарегистрироваться	Система перенаправит пользователя на страницу авторизации, либо выдаст ошибку регистрации.
Просмотреть все мероприятия	Система отобразит мероприятия по категориям в списке на главной странице
Просмотреть историю поиска	Система отобразит 5 последних просмотренных мероприятий.
Просмотреть конкретное мероприятие	Система откроет страницу с подробной информацией по выбранному мероприятию, отобразит дату проведения, место, описание, категории мест, цену по категории и количество доступных билетов.

Продолжение таблицы 7

Действия пользователя	Реакция системы
Просмотреть доступные билеты	Система отобразит в списке на странице с конкретным мероприятием доступные категории мест.
Купить билет	После выбора мест и оплаты, билет появится в личном кабинете пользователя и будет доступен для скачивания PDF файл со счетом и основной информацией по мероприятию.
Отменить купленное мероприятие	Мероприятие больше не будет отображаться в личном кабинете с купленными мероприятиями.
Просмотреть фильтрованные мероприятия	Система выдаст список мероприятий по выбранным фильтрам.
Просмотреть избранные мероприятия	Система отобразит мероприятия на странице избранных, которые были выбраны как избранное.
Создать мероприятие	Система добавит мероприятие в общий список мероприятий, либо выдаст ошибку создания. Далее перенаправит на главную страницу.
Изменить мероприятие	Система изменит мероприятие, оно будет отображаться с другими данными, либо выдаст ошибку изменения. Далее перенаправит на главную страницу.
Архивировать мероприятие	Система больше не будет отображать мероприятие в списке всех мероприятий, либо выдаст ошибку архивирования.
Просмотреть аналитику	Система на странице профиля отобразит графики для аналитики.
Просмотреть аналитику по продажам	Система на странице профиля отобразит график по продажам мероприятий авторизованного пользователя.
Просмотреть аналитику по покупкам	Отобразится график с суммой цен по категориям купленных мероприятий.
Отправить на согласование мероприятие	Система перенаправит на страницу профиля с согласованиями и отобразит новое мероприятие в согласованиях со статусом «На согласовании», либо отобразится ошибка.
Согласовать мероприятие	Система перенаправит на страницу профиля с согласованиями и отобразит новое мероприятие в согласованиях со статусом «Согласовано».

Составим диаграмму деятельности для действия бронирования (рисунок 11). Диаграмма деятельности позволяет расширить диаграммы состояний за счет наглядного представления шагов выполнения.

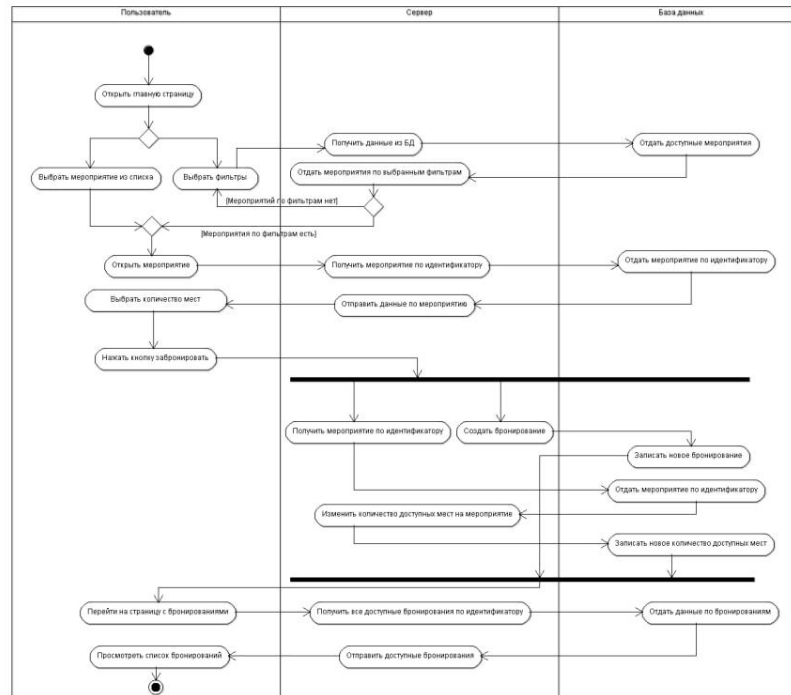


Рисунок 11 – Диаграмма деятельности для бронирования

Рассмотрим подробнее процесс бронирования развлекательного мероприятия. Пользователь заходит на главную страницу Web-приложения, где доступен список развлекательных мероприятий по категориям. При большом объеме данных, есть вариант поиска мероприятий по фильтрам. После выбора фильтров, отправляется запрос на сервер, откуда сервер также делает запрос к базе данных на получение всех мероприятий. На сервере идет перебор отправленных фильтров и при нахождении совпадений, он отправляет результат клиенту, где в браузере отображаются фильтрованные мероприятия.

Далее пользователь выбирает мероприятие и переходит на страницу с более подробными данными о нем. Эти данные так же получается сервером из БД по идентификатору. На странице мероприятия отображаются доступные места на данное мероприятие. Пользователь выбирает количество билетов, которые он хочет оформить и нажимает кнопку «Забронировать». После этого на стороне сервера идет запись нового бронирования, а также обновление данных о доступных местах на данное мероприятие. Новое бронирование отобразится на странице профиля пользователя.

2 Анализ технологий разработки

2.1 Язык моделирования UML

Проектирование клиентской и серверной частей продукта проводится с использованием языка моделирования UML. UML (от английского: Unified Modeling Language) – это язык моделирования, который используется для создания диаграмм, чтобы описать структуру и поведение системы [3]. Диаграммы UML могут использоваться для анализа, проектирования и документирования различных типов систем.

Язык моделирования UML поддерживает объектно-ориентированный подход, позволяя описывать классы и их сущности в визуальном виде, что облегчает последующую разработку серверной части приложения.

Выделяют три категории диаграмм, написанных на языке UML:

- объектные – это диаграммы, описывающие объекты, между которыми выстроена связь;
- поведенческие – это диаграммы, описывающие объекты в действии, их последовательность прохождения;
- функциональные – это диаграммы, описывающие поведение в работе определенных объектов.

Для построения UML диаграмм используются программы ArgoUML, Draw.io – онлайн сервис по построению диаграмм, а также VisualParadigm, для моделирования базы данных.

2.2 Язык программирования серверного приложения

При создании сайта в последние годы существования Интернет-пространства реализуется взаимодействие между сервером и клиентом. Для этого создаются RESTful приложения. Данная технология может реализовываться с использованием многих языков программирование, как с ООП, так и с функциональным программированием. Рассмотрим несколько

наиболее популярных решений, использующихся для реализации данной технологии.

Для разработки серверной части был выбран язык программирования C#.

C# (от английского: C Sharp) – это язык программирования, который был разработан компанией Microsoft. C# был создан как язык, который мог бы использоваться для разработки приложений для платформы .NET.

Можно выделить несколько особенностей данного языка программирования. Он полностью поддерживает объектно-ориентированное программирование, позволяет определять классы, наследовать классы и использовать полиморфизм. C# позволяет использовать как статическую, так и динамическую типизацию. Он поддерживает как компиляцию во время выполнения, так и компиляцию во время разработки. Также он использует автоматическое управление памятью, что означает, что разработчикам не нужно заботиться о выделении или освобождении памяти. Вместо этого сборщик мусора автоматически удаляет объекты, которые больше не используются. Для выполнения сразу нескольких запросов есть поддержка многопоточности [4].

Для разработки web-приложений используется платформа ASP.NET. Особенности данного языка является кроссплатформенность, с поддержкой в разных операционных системах, таких как Windows, Mac OS и Linux. Фреймворк является модульным, благодаря чему отдельные части кода могут загружаться как модули Nuget-пакетов [5]. ASP.NET предоставляет множество функций безопасности, которые помогают защищать приложение от различных видов атак [6].

Архитектура приложения выстроена с использованием микросервисов. Микросервисы – это подход к организации сервисов, использующий отдельно развертываемые части [7]. Такой подход позволяет использовать разные языки, версии фреймворков и в целом различные технологии, в зависимости от настраиваемого сервиса.

Разработка с помощью этой платформы велась в программе Visual Studio, предоставляемой компанией Microsoft.

2.3 Язык программирования клиентского приложения

Для разработки клиентской части приложения использовался язык программирования JavaScript. Данный язык позволяет преподносить информацию пользователю динамически и взаимодействовать с сервером. JavaScript интерпретируется браузерами с помощью специализированных программ. Так для браузеров Chrome, Opera и Edge используется ядро V8, а для Firefox используется SpiderMonkey. Ядра имеют встроенный сборщик мусора, а также введение скрытых классов и ускорение вызова функций, используя память с большой скоростью доступа.

Существуют различные оболочки под JavaScript, позволяющие оптимизировать разработку клиентского приложения.

Для разработки клиентской части была выбрана библиотека React.

React – это библиотека JavaScript описывающая пользовательские интерфейсы. Приложение, написанное с использованием React, состоит из компонентов [8]. Данный подход позволяет разбить SPA на удобные для программиста части, с последующей легкой поддержкой. Особенностью библиотеки является использование виртуального DOM. Данная возможность позволяет обновлять не всю страницу полностью, а лишь часть, где обновились данные. React использует JSX, расширение языка JavaScript, которое позволяет использовать HTML-подобный синтаксис для создания пользовательского интерфейса. Это делает код более читаемым и понятным.

Клиентская часть писалась в приложении Visual Studio Code, приложение запускается через консоль.

2.4 Базы данных

Существует множество систем управления базами данных (СУБД), которые можно поделить на несколько категорий:

- реляционные – используются для работы, где требуется отсылать ограниченное число транзакций, но с большим идущим потоком;
- ключ-значение – база данных, где используется структура с уникальным ключом и принадлежащим ему значением;
- документные – база данных, где моделью является документ, состоящий из определенного текста с уникальным синтаксисом;
- графовые – тип базы данных, моделью которого является граф с узлами и свойствами;
- колоночные – тип базы данных, схожий с реляционной моделью, где отличительной чертой является хранение данных не построчно, а в столбцах.

MongoDB – это документоориентированная система управления базами данных, которая не требует описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных.

MongoDB позволяет хранить данные в формате документов, что делает ее гибкой и удобной для работы с неструктурированными данными [9]. Кроме того, MongoDB позволяет хранить данные различных типов, включая числа, строки, массивы и даже другие документы. Также обеспечивает высокую производительность благодаря своей архитектуре и возможности распределения нагрузки на несколько серверов.

MongoDB доступна для использования бесплатно в коммерческих и некоммерческих проектах, что делает ее доступной для разработчиков с различным бюджетом.

MongoDB Atlas – это облачный сервис для хранения данных. База данных делится на кластеры с хранимыми таблицами.

3 Проектирование приложения

3.1 Проектирование базы данных

При проектировании базы данных использовалась программа Visual Paradigm, предоставляющая моделирование Entity Relationship, подходящих для описания связей между таблицами.

Опишем логическую модель для базы данных. Она будет состоять из шести баз данных:

- Events – описывает таблицы для мероприятий;
- Categories – описывает таблицы для категорий, используемых в мероприятиях;
- Users – описывает таблицы пользователя;
- Favorites – описывает таблицы избранных мероприятий для каждого пользователя;
- Booking – описывает таблицы покупок пользователем определенных мероприятий;
- Approvals – описывает таблицы для согласования мероприятий.

Логические модели баз данных представлены в приложении Е.

3.2 Проектирование серверной части

Серверная часть использует язык программирования C# с использованием фреймворка для REST API – ASP.NET. Микросервисы могут быть объединены общим слоем – API шлюзом, через который будет проходить каждый запрос.

Проходя через данный шлюз каждый REST запрос будет менять хост и порт, позволяя хранить доступные маршруты всех микросервисов. Также это позволяет проверять на корректность данные благодаря описанию маршрутов.

Данный процесс можно описать с использованием диаграммы контейнеров (рисунок 12).

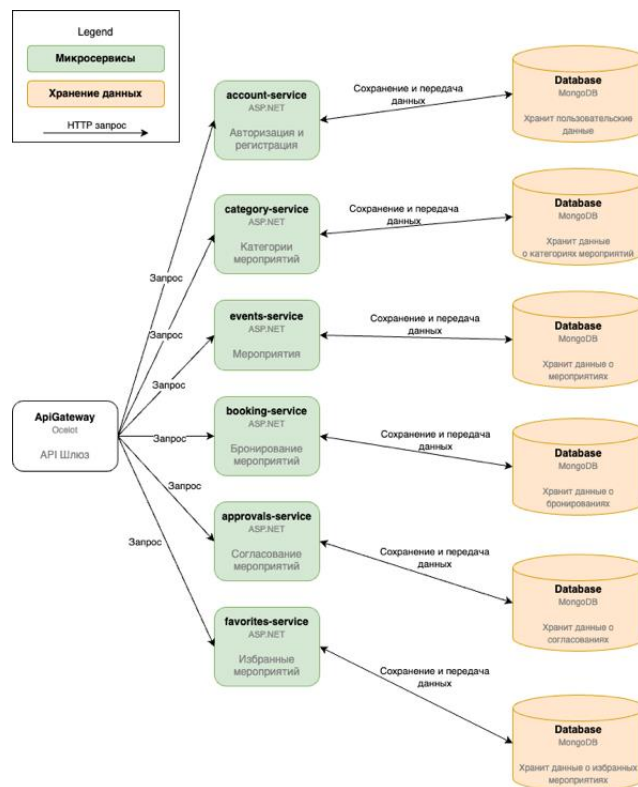


Рисунок 12 – Диаграмма контейнеров для микросервисов

В данном случае, API шлюзом будет являться настраиваемый Nuget пакет Ocelot.

Перечислим микросервисы, предоставляющие API шлюзу:

- account-service – сервис, отвечающий за аутентификацию и авторизацию;
- category-service – сервис, отвечающий за категории для мероприятий;
- events-service – сервис, отвечающий за работу с мероприятиями;
- booking-service – сервис, отвечающий за бронирование;
- approvals-service – сервис, отвечающий за согласования мероприятий;
- favorites-service – сервис, отвечающий за избранные мероприятия.

Каждый микросервис реализован со своими определенными методами, моделями и сервисами. Опишем микросервисы, используя диаграммы классов в приложении Д.

Опишем методы микросервиса для пользователей с принимаемыми значениями и возвращаемыми типами в таблице 8.

Таблица 8 – Описание методов микросервиса account-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
account-service	LoginController	Get	Получение пользователей	HttpGet
		Login	Нахождение пользователя и генерация JWT-токена	HttpPost
		Generate	Генерация JWT-токена	-
		Post	Регистрация	HttpPost
	UserController	AdminsEndpoint	Получение текущего пользователь, если его роль «Администратор»	HttpGet
		AdminsAndSellersEndpoint	Получение текущего пользователь, если его роль «Администратор» или «Пользователь»	HttpGet
		GetCurrentUser	Получение текущего пользователя	-
	UsersService	GetAsync	Получение из базы данных пользователей	-
		CreateAsync	Отправка в базу данных нового пользователя	-
		GetAsync	Получение пользователя по идентификатору из базы данных	-

Для отправки данных в базу данных, была описана модель данных в классе UserModel с необходимыми полями, которые ранее описывались в логической модели для базы данных «Users».

Опишем методы микросервиса для мероприятий в таблице 9.

Таблица 9 – Описание методов микросервиса events-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
events-service	EventsController	Get	Получение всех мероприятий	HttpGet
		Get	Получение по id мероприятия	HttpGet
		GetByUser	Получение по userId мероприятия	HttpGet
		Delete	Удаление мероприятия	HttpDelete
		Post	Добавление мероприятия	HttpPost
		Update	Изменение мероприятия	HttpPut
		UploadedFile	Загрузка файла на сервер	-
		GetImage	Получение изображения с сервера	HttpGet
		GetEventByFilters	Получение фильтрованных мероприятий	HttpGet
	EventsService	GetAsync	Получение из базы данных всех мероприятий	-
		GetAsync	Получение из базы данных мероприятия по идентификатору	-
		GetByUserAsync	Получение из базы данных мероприятия по его создателю	-
		CreateAsync	Создание в базе данных мероприятия	-
		UpdateAsync	Обновление в базе данных мероприятия	-
		RemoveAsync	Удаление из базы данных мероприятия	-

Для отправки данных в базу данных, была описана модель данных в классе EventsViewModel, с необходимыми полями, которые ранее описывались в логической модели для базы данных «Events». Модель Events описана для отправленных данных от клиента с типом для поля ProfilePicture – IFormFile. Для

сохранения данных в базе данных была описана модель EventsViewModel, где будет сохранено название изображения для дальнейшего получения из базы данных. ISeats и ILocation являются описанием, какие поля будут содержать поле Seats и Location.

Опишем методы микросервиса для категорий мероприятий в таблице 10.

Таблица 10 – Описание методов микросервиса categories-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
categories-service	CategoriesController	Get	Получение всех категорий	HttpGet
		Post	Добавление категории	HttpPost
	CategoriesService	GetAsync	Получение из базы данных всех категорий	-
		CreateAsync	Создание в базе данных категории	-

Для отправки данных в базу данных, была описана модель данных в классе Categories с необходимыми полями, которые ранее описывались в логической модели для базы данных «Categories».

Опишем методы микросервиса для согласований в таблице 11.

Таблица 11 – Описание методов микросервиса approvals-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
approvals-service	ApprovalsController	Get	Получение всех согласований	HttpGet
		Get	Получение по id согласования	HttpGet
		GetByUser	Получение по userId согласования	HttpGet
		Delete	Удаление согласования	HttpDelete
		Post	Добавление согласования	HttpPost
		Update	Изменение согласования	HttpPut
		UploadedFile	Загрузка файла на сервер	-

Продолжение таблицы 11

Сервис	Классы	Метод	Описание метода	HTTP-метод
	ApprovalsService	GetImage	Получение изображения с сервера	HttpGet
		GetAsync	Получение из базы данных всех согласований	-
		GetAsync	Получение из базы данных согласования по идентификатору	-
		GetByUserAsync	Получение из базы данных согласования по его создателю	-
		CreateAsync	Создание в базе данных согласования	-
		UpdateAsync	Обновление в базе данных согласования	-
		RemoveAsync	Удаление из базы данных согласования	-

Для отправки данных в базу данных, была описана модель данных в классе ApprovalsViewModel, с необходимыми полями, которые ранее описывались в логической модели для базы данных «Approvals». Модель Approvals описана для отправленных данных от клиента. Для сохранения данных в базе данных была описана модель ApprovalsViewModel, где будет сохранено название изображения для дальнейшего получения из базы данных. IComment является описанием, какие поля будет содержать поле Comments.

Опишем методы микросервиса для бронирований в таблице 12.

Таблица 12 – Описание методов микросервиса booking-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
booking-service	BookingController	Get	Получений бронирований	HttpGet
		Get	Получение по userId бронирований	HttpGet

Продолжение таблицы 12

Сервис	Классы	Метод	Описание метода	HTTP-метод
		Post	Создание бронирования	HttpPost
		GetByUser	Получение бронирования по создателю мероприятия	HttpGet
		Delete	Отмена бронирования	HttpDelete
		GetByBuyer	Получение бронирования по покупателю	HttpGet
	BookingService	GetAsync	Получение из базы данных всех бронирования	-
		GetAsync	Получение из базы данных бронирования по идентификатору	-
		GetByUserAsync	Получение из базы данных бронирования по его покупателю	-
		GetByBuyerAsync	Получение из базы данных бронирования по его создателю	-
		CreateAsync	Создание в базе данных бронирования	-
		UpdateAsync	Обновление в базе данных бронирования	-
		RemoveAsync	Удаление из базы данных бронирования	-

Для отправки данных в базу данных была описана модель данных в классе Booking с необходимыми полями, которые ранее описывались в логической модели для базы данных «Booking».

ISeats и ILocation являются описанием, какие поля будут содержать поле BookingSeats и Location.

Опишем методы микросервиса для избранных мероприятий в таблице 13.

Таблица 13 – Описание методов микросервиса favorites-service

Сервис	Классы	Метод	Описание метода	HTTP-метод
favorites-service	FavoritesController	Get	Получений избранных мероприятий	HttpGet
		Post	Создание избранного мероприятия	HttpPost
		GetByUser	Получение избранного мероприятия по пользователю	HttpGet
		Delete	Удаление избранного мероприятия	HttpDelete
	FavoritesService	GetAsync	Получение из базы данных всех избранных мероприятий	-
		GetByUserAsync	Получение из базы данных избранного мероприятия по пользователю	-
		CreateAsync	Создание в базе данных избранного мероприятия	-
		RemoveAsync	Удаление из базы данных избранного мероприятия	-
		CreateAsync	Создание в базе данных избранного мероприятия	-
		RemoveAsync	Удаление из базы данных избранного мероприятия	-

Для отправки данных в базу данных была описана модель данных в классе FavoritesViewModel, с необходимыми полями, которые ранее описывались в логической модели для базы данных «Favorites». Модель Favorites описана для отправленных данных от клиента. Для сохранения данных в базе данных была описана модель FavoritesViewModel, где будет сохранено название изображения для дальнейшего получения из базы данных.

Данные микросервисы по диаграмме контейнеров объединяются через шлюз ApiGateway с настройками Ocelot.

3.3 Проектирование клиентской части

Для разработки клиентской части используется библиотека React, позволяющая создавать SPA приложения. Какой компонент в данный момент отображать определяется с помощью навигации. Для проектирования клиентской части используем диаграмму компонентов (рисунок 13).

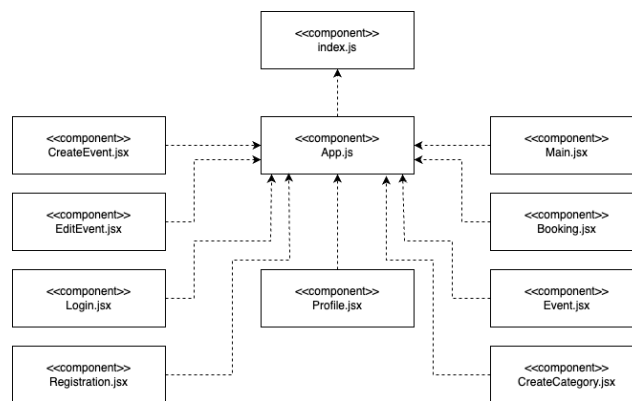


Рисунок 13 – Диаграмма компонентов

Первое, что читает браузер – это файл index.html. В него внедряются последующие компоненты, чтобы браузер их мог прочитать.

В App.js происходит распределение по навигации каждой условной страницы. Навигация настраивается с использованием пакета react-router-dom.

Main.jsx – это главная страница, получаемая по ссылке «/», где будет отображаться список мероприятий, будет доступна фильтрация по этому списку и автозаполнению по мероприятиям.

Event.jsx – страница, получаемая по ссылке «/event/{id}», на которую идет перенаправление при выборе из списка определенного мероприятия. На данной странице можно будет подробнее посмотреть информацию и выбрать билет.

Login.jsx – страница, доступная по ссылке «/login», перенаправление на которую идет при выборе кнопки «Авторизация». На данной странице отображаются блоки ввода: логин и пароль, а также кнопка «Войти».

Registration.jsx – страница, доступная по ссылке «/registration», перенаправление на которую идет с главной странице по кнопке «Регистрация», отображаемой в заголовке.

CreateEvent.jsx – страница, доступная по ссылке «/createEvent», перенаправление на которую идет по кнопке «Создать мероприятие». Данная кнопка и страница доступны лишь для пользователя с ролью «Администратор». На странице отображаются поля, нужные для создания мероприятия и кнопка «Создать».

EditEvent.jsx – страница, доступная по ссылке «/editEvent», перенаправление на которую идет по кнопке, отображаемой на каждом мероприятии, отображаемом в списке. На странице отображаются поля, нужные для редактирования мероприятия и кнопка «Редактировать».

CreateCategory.jsx – страница, доступная по ссылке «/createCategory», перенаправление на которую идет по кнопке «Создать категорию». На странице отображаются поля, нужные для создания категории и кнопка «Создать».

Booking.jsx – страница, доступная по ссылке «/booking/{id}», перенаправление на которую идет при выборе места у мероприятия и нажатия на кнопку «Забронировать».

На главной странице отображаются и фильтруются все мероприятия (рисунок 14).

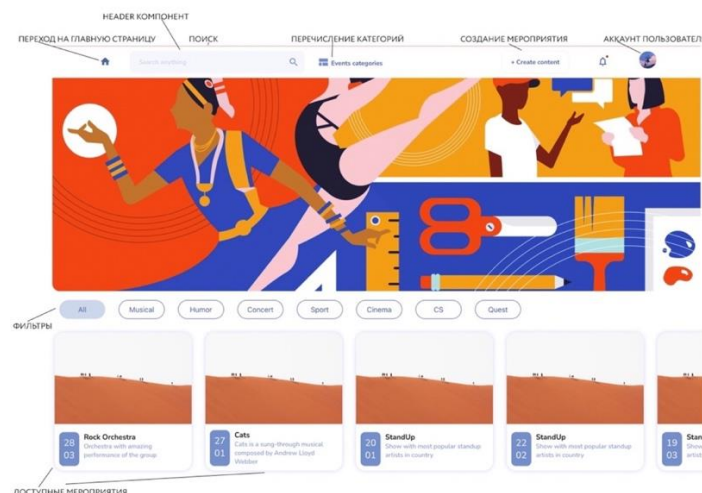


Рисунок 14 – Главная страница

В заголовке вынесена навигация для доступа к любой странице из любого места приложения. Изображение дома отвечает за перемещение на основную страницу, доступную по ссылке «/». Поиск осуществляет фильтрацию по мероприятиям по их наименованию.

Страница создания мероприятия представлена на рисунке 15.

The image shows a web application header and a 'Create event' form. The header contains a home icon labeled 'ПЕРЕХОД НА ГЛАВНУЮ СТРАНИЦУ', a search bar labeled 'ПОИСК' with the text 'Search anything', a link to 'Events categories' labeled 'ПЕРЕЧИСЛЕНИЕ КАТЕГОРИЙ', a '+ Create content' button labeled 'СОЗДАНИЕ МЕРОПРИЯТИЯ', and a user profile icon labeled 'АККАУНТ ПОЛЬЗОВАТЕЛЯ'. The 'Create event' form has the following fields: 'Event Title' (text input), 'Event Description' (text input), 'Price' (text input with '0'), 'Type' (dropdown menu with 'Musical' selected), 'Location' (text input), and 'Date' (text input). A blue 'Add event' button is at the bottom. Annotations in Russian point to these elements: 'HEADER КОМПОНЕНТ' points to the header area, 'ПОЛЯ ДЛЯ ЗАПОЛНЕНИЯ' points to the form fields, and 'КНОПКА ОТПРАВЛЕНИЯ ЗАПОЛНЕННОЙ ФОРМЫ' points to the 'Add event' button.

Рисунок 15 – Страница создания мероприятия

На ней располагаются все поля, которые заданы в модели на сервере, и нужные для отправки.

4 Реализация

4.1 Реализация базы данных

При реализации базы данных использовался MongoDB Atlas. Были созданы базы данных, таблицы которых хранятся в JSON-формате. Опишем базы данных в приложении Г, таблице Г.1.

Пример таблицы «Events», с записанными в базу банными объектами, представлен на рисунке 16.

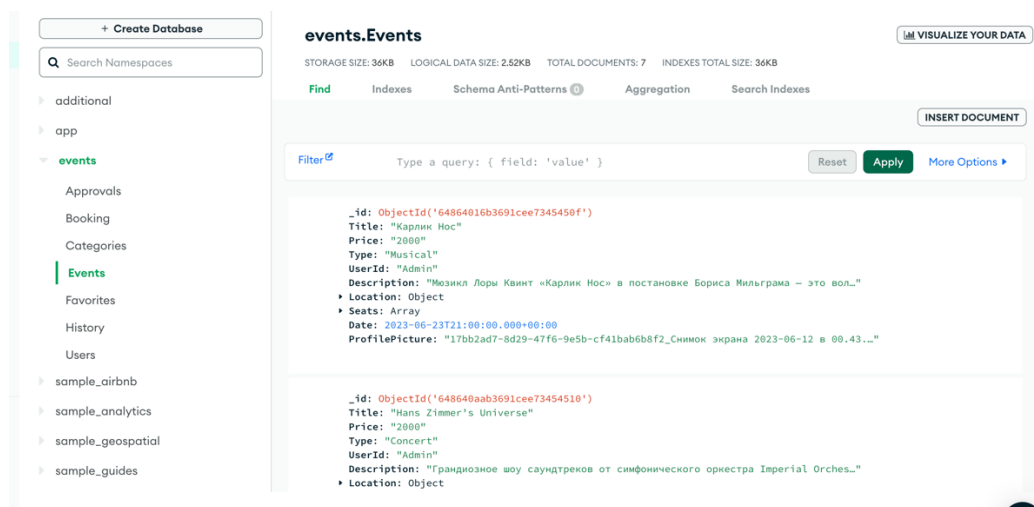


Рисунок 16 – Таблица в MongoDB

У каждого созданного объекта имеется уникальный идентификатор, хранящийся в поле «_id», а также поля с определенным типом данных.

4.2 Реализация серверной части

Разработка микросервисов велась на языке программирования C# используя фреймворк ASP.NET и IDE – Visual Studio. Для описания микросервисной архитектуры и связи между микросервисами, с описанными сервисами, контроллерами и моделями, реализованными в web-приложении, подойдет описание с использованием UML диаграммы классов [10].

Список исходников приложения для мероприятий представлена на рисунке 17.

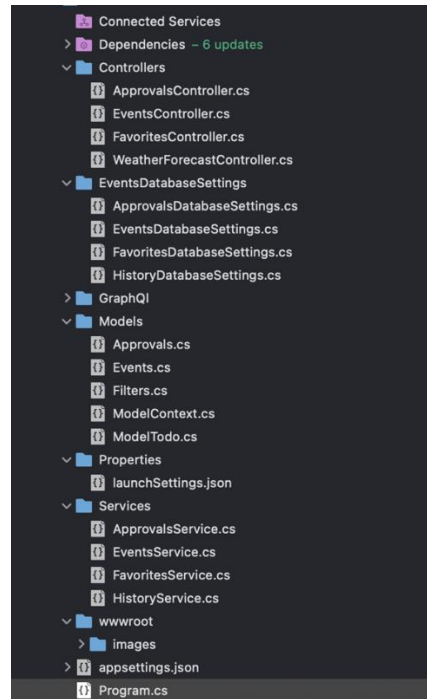


Рисунок 17 – Архитектура микросервиса для мероприятий

Исходники делятся на следующие элементы:

- модели данных – папка **Models** с описанием используемых данных и их типов;
- зависимости – папка **Dependencies** с подключенными NuGet-пакетами;
- контроллеры – папка **Controllers** с описанием запросов, по которым идет обращение с клиентской части приложения;
- сервисы – папка **Services** с описанием асинхронных обращений к базе данных;
- настройки взаимодействия с базой данных – папка **EventsDatabaseSettings**;
- хранение локальных данных – папка **wwwroot** с сохраненными изображениями мероприятий;
- настройка запуска приложения – папка **Properties** с Json-объектом.

Для описания API используется набор инструментов – Swagger. Для его подключения в Program.cs пишется `builder.Services.AddSwaggerGen()` и `app.UseSwaggerUI()` для использования.

Чтобы соединить приложение с MongoDB Atlas устанавливался NuGet пакет `MongoDB.Driver`.

В `appsettings.json` задаются параметры обращения к определенной БД, с указанием пользователя самой базы данных и таблицы в ней.

Перед сборкой приложения, подключаем обращение к базе данных в Program.cs (листинг 1).

Листинг 1 – Подключение к базе данных

```
builder.Services.Configure<EventsDatabaseSettings>(builder.Configuration.GetSection("EventsDatabase"));
```

Создавая запросы, важно учитывать данные, которые понадобятся при использовании приложения. На основе проектирования, было решено создавать 6 запросов.

Для получения всех данных по мероприятиям получаем всю информацию из базы данных.

Листинг 2 – Получение данных из таблицы Events

```
public EventsService(
    IOptions<EventsDatabaseSettings> eventsDatabaseSettings)
{
    var mongoClient = new MongoClient(
        eventsDatabaseSettings.Value.ConnectionString);

    var mongoDatabase = mongoClient.GetDatabase(
        eventsDatabaseSettings.Value.DatabaseName);

    _eventsCollection = mongoDatabase.GetCollection<Events>(
        eventsDatabaseSettings.Value.EventsCollectionName);
}
```

Обращаемся к полученной коллекции «`_eventsCollection`» и с помощью встроенного в язык метода «Find», находим все значения. Затем переводим их в

тип «List» из асинхронного перечисления с помощью метода «ToListAsync» (листинг 3).

Листинг 3 – Асинхронный запрос к базе данных

```
public async Task<List<Events>> GetAsync() =>
    await _eventsCollection.Find(_ => true).ToListAsync();
```

В «EventsController» для метода «Get» получаем все данные о мероприятиях, обращаясь к созданному сервису.

В контроллере формируем запрос «Get» метода на получение данных о выбранном мероприятии из таблицы «Events», где при отсутствии данного значения отдается ошибка 404 – не найдено, а при нахождении данного объекта, идет сохранение в базу данных и отдается сам объект со статусом 200 (листинг 4).

Листинг 4 – Формирование Get запроса на получение мероприятия по id

```
[HttpGet («{id:length(24)}»)]
public async Task<ActionResult<Events>> Get(string id)
{
    var events = await _eventsService.GetAsync(id);
    if (events is null)
    {
        return NotFound();
    }

    return events;
}
```

Остальные запросы формируются аналогично выше представленному примеру описания микросервиса «Events» и представлены в приложении В.

Каждый из созданных микросервисов проходит через шлюз ApiGateway. В этом шлюзе используется набор программного обеспечения – Ocelot. Он подключается как NuGet пакет и используется как конфигурация ocelot.json в Program.cs. Это Json-схема, которая из объектов, принимающих несколько значений:

- UpstreamPathTemplate – принимающий запрос;

- UpstreamHttpMethod – набор методов, доступных по данному запросу;
- DownstreamHostAndPorts – набор, состоящий из хоста и порта запроса, описанного в микросервисе;
- DownstreamPathTemplate – отдающий запрос.

Приведем пример описанной схемы формирования настройки промежуточного слоя на основе микросервиса с согласованиями (листинг 5).

Листинг 5 – Настройка шлюза для микросервиса с согласованиями

```
// Approvals
{
  "UpstreamPathTemplate": "/api/Approvals",
  "UpstreamHttpMethod": [ "Get", "Post" ],
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 7259
    }
  ],
  "DownstreamPathTemplate": "/api/Approvals"
},
{
  "UpstreamPathTemplate": "/api/Approvals/{id}",
  "UpstreamHttpMethod": [ "Get", "Put", "Delete" ],
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 7259
    }
  ],
  "DownstreamPathTemplate": "/api/Approvals/{id}"
},
{
  "UpstreamPathTemplate": "/api/Approvals/{user}/user",
  "UpstreamHttpMethod": [ "Get" ],
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 7259
    }
  ],
}
```

```

    "DownstreamPathTemplate": "/api/Approvals/{user}/user"
  }

```

В представленной настройке выделяется три объекта. В первом представлен запрос «/api/Approvals», с доступными методами Get и Post, первоначальным обращением к которому идет через «https://localhost:7259» и далее после прохождения через шлюз становится доступным под общим хостом «http://localhost:5152» и обращением также по «/api/Approvals».

Шлюз в данном случае позволяет делать запросы по одному хосту и порту.

Описание остального кода клиентского приложения представлен в приложении А.

4.3 Реализация клиентской части

Разработка клиентской части велась с использованием библиотеки React в IDE – Visual Studio Code. Опишем, как логически выстраивалась разработка приложения. Изначально была выбрана архитектура приложения. Список исходников клиентского приложения представлен на рисунке 18.

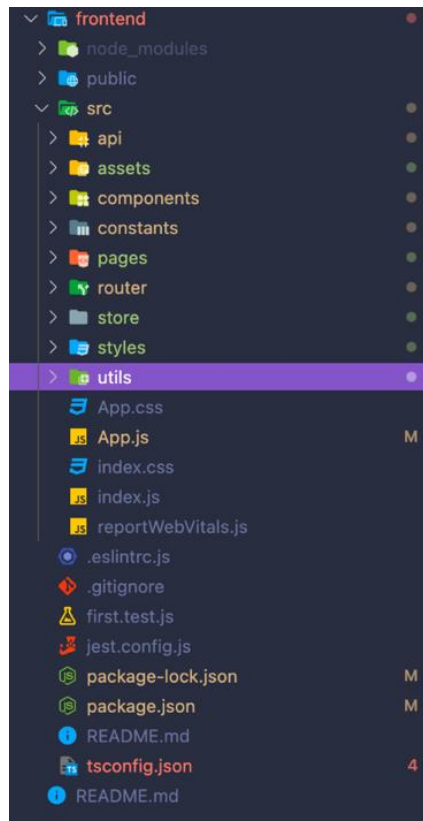


Рисунок 18 – Архитектура клиентского приложения

Описание исходников frontend-части:

- `node_modules` – подключенные react библиотеки [11];
- `public` – общая папка, содержащая HTML страница, в которую встраивается далее описанные страницы;
- `src` – папка, в которой хранится описание SPA приложения;
- `api` – папка, хранящая запросы к серверу, а также настройку библиотеки `axios`, основанную на `Promise`-объекте;
- `assets` – общие файлы, используемые в проекте, такие как картинки;
- `components` – папка, содержащая общие визуальные части приложения, которые встраиваются в страницы, а также собственно стилизованные UI-элементы, такие как `Input`, `Select`, `Button`, `Card`;
- `constants` – папка с заранее зарезервированными значения, для использования их по всему проекту;
- `pages` – папка, хранящая все страницы, используемые в приложении;
- `router` – папка, в которой хранится настройка маршрутизации SPA приложения;
- `store` – папка, хранящая логическую часть приложения с использованием хранилища состояния приложения – `mobx`, с файлами, описывающими поведение после совершения запроса к серверу;
- `styles` – папка, хранящая общие стили и переменные в приложении;
- `utils` – папка, хранящая общие для всего приложения функции;
- `.eslintrc.js` – настройка для проекта, позволяющая следить за стилистикой написания кода;
- `.gitignore` – настройка игнорируемых файлов и папок для распределенной системы управления версии `git`;
- `first.test.js` – файл с описанием автотестирования с использованием библиотеки `jest`;

- jest.config.js – настройка для тестирования приложения библиотекой jest;
- package-lock.json – файл, содержащий зависимости версий, подключаемых библиотек между собой;
- package.json – файл с описанием подключаемых библиотек, а также настройку запуска приложения.

Опишем логику получения и отображения данных на примере страницы профиля.

Для получения данных для страницы профиля, идет запрос в микросервисы «booking-servise» и «event-service». Они получаются по email пользователя, который в данный момент находится на странице (листинг 6).

Листинг 6 – Получение бронирований и продаж из booking-service

```
export const _getBook = (userId) =>
  axios.get(`/Booking/${userId}/buyer`, {});
export const _getSeller = (userId) =>
  axios.get(`/Booking/${userId}/user`, {});
```

В настройках библиотеки axios уже описано, что каждый запрос будет идти по стандартному хосту и порту «http://localhost:5152». Для получения данных о бронирования и продажах используются метод Get.

Далее полученные данные сохраняются в папке store с использованием mvx библиотеки (листинг 7).

Листинг 7 – Сохранение данных в хранилище состояний приложения

```
getBook = async (userId) => {
  try {
    const { data } = await _getBook(userId);

    this.booking = data;
  } catch (e) {
    console.log(e);
  }
};

getSeller = async (userId) => {
  try {
```

```

        const { data } = await _getSeller(userId);

        this.seller = data;
    } catch(e) {
        console.log(e);
    }
};

```

Здесь, в массив «booking» и массив «seller» сохраняются все полученные данные при успешном запросе, а при неудачном запросе (коды кроме 2xx) функция попадает в catch и в консоли отображается причина ошибки. Далее полученные данные используются в компоненте «Profile» для отображения. Отображение данных представлено в листинге 8.

Листинг 8 – Получение в компоненте данных о бронированиях и продажах

```

useEffect(() => {
    booking.getBook(auth.user.emailAddress);
    booking.getSeller(auth.user.emailAddress);
}, []);

```

Полученные данные для отображения бронирований передаются в компонент «CardProfile», где идет их представление в DOM (листинг 9).

Листинг 9 – Отображение бронирований

```

<TabPanel value={ value } index={ 0 }>
    {
        booking.booking ? booking.booking.map((item) => {
            const { id } = item;

            return (
                <CardProfile
                    key={ id }
                    event={ item }
                    withEdit={ false }
                    onClickDelete={ handleClickDelete }
                />
            );
        })
        : <Text text='Сейчас здесь нет бронирований :('
            style='text-a2' />
    }
</TabPanel>

```

При не пустом массиве бронирований перебираем массив данных с помощью встроенного метода JavaScript – map и отображаем каждый элемент.

Также в «CardProfile» передается событие на удаление бронирования. Если массив пустой, то отображается стандартный текст, указывающий пользователю, что забронированных мероприятий пока что нет.

Результат страницы профиля с доступными бронированиями представлен на рисунке 19.

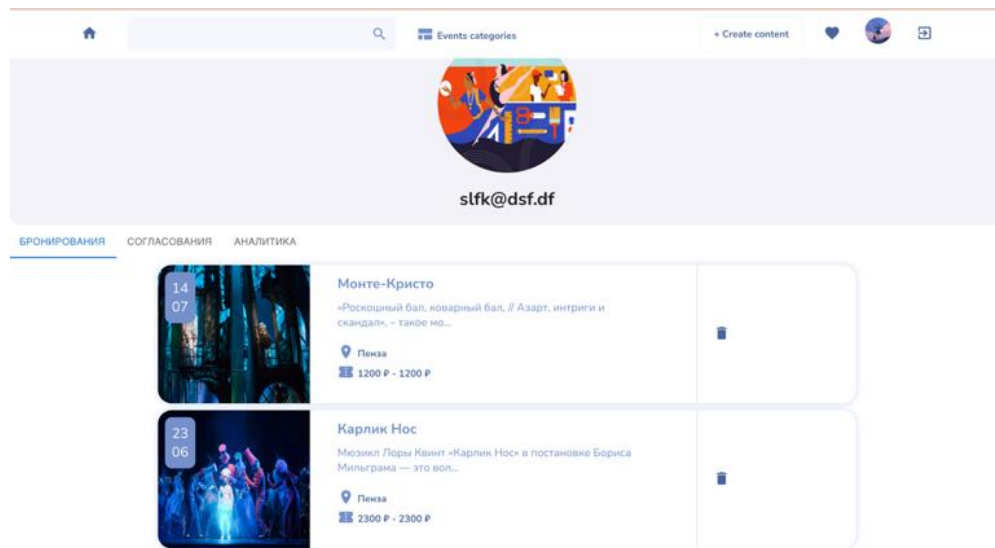


Рисунок 19 – Отображение бронирований

Данные с продажами мероприятий используются в аналитике данных. Для графиков используется библиотека «react-chartjs-2». Отображаем компонент во вкладке «Аналитика» (листинг 10).

Листинг 10 – Отображение компонента аналитики

```
<TabPanel value={ value } index={ 2 }>
  <Analytics booking={ booking.booking } sell={ booking.seller }
/>
</TabPanel>
```

Для удобного представления данных о продажах был выбран график гистограмма, который импортируется из библиотеки «react-chartjs-2» как компонент «Bar». Он принимает в себе данные с названиями мероприятий по X-оси и сумме продаж по каждому мероприятию по Y-оси.

Для представления данных пользователю по его покупкам была выбрана круговая диаграмма и из библиотеки использовался компонент «Pie», принимающий данные по категориям и суммам покупок по каждой из категорий.

Результат аналитики на странице «Analytics» представлен на рисунке 20.

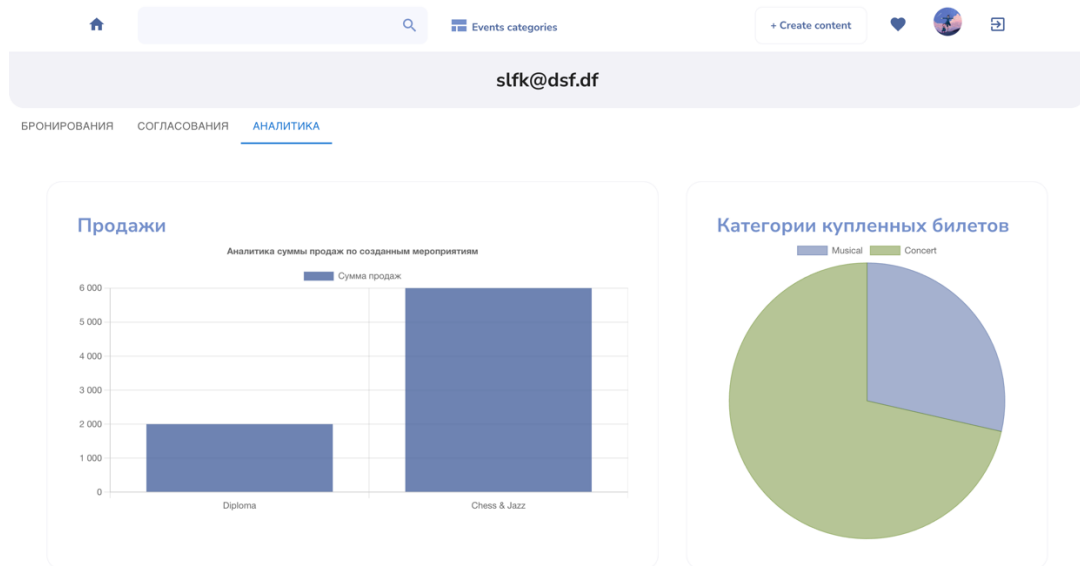


Рисунок 20 – Аналитика данных по покупкам и продажам

По данной аналитике можно предложить пользователю добавить в интересы концерты, так как по данной категории совершалось наибольшее количество покупок. Это позволит делать подборку на главной странице, наиболее подходящей под него.

Описание остального кода клиентского приложения представлен в приложении Б.

4.4 Реализация развертывания приложения

Развертывание делится на развертывание микросервисов и развертывание клиентского приложения. Все микросервисы проходят через ApiGateway шлюз, формируя один хост, к которому обращается клиентская часть. Микросервисы в то же время обращаются к базам данных MongoDB, отдающие и хранящие данные. Клиентская часть представляет собой React приложение, состоящее из компонентов, проходящее через один файл – index.js, что делает его

одностраничным приложением. Приложение обращается к серверу и получает данные с сервера. Опишем развертывание в виде диаграммы развертывания (рисунок 21).

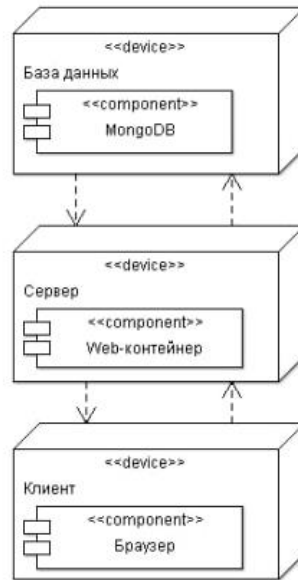


Рисунок 21 – Диаграмма развертывания

Узлами в данной диаграмме являются:

- «База данных»;
- «Сервер»;
- «Клиент».

Артефактами являются:

- MongoDB;
- Web-контейнер;
- Браузер.

Каждый узел может взаимодействовать друг с другом. Так клиентская часть обращается к серверной части с помощью REST запросов. Серверная часть обращается к базам данных с помощью Nuget-пакета MongoDB.Driver.

5 Тестирование информационной системы по подбору и продаже билетов на развлекательные мероприятия

Для тестирования информационной системы применялись методы тестирования черным ящиком, метрики программного обеспечения и автоматизированное тестирование.

5.1 Тестирование создания и редактирование развлекательного мероприятия

Тестирование по стратегии чёрного ящика, часто называемое функциональным тестированием – это методика, изучающая функциональность ПО без необходимости знания внутренней структуры кода.

При тестировании по стратегии чёрного ящика исследуются все отдельные компоненты [12].

Тестирование проведено для поля даты в создании и редактировании мероприятия, используя классы эквивалентности.

5.1.1 Классы эквивалентности

Для создания и редактирования развлекательного мероприятия выделено поле даты проведения. Формат даты hhhh-mm-dd, поле является обязательным для заполнения (таблица 14).

Таблица 14 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина поля	10(1)	>10(2) <10(3) Null(4)
Структура	Содержит цифры, год(4 цифры) месяц(2 цифры) и день(2 цифры) с разделителями –(5)	Содержит буквы и символы кроме –(6)
Содержание	Цифры: год – цифры, месяц – цифры, день – цифры (7) Дефис(8)	Остальные(9)

Графическое представление для классов эквивалентности представлено на рисунке 22.



Рисунок 22 – Графическое представление

Структура для классов эквивалентности представлена на рисунке 23.

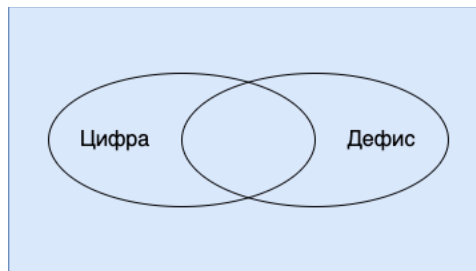


Рисунок 23 – Структура для классов эквивалентности

Для классов эквивалентности были составлены тест-кейсы (таблица 15).

Таблица 15 – Тест кейсы

№	КЭ	Входные данные	Выходные данные
0	Все правильные (1,5,7,8)	2022-02-13	Запись в БД
1	Заменяем только один критерий на неправильный: 2,5,7,8	2022-02-123	Ошибка: Больше 10 символов
2	4	Null	Ошибка: Поле обязательное для заполнения
3	3,6,7,8	22-02-13	Ошибка: Меньше 10 символов
4	2,5,9	2022-12-34A	Ошибка: Некорректные символы. Некорректная длина
5	1,5,9	2022-12-AA	Ошибка: Некорректные символы
6	3,5,9	22-02-AA	Ошибка: Меньше 10 символов. Некорректные символы

Были предусмотрены тесты непересекающихся и пересекающихся множеств, на основе которых выявлялась корректность работы поля дата.

5.1.2 Функциональная диаграмма и таблица решений для информационной системы

Для тестирования создания мероприятия была создана таблица причин и следствий и представлена в таблице 16.

Таблица 16 – Таблица причин и следствий

Причины	
П1	Корректная длина названия мероприятия
П2	Корректная цена
П3	Корректная длина даты
П4	Цена состоит из корректных символов
Следствия	
С1	Сообщение «Введите название мероприятия»
С2	Сообщение «Введите цену»
С3	Сообщение «Введите дату»
С4	Сообщение «Введите корректную цену»

Для успешного создания мероприятия нужно заполнить поля названия, цены и даты. Визуальное представление причин и следствий изображено на функциональной диаграмме на рисунке 24.

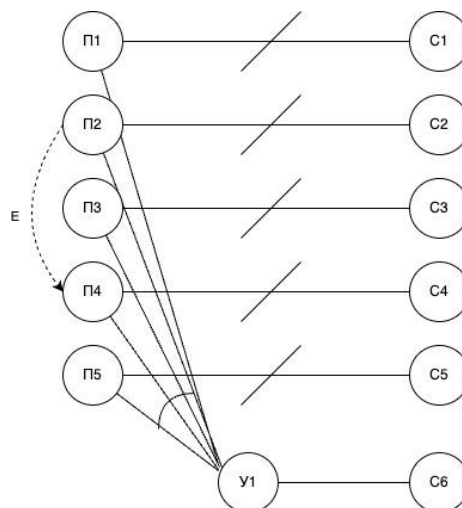


Рисунок 24 – Функциональная диаграмма причин и следствий

Все поля данной формы по созданию мероприятия являются обязательными для заполнения. Причина 2 связана с причиной 4, так как пустое значение цены и некорректные символы не могут пересекаться.

На основе таблицы и диаграммы была построена таблица решений (таблица 17), позволяющая составить тесты по всем возможным комбинациям входных данных.

Таблица 17 – Таблица решений

П1	0	1	0	1	0	1	0	1	0	1	0	1	1
П2	1	1	1	1	0	0	1	1	0	0	1	1	1
П3	0	0	1	1	0	0	0	0	1	1	1	1	1
П4	0	0	0	0	1	1	1	1	1	1	1	1	1
П5	0	0	0	0	0	0	0	0	0	0	0	0	1
С1	1	0	1	0	1	0	1	0	1	0	1	0	0
С2	0	0	0	0	1	1	0	0	1	1	0	0	0
С3	1	1	0	0	1	1	1	1	0	0	0	0	0
С4	1	1	1	1	0	0	0	0	0	0	0	0	0
С5	0	0	0	0	0	0	0	0	0	0	0	1	0
С6	0	0	0	0	0	0	0	0	0	0	0	0	1

По полученным данным были написаны тест-кейсы, позволяющие проверить каждую комбинацию причин. Тест-кейсы представлены в таблице 18.

Таблица 18 – Тест-кейсы таблицы решений

№	Входные данные	Выходные данные
1	Название: А Цена: null Дата: 22-02-А	Некорректная длина названия Некорректная длина цены Некорректная длина даты Некорректные символы в дате
2	Название: А Цена: aa Дата: 22-02-А	Некорректная длина названия Некорректная длина даты Некорректные символы в цене Некорректные символы в дате
3	Название: Фильм Цена: null Дата: 22-02-А	Некорректная длина даты Некорректная длина цены Некорректные символы в дате

Продолжение таблицы 18

№	Входные данные	Выходные данные
4	Название: А Цена: 200 Дата: 22-02-А	Некорректная длина названия Некорректная длина даты Некорректные символы в дате
5	Название: А Цена: null Дата: 2022-02-12	Некорректная длина названия Некорректная длина цены
6	Название: Фильм Цена: 200 Дата: 22-02-А	Некорректная длина даты Некорректные символы в дате
7	Название: Фильм Цена: 200 Дата: 22-02-02	Некорректная длина даты
8	Название: Фильм Цена: null Дата: 22-02-12	Некорректная длина цены Некорректная длина даты
9	Название: Фильм Цена: aa Дата: 22-02-12	Некорректные символы в цене Некорректная длина даты
10	Название: Фильм Цена: aa Дата: 22-02-aa	Некорректные символы в цене Некорректная длина даты Некорректные символы в дате
11	Название: а Цена: 200 Дата: 2022-02-12	Некорректная длина названия
12	Название: Фильм Цена: aa Дата: 2022-02-12	Некорректные символы в цене
13	Название: Фильм Цена: null Дата: 2022-02-12	Некорректная длина цены
14	Название: Фильм Цена: 200 Дата: 22-02-0A	Некорректные символы в дате
15	Название: Фильм Цена: 200 Дата: 2022-02-0f	Некорректные символы в дате
16	Название: А Цена: ff Дата: 2022-02-12	Некорректная длина названия Некорректные символы в цене

Форма для создания развлекательных мероприятий работает корректно и при проверке правильно выдает пользователю ошибки.

Результаты функционального тестирования остальных частей web-приложения представлено в приложении Ж.

5.2 Тестирование метрики валидации формы по созданию развлекательного мероприятия

Один из наиболее распространенных подходов к прогнозированию надежности программ основывается на анализе их текста. Этот подход впервые был изложен в книге М. Х. Холстеда «Начала науки о программах», предложенные им характеристики известны как «метрики Холстеда». Они относятся к эмпирическим моделям, которые можно построить на этапе проектирования ПО, когда уже осуществлена разбивка на модули и известна структура программ [13].

Для тестирования использовалась функция, хранящаяся в хранилище состояния приложения, через которую проходят все заполняемые поля перед проверкой. Код представлен на листинге 11.

Листинг 11 – Код функции по валидации полей при создании мероприятия

```
validateItems = (items) => {
  let isValid = true;

  Object.keys(items).forEach(key => {
    if (!items[key] || !items[key].length || items[key] ===
    0) {
      isValid = false;
    }
  });

  return isValid;
};
```

Для оценки сложности процедуры по метрике Холстеда, необходимо посчитать количество общих и уникальных операторов и операндов. Операторы представлены в таблице 19, операнды – в таблице 20.

Таблица 19 – Операторы процедуры

Операторы	Число вхождений
=	2
===	1
!	2

Продолжение таблицы 19

Операторы	Число вхождений
.	3
forEach	1
If	1
Return	1
	2
Object.keys	1
Object	1
Object.keys().forEach()	1
;	4
Length	1
N1	21

Таблица 20 – Операнды процедуры

Операнды	Число вхождений
isValid	3
Items	5
Key	4
True	1
False	1
0	1
N2	15

Длина функции рассчитывается по формуле:

$$N = N1 + N2,$$

где N1 – число вхождений операторов процедур;

N2 – число вхождений операндов процедур.

$$N = 21 + 15 = 26.$$

Словарь функции рассчитывается по формуле:

$$n = n1 + n2,$$

где n1 – количество операторов,

n_2 – количество операндов.

$$n = 13 + 6 = 19.$$

Объем функции рассчитывается по формуле:

$$HV = N * \log_2 n,$$

$$HV = 36 * 4,248 = 152,928.$$

Потенциальный объём функции рассчитывается по формуле:

$$V^* = (2 + n) * \log_2(2 + n),$$

$$V^* = 5 * \log_2(5) = 11,6.$$

Теоретический уровень программы рассчитывается по формуле:

$$L = \frac{V^*}{HV},$$

$$L = \frac{11,6}{152,928} = 0,076.$$

Сложность функции рассчитывается по формуле:

$$HD = \left(\frac{n_1}{2}\right) * \left(\frac{N_2}{n_2}\right),$$

$$HD = \left(\frac{13}{2}\right) * \left(\frac{15}{6}\right) = 16,25.$$

Сложность процедуры приблизительно равна 16, поэтому можно считать её простой.

5.3 Автоматизированное тестирование валидации формы по созданию мероприятия

Автоматизированное тестирование проводилось с использованием JS фреймворка – Jest.

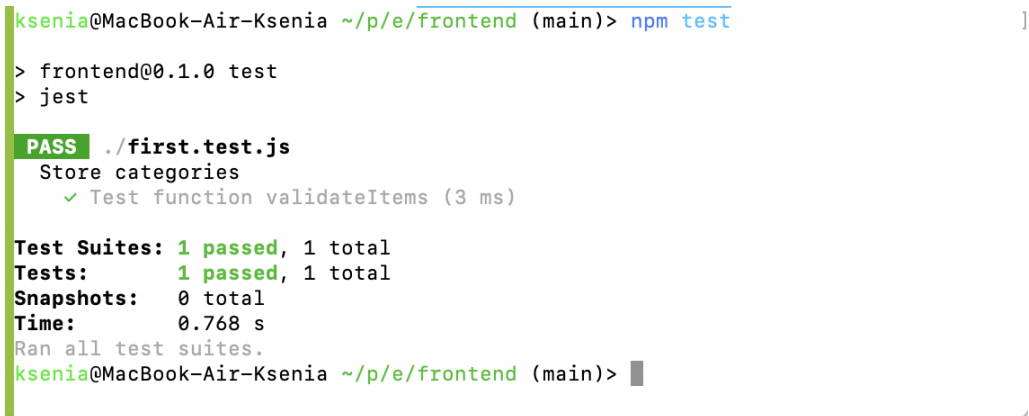
Jest – это фреймворк для тестирования JavaScript, разработанный для обеспечения уверенности в правильной работе любого JavaScript кода. Он позволяет вам писать тесты с приемлемым, знакомым и функциональным API, и быстро достигать желаемых результатов. Данный фреймворк также расширяем под определенные запросы [14].

Написан тест для функции по проверке на ошибки формы по созданию мероприятия и представлен на листинге 12.

Листинг 12 – Тест для функции по проверке на ошибки формы

```
describe("Store categories", function() {
  it("Test function validateItems", function() {
    expect(validateItems({ Name: 'Moovies' })).toBe(true);
    expect(validateItems({ Name: ' ' })).toBe(false);
    expect(validateItems({ Name: null })).toBe(false);
    expect(validateItems({ Name: 0 })).toBe(false);
  });
});
```

Результат работы теста представлен на рисунке 25.



```
ksenia@MacBook-Air-Ksenia ~/p/e/frontend (main)> npm test
> frontend@0.1.0 test
> jest
PASS ./first.test.js
  Store categories
    ✓ Test function validateItems (3 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.768 s
Ran all test suites.
ksenia@MacBook-Air-Ksenia ~/p/e/frontend (main)>
```

Рисунок 25 – Результат тестирования функции по валидации

В результате запуска можно увидеть, что тестирование прошло успешно при прописанных тест-кейсах. Тест-кейсы включали в себе проверку на заполненное значение, пустое значение, пустую строку и нулевое значение.

Заключение

В результате выполнения выпускной квалификационной работы была разработана система по подбору и продаже билетов на развлекательные мероприятия.

Был произведен анализ требований к выполняемой работе. Также проведен анализ с продуктами-аналогами выбранной предметной области, после чего выявлены их недостатки и составлены основные моменты функционала, которые требуются к реализации в данной работе. Функциональными особенностями приложения были предложены «Аналитика данных» и «Согласование мероприятий», позволяющие улучшить конкурентоспособность рынка сервисов по продажам билетов на развлекательные мероприятия.

Был составлен бюджет на разрабатываемое приложение, в ресурсы которого входили «Project-менеджер», «Frontend-разработчик», «Backend-разработчик», «Тестировщик» и «Дизайнер». На каждого выделена дневная ставка и распланировано количество часов на выполнение поставленных требований. В результате планирования бюджета было определено, что всего разработка приложения займет 313 часов работы, а затраты на реализацию системы составят 95 150 рублей.

При планировании были определены технологии, которые будут использоваться при реализации приложения. Для баз данных выбрана MongoDB и составлены логические модели, для серверной части выбран язык программирования C#, фреймворк ASP.NET, микросервисная архитектура и составлены диаграммы классов, а для клиентского приложения выбрана библиотека React, составлена диаграмма компонентов и спроектирован интерфейс, подходящего под удобный пользовательский опыт.

Для разработанного приложения было осуществлено функциональное и автоматизированное тестирование с использованием библиотеки Jest, в ходе которых ошибок работы приложения обнаружено не было. Для исходного кода системы были рассчитаны метрики, чтобы оценить его качество.

Список использованных источников

1. IDEF0 // Wikipedia [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/IDEF0> (дата обращения: 16.05.2023).
2. Элементы графической нотации диаграммы вариантов использования // Intuit [Электронный ресурс]. – URL: <https://intuit.ru/studies/courses/32/32/lecture/1004> (дата обращения: 19.05.2023).
3. UML // Skillfactory [Электронный ресурс]. – URL: <https://blog.skillfactory.ru/glossary/uml/> (дата обращения: 17.05.2023).
4. Гамма, Э. Приемы объектно-ориентированного программирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес – С. :Питер, 2001. – 368 с.
5. Клеппман, М. Высоконагруженные приложения / М. Клеппман – С. :Питер, 2018. – 640 с.
6. ASPNET 6 // Metanit [Электронный ресурс]. – URL: <https://metanit.com/sharp/aspnet6/1.1.php> (дата обращения: 17.05.2023).
7. Крис, Р. Микросервисы. Паттерны разработки и рефакторинга / Р. Крис – С. :Питер, 2019. – 542 с.
8. Что такое React и как его освоить // Yandex Academy [Электронный ресурс]. – URL: <https://academy.yandex.ru/journal/chto-takoe-react-i-kak-ego-osvoit> (дата обращения: 19.05.2023).
9. MongoDB // Skillfactory [Электронный ресурс]. – URL: <https://blog.skillfactory.ru/glossary/mongodb/> (дата обращения: 19.05.2023).
10. Диаграмма классов // Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2 (дата обращения: 20.05.2023).
11. React // Mozilla Developer [Электронный ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started (дата обращения: 20.05.2023).

12. Что такое Black Box Testing // Habr [Электронный ресурс]. – URL: <https://habr.com/ru/company/ruvds/blog/700858/> (дата обращения: 29.05.2023).
13. Холстед М. Х., Начала науки о программах / М. Холстед – М.: Финансы и статистика, 1981. – 128 с.
14. Jest // JestJs [Электронный ресурс]. – URL: <https://jestjs.io/ru/> (дата обращения: 29.05.2023).

Приложение А
ГЛОССАРИЙ
(справочное)

SPA – это web-приложение, использующий единственный HTML-документ, как оболочку для всех web-страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript.

Микросервисная архитектура – вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие нескольких это возможно небольших, слабо связанных и легко изменяемых модулей – микросервисов.

Ocelot – это набор ПО промежуточного слоя, которое можно применять в определенном порядке.

IDE – это программное приложение, которое помогает программистам эффективно разрабатывать программный код.

DOM – это представление пользовательского интерфейса в приложении.

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

Приложение Б
ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

Листин Б.1 – Страница профиля

```

const Profile = observer(() => {
  const router = useNavigate();

  const [value, setValue] = useState(0);
  const [open, setOpen] = useState(false);
  const [comment, setComment] = useState('');
  const [status, setStatus] = useState('');
  const [event, setEvent] = useState({});

  const handleChange = (_, newValue) => {
    setValue(newValue);
  };

  useEffect(() => {
    events.getEventsByUser('user');
    booking.getBook(auth.user.emailAddress);
    booking.getSeller(auth.user.emailAddress);
    if (auth.user.role === 'user') {
      approvals.getApprovals(auth.user.emailAddress);
    } else {
      approvals.getAllApprovals();
    }
  }, []);

  const handleClickDelete = (id) => {
    booking.deleteBook(id);
  };

  const handleApproveStatus = (e, event, status) => {
    setOpen(!open);
    setStatus(status);
    setEvent(event);
  };

  const handleChangeComment = (e) => {
    setComment(e);
  };

  const handleApprove = () => {
    const newStatusApprove = {
      id: event.id,
      Status: status,
      Comments: {
        User: event.comments.user,
        Admin: comment,
      },
    },
  };

  approvals.updateStatus(newStatusApprove);

```

```

};

const renderTabs = () => {
  return (
    <div>
      <Tabs value={ value } onChange={ handleChange }>
        <Tab label="Бронирования" id={ 0 } />
        <Tab label="Согласования" id={ 1 } />
        <Tab label="Аналитика" id={ 2 } />
      </Tabs>
      <TabPanel value={ value } index={ 0 }>
        {
          booking.booking ? booking.booking.map((item) => {
            const { id } = item;

            return (
              <CardProfile
                key={ id }
                event={ item }
                withEdit={ false }
                onClickDelete={ handleClickDelete }
              />
            );
          })
          : <Text text='Сейчас здесь нет бронирований :('
style='text-a2' />
        }
      </TabPanel>
      <TabPanel value={ value } index={ 1 }>
        {
          approvals.approvals ? approvals.approvals.map((item)
=> {
            const { id } = item;

            return (
              <CardProfile
                key={ id }
                role={ auth.user.role }
                status={ item.status }
                event={ item }
                withEdit={ auth.user.role === 'Admin' }
                onClickDelete={ handleClickDelete }
                onApprove={ handleApproveStatus }
              />
            );
          })
          : <Text text='Сейчас здесь нет согласований :('
style='text-a2' />
        }
      </TabPanel>
      <TabPanel value={ value } index={ 2 }>
        <Analytics booking={ booking.booking } sell={
booking.seller } />

```

```

        </TabPanel>
    </div>
  );
};

return (
  <div>
    <Header />
    <div className={ styles.main__header }>
      <div className={ styles.avatar }>
        <img src={ require('../assets/events.webp') }
className={ styles.avatar__img } />
      </div>
      <Text text={ auth.user.emailAddress } style={
TEXT_STYLE['A2-b'] } margin={ '20px 0 0 0' } />
    </div>
    <div className={ styles.main }>
      { renderTabs() }
    </div>
    <Dialog onClose={ () => setOpen(!open) } open={ open }>
      <div className={ styles.dialog }>
        <Text text='Введите комментарий к согласованию' style={
TEXT_STYLE['A2-b'] } />
        <Input
          placeholder='Комментарий'
          valueInput={ comment }
          onChange={ handleChangeComment }
        />
        <Button label='Отправить' onClick={ handleApprove }
buttonType='button-fill' />
      </div>
    </Dialog>
  </div>
);
});

export default Profile;

```

Листинг Б.2 – Основная страница

```

const minDistance = 100;

const Main = observer(() => {
  const router = useNavigate();

  const maxPrice = events.items.length ?
    events.items.reduce((prev, current) => (prev.price >
current.price) ? prev : current).price :
    10000;

  const [priceRange, setPriceRange] = useState([0, maxPrice]);

```

```

const [category, setCategory] = useState(null);
const [dateRange, setDateRange] = useState([null, null]);

useEffect(() => {
  auth.getCurrentUser();
  events.getEvents();
  categories.getCategories();
}, []);

const handleChangePrice = (_, newValue, activeThumb) => {
  if (!Array.isArray(newValue)) {
    return;
  }

  if (activeThumb === 0) {
    setPriceRange([Math.min(newValue[0], priceRange[1] -
minDistance), priceRange[1]]);
  } else {
    setPriceRange([priceRange[0], Math.max(newValue[1],
priceRange[0] + minDistance)]);
  }
};

const handleClickItem = (item) => {
  router(`/event/${item.id}`);
};

const handleRequestFilter = () => {
  const data = {
    priceRange: {
      from: priceRange[0],
      to: priceRange[1],
    },
    category,
    dateRange: {
      from: dateRange[0],
      to: dateRange[1],
    },
  };

  events.getFilteredItems(data);
};

const handleSelectCategory = (value) => {
  setCategory(value.nativeEvent.target.outerText);
};

const handleChangeDate = (value, type) => {
  if (type === 'from') {
    setDateRange([value, dateRange[1]]);
  } else {
    setDateRange([dateRange[0], value]);
  }
}

```

```

};

return (
  <div>
    <Header />
    <div className={ styles.poster }>
      <img className={ styles.poster__img } src={
require('../../assets/events.webp') }></img>
    </div>
    <div className={ styles.filters }>
      <PriceFilter
        priceRange={ priceRange }
        maxPrice={ maxPrice }
        handleChangePrice={ handleChangePrice }
        handleRequestFilter={ handleRequestFilter }
      />
      <CategoryFilter
        selectedCategory={ category }
        categories={ categories.items }
        handleSelectCategory={ handleSelectCategory }
        handleRequestFilter={ handleRequestFilter }
      />
      <DateFilter
        selectedDate={ dateRange }
        handleChangeDate={ handleChangeDate }
        handleRequestFilter={ handleRequestFilter }
      />
    </div>
    <div>
      <Text text='Юмористические выступления' style='text-a2' />
      <MenuCards items={ events.items.filter(item => item.type
=== 'Humor') } onClick={ handleClickItem } />
    </div>
    <div>
      <Text text='Театральные постановки' style='text-a2' />
      <MenuCards items={ events.items.filter(item => item.type
=== 'Musical') } onClick={ handleClickItem } />
    </div>
    <div>
      <Text text='Концерты' style='text-a2' />
      <MenuCards items={ events.items.filter(item => item.type
=== 'Concert') } onClick={ handleClickItem } />
    </div>
    <div>
      <Text text='Квесты' style='text-a2' />
      <MenuCards items={ events.items.filter(item => item.type
=== 'Quest') } onClick={ handleClickItem } />
    </div>
  </div>
);
});

export default Main;

```

Приложение В
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг В.1 – Контроллер авторизации

```

namespace account_service.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LoginController : ControllerBase
    {
        private readonly IConfiguration _config;

        private readonly UsersService _usersCollection;

        /// <summary>
        /// UsersService декларируется в Program.cs
        /// </summary>
        /// <param name="config"></param>
        /// <param name="userService"></param>
        public LoginController(IConfiguration config, UsersService
usersService)
        {
            _config = config;

            _usersCollection = usersService;
        }

        /// <summary>
        /// Получение Пользователей
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public async Task<List<UserModel>> Get() =>
            await _usersCollection.GetAsync();

        /// <summary>
        /// Нахождение пользователя и генерация JWT токена
        /// </summary>
        /// <param name="userLogin"></param>
        /// <returns></returns>
        [AllowAnonymous]
        [HttpPost]
        public async Task<IActionResult> Login([FromBody]
UserLogin userLogin)
        {
            var user = await _usersCollection.GetAsync(userLogin);

            if (user != null)
            {
                var token = Generate(user);
                return Ok(token);
            }

            return NotFound("User not found");
        }
    }
}

```

```

    }

    /// <summary>
    /// Генерация JWT токена
    /// </summary>
    /// <param name="user"></param>
    /// <returns></returns>
    private string Generate(UserModel user)
    {
        var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
        var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

        var claims = new[]
        {
            new Claim(ClaimTypes.NameIdentifier,
user.Username),
            new Claim(ClaimTypes.Email, user.EmailAddress),
            new Claim(ClaimTypes.GivenName, user.GivenName),
            new Claim(ClaimTypes.Surname, user.Surname),
            new Claim(ClaimTypes.Role, user.Role),
            new Claim(ClaimTypes.DateOfBirth, user.Birthday),
            new Claim("Interests", user.Interests),
        };

        var token = new
JwtSecurityToken(_config["Jwt:Issuer"],
        _config["Jwt:Audience"],
        claims,
        expires: DateTime.UtcNow.AddMinutes(20),
        signingCredentials: credentials);

        return new
JwtSecurityTokenHandler().WriteToken(token);
    }

    /// <summary>
    /// Регистрация
    /// </summary>
    /// <param name="newUser"></param>
    /// <returns></returns>
    [HttpPost("Create")]
    public async Task<IActionResult> Post(UserModel newUser)
    {
        await _usersCollection.CreateAsync(newUser);

        return CreatedAtAction(nameof(Get), new { login =
newUser.Username }, newUser);
    }
}

```

Листинг В.2 – Контроллер бронирования

```

namespace booking_service.Controllers
{
    [Route("/api/[controller]")]
    public class BookingController : Controller
    {
        private readonly BookingService _bookingService;
        private readonly EventsService _eventsService;

        public BookingController(BookingService bookingService,
EventsService eventsService)
        {
            _bookingService = bookingService;

            _eventsService = eventsService;
        }

        /// <summary>
        /// Получение всех бронирований
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public async Task<List<Booking>> Get() =>
            await _bookingService.GetAsync();

        /// <summary>
        /// Получение по id бронирования
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("{id:length(24)}")]
        public async Task<ActionResult<Booking>> Get(string id)
        {
            var booking = await _bookingService.GetAsync(id);

            if (booking is null)
            {
                return NotFound();
            }

            return booking;
        }

        /// <summary>
        /// Получение по id бронирования
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("{buyer}/buyer")]

```

```

=> public async Task<List<Booking>?> GetByBuyer(string buyer)

        await _bookingService.GetByBuyerAsync(buyer);

    /// <summary>
    /// Получение по userId бронирований
    /// </summary>
    /// <param name="userId"></param>
    /// <returns></returns>
    [HttpGet("{userId}/user")]
    public async Task<List<Booking>?> GetByUser(string userId)
=>
        await _bookingService.GetByUserAsync(userId);

    /// <summary>
    /// Добавление бронирования
    /// </summary>
    /// <param name="newBook"></param>
    /// <returns></returns>
    [HttpPost("{id:length(24)}")]
    public async Task<IActionResult> Post(string id,
[FromBody] BookingPost newBook)
    {
        var events = await _eventsService.GetAsync(id);

        if (events is not null)
        {
            EventsViewModel newEvent = new EventsViewModel
            {
                Id = events.Id,
                Title = events.Title,
                Price = events.Price,
                Type = events.Type,
                UserId = events.UserId,
                Description = events.Description,
                Location = events.Location,
                Seats = newBook.EventSeat,
                Date = events.Date,
                ProfilePicture = events.ProfilePicture,
            };

            await _eventsService.UpdateAsync(events.Id,
newEvent);
        }

        Booking newBooking = new Booking
        {
            Title = events.Title,
            Price = events.Price,
            Type = events.Type,
            UserId = events.UserId,
            Buyer = newBook.Buyer,
            Description = events.Description,

```

```

        Location = events.Location,
        BookingSeat = newBook.BookingSeat,
        Date = events.Date,
        DateCreate = DateTime.Today,
        ProfilePicture = events.ProfilePicture,
    };

    await _bookingService.CreateAsync(newBooking);

    return CreatedAtAction(nameof(Get), new { id =
newBook.Id }, newBook);
}

/// <summary>
/// Отмена бронирования
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
[HttpDelete("{id:length(24)}")]
public async Task<IActionResult> Delete(string id)
{
    var booking = await _bookingService.GetAsync(id);

    if (booking is null)
    {
        return NotFound();
    }

    await _bookingService.RemoveAsync(id);

    return NoContent();
}
}
}

```

Листинг В.3 – Контроллер мероприятий

```

namespace lab2_VS.Controllers
{
    [Route("/api/[controller]")]
    public class EventsController : Controller
    {
        private readonly EventsService _eventsService;
        public static IWebHostEnvironment _environment;

        public EventsController(EventsService eventsService,
IWebHostEnvironment environment)
        {
            _eventsService = eventsService;

            _environment = environment;
        }
    }
}

```

```

    /// <summary>
    /// Получение всех мероприятий
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public async Task<List<EventsViewModel>> Get() =>
        await _eventsService.GetAsync();

    /// <summary>
    /// Получение по id мероприятия
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    [HttpGet("{id:length(24)}")]
    public async Task<ActionResult<EventsViewModel>>
Get(string id)
    {
        var events = await _eventsService.GetAsync(id);

        if (events is null)
        {
            return NotFound();
        }

        return events;
    }

    /// <summary>
    /// Получение по userId мероприятия
    /// </summary>
    /// <param name="userId"></param>
    /// <returns></returns>
    [HttpGet("{userId}/user")]
    public async Task<List<EventsViewModel>?> GetByUser(string
userId) =>
        await _eventsService.GetByUserAsync(userId);

    /// <summary>
    /// Удаление мероприятия
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    [HttpDelete("{id:length(24)}")]
    public async Task<IActionResult> Delete(string id)
    {
        var events = await _eventsService.GetAsync(id);

        if (events is null)
        {
            return NotFound();
        }
    }

```

```

    }

    await _eventsService.RemoveAsync(id);

    return NoContent();
}

/// <summary>
/// Добавление мероприятия
/// </summary>
/// <param name="eventItem"></param>
/// <returns></returns>
[HttpPost]
public async Task<IActionResult> Post([FromForm] Events
newEvent)
{
    string uniqueFileName = UploadedFile(newEvent);

    EventsViewModel eventNew = new EventsViewModel
    {
        Title = newEvent.Title,
        Price = newEvent.Price,
        Type = newEvent.Type,
        UserId = newEvent.UserId,
        Description = newEvent.Description,
        Location = newEvent.Location,
        Seats = newEvent.Seats,
        Date = newEvent.Date,
        ProfilePicture = uniqueFileName,
    };
    await _eventsService.CreateAsync(eventNew);

    return CreatedAtAction(nameof(Get), new { id =
newEvent.Id }, eventNew);
}

private string UploadedFile(Events model)
{
    string uniqueFileName = null;

    if (model.ProfilePicture != null)
    {
        string uploadsFolder =
Path.Combine(_environment.WebRootPath, "images");
        uniqueFileName = Guid.NewGuid().ToString() + "_" +
model.ProfilePicture.FileName;
        string filePath = Path.Combine(uploadsFolder,
uniqueFileName);
        using (var fileStream = new FileStream(filePath,
FileMode.Create))
        {
            model.ProfilePicture.CopyTo(fileStream);
        }
    }
}

```

```

        }
        return uniqueFileName;
    }

    [HttpGet("image/{name}")]
    public IActionResult GetImage(string name)
    {
        var path = Path.Combine(_environment.WebRootPath,
            "images", $"{name}");
        var imageFileStream = System.IO.File.OpenRead(path);
        return File(imageFileStream, "image/jpeg");
    }

    /// <summary>
    /// Изменение мероприятия
    /// </summary>
    /// <param name="eventItem"></param>
    /// <returns></returns>
    [HttpPut("{id:length(24)}")]
    public async Task<IActionResult> Update(string id,
        [FromForm] Events updatedEvent)
    {
        var events = await _eventsService.GetAsync(id);

        if (events is null)
        {
            return NotFound();
        }

        updatedEvent.Id = events.Id;

        string uniqueFileName = UploadedFile(updatedEvent);

        EventsViewModel eventUpdated = new EventsViewModel
        {
            Id = updatedEvent.Id,
            Title = updatedEvent.Title,
            Price = updatedEvent.Price,
            Type = updatedEvent.Type,
            UserId = updatedEvent.UserId,
            Description = updatedEvent.Description,
            Location = updatedEvent.Location,
            Seats = updatedEvent.Seats,
            Date = updatedEvent.Date,
            ProfilePicture = uniqueFileName,
        };

        await _eventsService.UpdateAsync(updatedEvent.Id,
            eventUpdated);

        return NoContent();
    }

```

```

        [HttpPost("filters")]
        public async Task<IActionResult>
        GetEventByFilters([FromBody] Filters filters)
        {
            var events = await _eventsService.GetAsync();

            if (!String.IsNullOrEmpty(filters.Category))
            {
                events = events.FindAll(item =>
item.Type.Equals(filters.Category));
            }

            if (filters.PriceRange is not null)
            {
                events = events.FindAll(item => item.Price >=
filters.PriceRange.From && item.Price <= filters.PriceRange.To);
            }

            if (filters.DateRange is not null)
            {
                events = events.FindAll(item => item.Date >=
filters.DateRange.From && item.Date <= filters.DateRange.To);
            }

            return Ok(events);
        }
    }
}

```

Приложение Г
ОПИСАНИЕ БАЗ ДАННЫХ
(рекомендуемое)

Таблица Г.1 – Описание MongoDB БД таблиц

База данных	Таблицы	Поля	Тип данных
Events	Events	_id	BsonType.ObjectId
		Title	String
		Type	String
		UserId	String
		Description	String
		Location	Object: { City: string, Street: string, Latitude: string, Longitude: string, }
		Seats	Array: { Name: string, Count: Decimal128, Price: Decimal128, }
		Date	Date
		ProfilePicture	String
	Approvals	_id	BsonType.ObjectId
		Type	String
		UserId	String
		Description	String
		Location	String
		Seats	Object: { City: string, Street: string, Latitude: string, Longitude: string, }
		Date	Array: { Name: string, Count: Decimal128, Price: Decimal128, }
		ProfilePicture	Date
		Status	String
		Comments	Object: { User: string, Admin: string, }
	Favorites	_id	BsonType.ObjectId
		Title	String
		Type	String
		UserId	String
		Description	String
		Location	Object: { City: string, Street: string, Latitude: string,

Продолжение таблицы Г.1

База данных	Таблицы	Поля	Тип данных
		Seats	Array: { Name: string, Count: Decimal128, Price: Decimal128, }
		Date	Date
		ProfilePicture	String
	History	_id	BsonType.ObjectId
		Title	String
		Type	String
		UserId	String
		Description	String
		Location	Object: { City: string, Street: string, Latitude: string, Longitude: string, }
		Seats	Array: { Name: string, Count: Decimal128, Price: Decimal128, }
		Date	Date
		ProfilePicture	String
		_id	BsonType.ObjectId
		Title	String
		Type	String
Booking	Booking	UserId	String
		Description	String
		Location	Object: { City: string, Street: string, Latitude: string, Longitude: string, }
		BookingSeat	Array: { Name: string, Count: Decimal128, Price: Decimal128, }
		Date	Date
		ProfilePicture	String
		Buyer	String
		DateCreate	Date

Окончание таблицы Г.1

База данных	Таблицы	Поля	Тип данных
Users	Users	_id	BsonType.ObjectId
		Username	String
		Password	String
		EmailAddress	String
		Role	String
		Surname	String
		GivenName	String
		Birthday	String
		Interests	String
Categories	Categories	_id	BsonType.ObjectId
		Name	String

Приложение Д
ДИАГРАММЫ КЛАССОВ
(рекомендуемое)

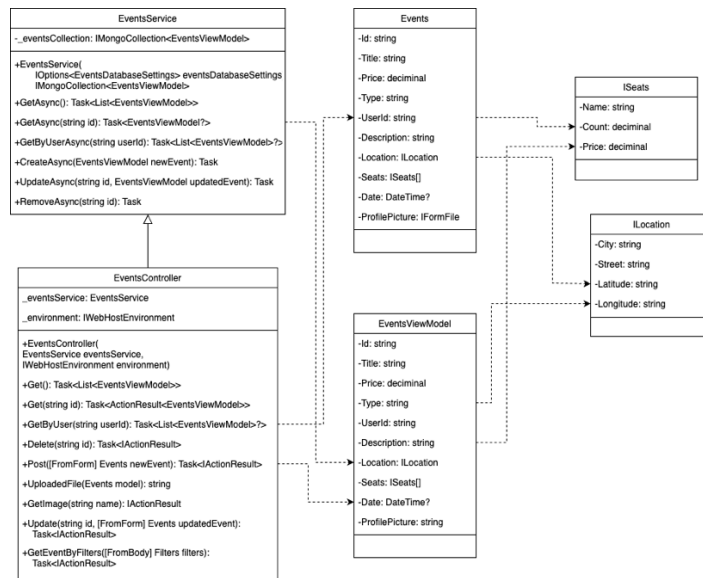


Рисунок Д.1 – Диаграмма классов для микросервиса мероприятий

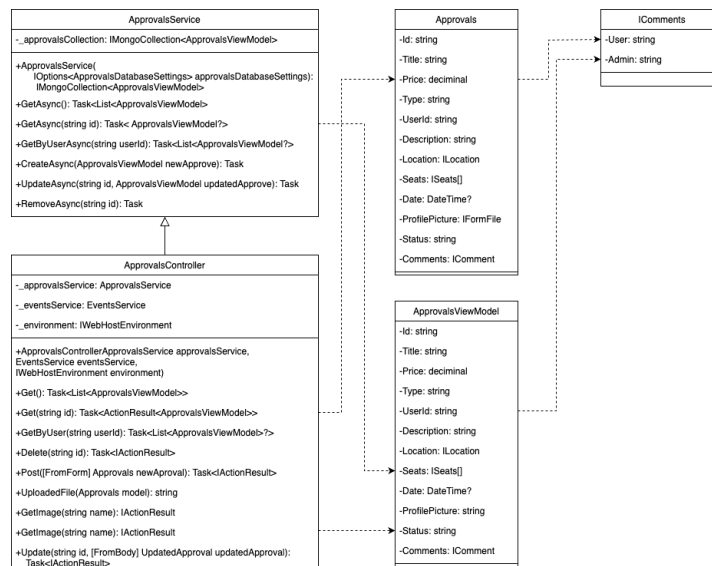


Рисунок Д.2 – Диаграмма классов для микросервиса согласований

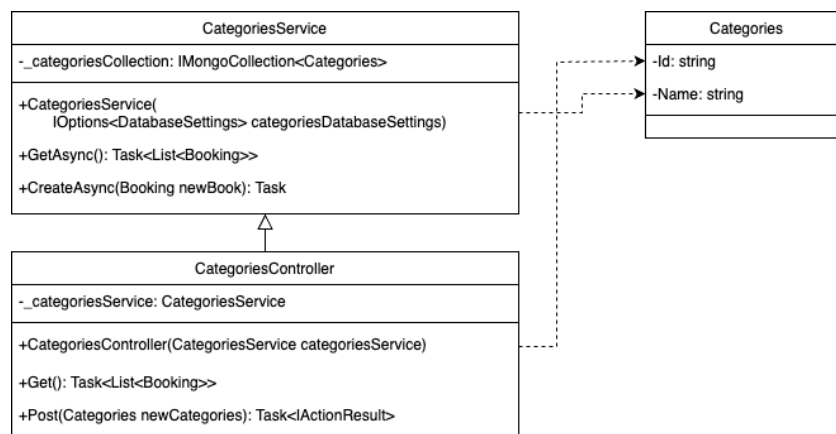


Рисунок Д.3 – Диаграмма классов для микросервиса категорий

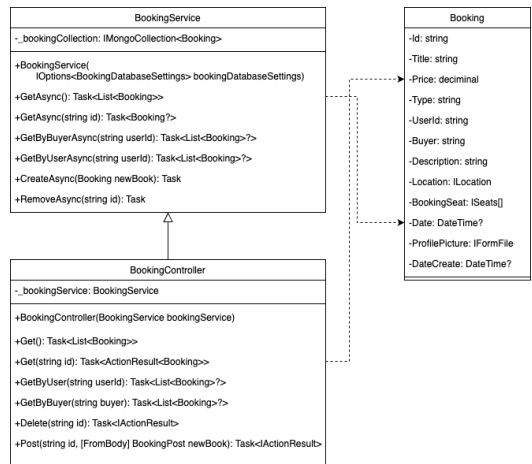


Рисунок Д.4 – Диаграмма классов для микросервиса бронирований

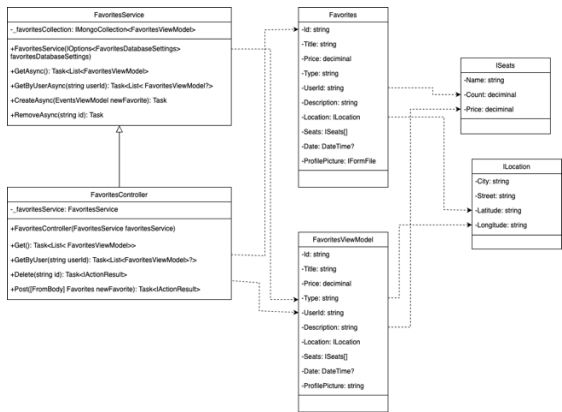


Рисунок Д.5 – Диаграмма классов для микросервиса избранных мероприятий

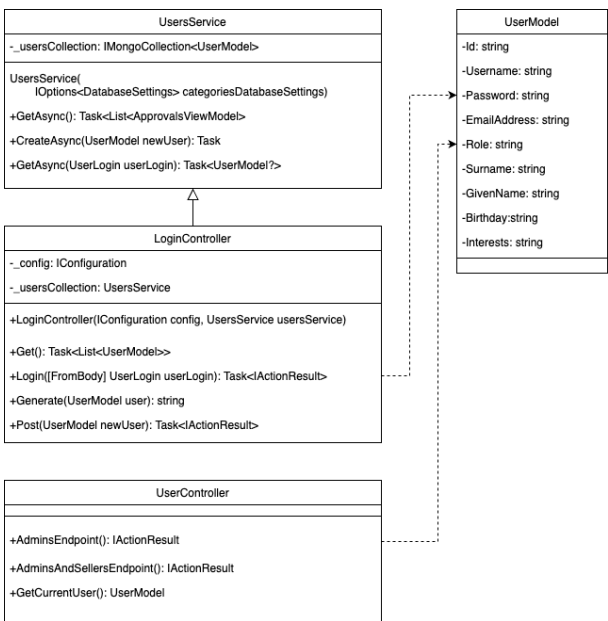


Рисунок Д.6 – Диаграмма классов для микросервиса с пользователями

Приложение Е
ЛОГИЧЕСКИЕ МОДЕЛИ БАЗ ДАННЫХ
(рекомендуемое)

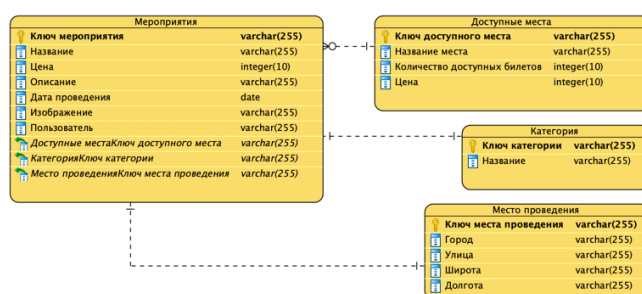


Рисунок Е.1 – Логическая модель БД для мероприятий

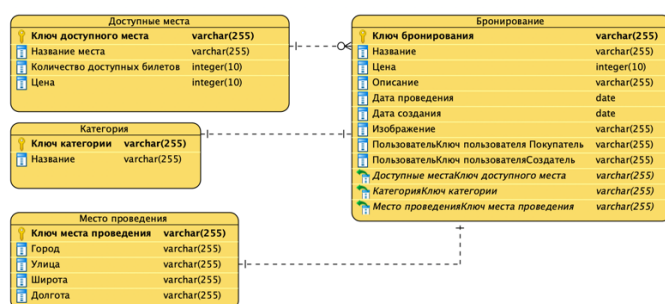


Рисунок Е.2 – Логическая модель БД для бронирований



Рисунок Е.3 – Логическая модель БД для пользователей



Рисунок Е.4 – Логическая модель БД для избранных мероприятий

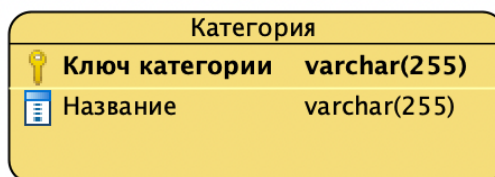


Рисунок Е.5 – Логическая модель БД для категорий мероприятий

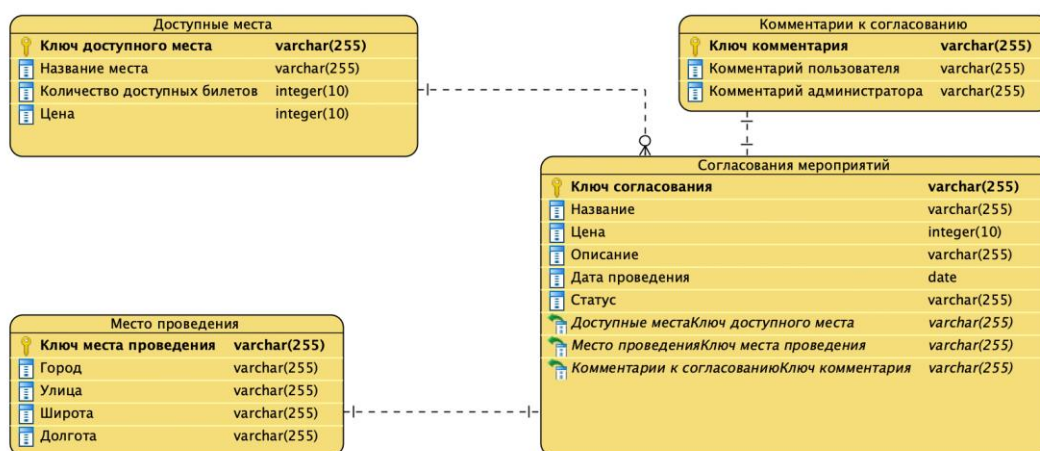


Рисунок Е.5 – Логическая модель БД для согласований мероприятий

Приложение Ж
ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ
(рекомендуемое)

Страница регистрации

Результат валидации страницы регистрации с основной информацией представлен на рисунке Ж.1.

Регистрация

Авторизоваться

Общая информация

Интересы

Фамилия *

Имя *

Отчество

Дата рождения *

Email *

Пароль *

←

→

Зарегистрироваться

Рисунок Ж.1 – Валидация основной информации

Результат валидации страницы регистрации с информацией о интересах представлен на рисунке Ж.2.

Регистрация

Авторизоваться

Общая информация

Интересы

Выберите интересы

Кино

Stand-up

Музыка

Наука

Театр

←

→

ЗАРЕГИСТРИРОВАТЬСЯ

Выберите не менее 3х интересов

Рисунок Ж.2 – Валидация информации о интересах

Страница авторизации

Результат валидации страницы авторизации при не существовании введенного пользователя представлен на рисунке Ж.3.

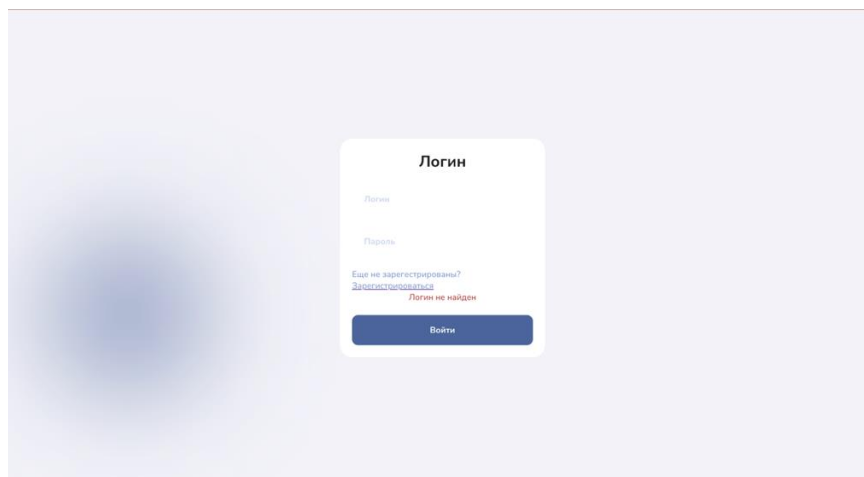


Рисунок Ж.3 – Валидация процесса авторизации

Главная страница

Результат фильтрованных данных с полученными данными (рисунок Ж.4).

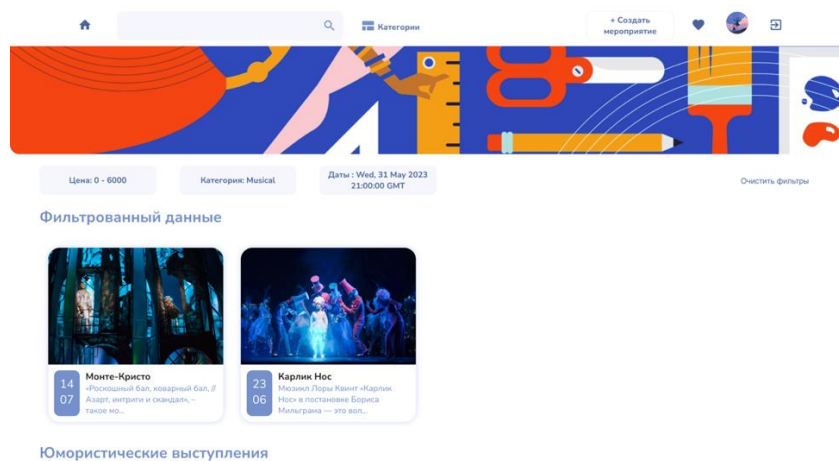


Рисунок Ж.4 – Мероприятия по выбранным фильтрам

Результат фильтрованных данных с пустым результатом (рисунок Ж.5).

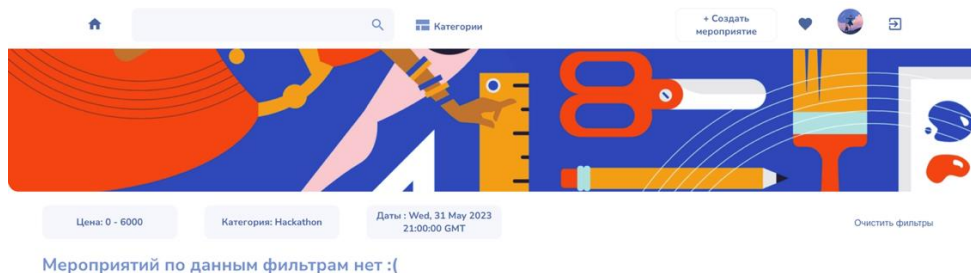


Рисунок Ж.5 – Результат получения пустых данных по фильтрам

Страница создания мероприятия

Валидация при создании мероприятия (рисунок Ж.6).

The screenshot displays a web form for creating an event. At the top, there is a navigation bar with a home icon, a search bar, a 'Категории' (Categories) link, and a '+ Создать мероприятие' (Create event) button. The form itself is organized into several sections. The first section contains three input fields: 'Название' (Name), 'Описание' (Description), and 'Цена' (Price). The 'Название' field has a validation error message 'Название' below it. The second section contains three input fields: 'Категория' (Category), 'Город' (City), and 'Улица' (Street). The 'Категория' dropdown is set to 'Musical'. The 'Город' field has a validation error message 'Город' below it. The third section contains three input fields: 'Широта' (Latitude), 'Долгота' (Longitude), and 'Дата' (Date). The 'Широта' field has a validation error message 'Широта' below it. The fourth section contains two input fields: 'Изображение' (Image) and 'Комментарий' (Comment). The 'Изображение' field has a validation error message 'Выберите файл' (Select a file) and a file name '(2).jpeg'. The 'Комментарий' field has a validation error message 'Комментарий' below it. Below these sections, there is a section titled 'Доступные места' (Available places) with a 'Добавить место' (Add place) button. At the bottom, there is a message 'Не заполнены поля' (Fields are not filled) and a large blue 'Добавить' (Add) button.

Название

Описание

Цена

Категория

Город

Улица

Широта

Долгота

Дата

Изображение

Комментарий

Доступные места

Добавить место

Не заполнены поля

Добавить

Рисунок Ж.6 – Валидация создания