МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное приложение для городской афиши событий

Студент _____ Новиков Иван Александрович
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ 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	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов «__»
_____2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Новиков Иван Александрович гр. 19ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Мобильное приложение для городской афиши событий

Тема утверждена приказом ПГУ №_____от "___"_____2023 г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) Назначение: приложение для предоставления информации о городских культурно-досуговых мероприятях.

2) Среда выполнения: операционная система Android 6 и выше

Среда выполнения сервера: операционная система Windows 10 и выше

Языковые средства разработки: C#, UML, SQL

Инструментальные средства разработки: Microsoft SQL Server 2018, *Microsoft Visual Studio 2019, Microsoft Project, Draw.io, Visual Paradigm*

Технологии разработки: ООП, WPF, Entity Framework, Xamarin Forms

3) Основные функции мобильного приложения:

а) регистрация и авторизация пользователей;

б) просмотр всех мероприятий;

в) просмотр мероприятий по категориям;

г) поиск мероприятий по названию;

д) бронирование билетов на мероприятия;

е) просмотр забронированных билетов пользователем.

5. Перечень подлежащих разработке вопросов:

1) анализ предметной области и постановка задачи;

2) проектирование;

3) *реализация;*

4) *качества программного обеспечения;*

5) *план разработки бюджета;*

6) *оформление пояснительной записки*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	14.04.2023 – 27.04.2023	
2	Проектирование	25%	27.04.2023 – 10.05.2023	
3	Реализация	35%	10.05.2023 – 30.06.2023	
4	Контроль качества программного обеспечения	10%	31.05.2023 – 02.06.2023	
5	План разработки бюджета	5%	02.06.2023 – 03.06.2023	
6	Оформление пояснительной записки	15%	03.06.2023 – 14.06.2023	

Дата выдачи задания «22» ____декабря____2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 111 листов, 58 рисунков, 11 таблиц, 12 использованный источник и 5 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ВИЗУАЛИЗАЦИЯ, КЛИЕНТ, СЕРВЕР ПРИЛОЖЕНИЙ, БАЗА ДАННЫХ, UML, C#.

Объектом разработки является средства автоматизации информирования населения о культурной жизни города.

Цель работы – создание мобильного приложения афиши городских событий, позволяющего пользователям просматривать информацию о мероприятиях, бронировать билеты.

Разработка серверной части происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2019. В качестве СУБД была использована СУБД MS SQL Server.

Клиентское приложение разрабатывалось на языке программирования C# в среде Microsoft Visual Studio 2019, с использованием фреймворка Xamarin Forms. Оно спроектировано для работы на устройствах под управлением операционной системы Android версии 6.0 и выше, а серверная его часть работает на операционной системе Windows 10 и выше.

Содержание

Введение 7

1 Анализ предметной области и требований приложения городской афиши 9

1.1 Анализ предметной области и постановка задачи приложения 9

1.2 Сравнительный анализ мобильных приложений городских событий 10

1.2.1 Мобильное приложение Афиша 10

1.2.2 Мобильное приложение «Киноафиша» 11

1.2.3 Мобильное приложение Афиша 7 11

1.3 Сравнительный анализ продуктов-аналогов 12

1.4 Анализ функциональных требований 14

1.5 Анализ нефункциональных требований 19

2 Проектирование мобильного приложения городских событий 20

2.1 Проектирование архитектуры программного обеспечения 20

2.2 Проектирование интерфейса пользователя 20

2.3 Проектирование структур данных 24

2.3.1 Концептуальная модель базы данных сервиса 24

2.3.2 Логическая модель базы данных сервиса 28

2.4 Проектирование серверной части приложения 29

3 Реализация мобильного приложения городской афиши 30

3.1 Реализация базы данных 30

3.2 Разработка серверной части 30

3.3 Реализация клиентской части приложения 33

4 Тестирование программного продукта 37

4.1 Тестирование серверной части приложения 37

4.2 Тестирование клиентской части приложения 39
4.2.1 Функциональное тестирование 39
4.2.2 Тестирование приложения методом черного ящика 40
4.2.3 Модульное тестирование 42
4.3 Метрики кода 45
4.3.1 Метрики размера программы 47
4.3.2 Метрики Холстеда 47
4.4 Планирование разработки и оценки бюджета 50
Заключение 53
Список использованных источников 54
Приложение А. Исходный код скриптов базы данных 55
Приложение Б. Исходный код серверной части 58
Приложение В. Исходный код мобильного приложения 70
Приложение Г. Результаты тестирования серверной части 97
Приложение Д. Результаты тестирования мобильного приложения 104

Введение

Современное общество активно развивается в области информационных и мобильных технологий, и мобильные устройства и интернет оказывают все большее влияние на нашу жизнь. В эпоху цифровых технологий создается множество программных продуктов, специально адаптированных для использования на мобильных устройствах. Современные передовые модели электронных устройств обладают высоким функционалом и характеристиками, не уступающими даже настольным компьютерам.

Мобильные приложения для поиска предстоящих городских событий становятся все более актуальными в нашей жизни. Современный человек живет в быстром темпе и постоянно находится в движении, и ему необходимо быть в курсе всех событий, которые происходят в его городе. Мобильные приложения на эту тему позволяют получать информацию о различных мероприятиях, концертах, выставках и других событиях, которые проходят в городе. Они также помогают пользователю быстро и удобно найти нужную информацию о событии, купить билеты или забронировать место. Благодаря мобильным приложениям, освещающим городскую афишу событий, люди могут легко планировать свое свободное время и не пропускать интересные мероприятия. Кроме того, подобные мобильные приложения способствуют развитию культурной жизни города, увеличению числа посетителей на мероприятиях и распространению информации о важных городских событиях. Таким образом, мобильные приложения для городской афиши событий становятся важным инструментом для общественности, помогая людям организовать свободное время и провести его с пользой. Этим и определяется значимость и актуальность исследования.

Объект исследования – средства автоматизации информирования населения о культурной жизни города.

Предмет исследования – мобильное приложение для городской афиши событий.

Целью данной выпускной квалификационной работы является разработка мобильного приложения городской афиши событий для повышения информированности жителей города о проходящих мероприятий. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения.

1 Анализ предметной области и требований приложения городской афиши

1.1 Анализ предметной области и постановка задачи приложения

Современный ритм жизни требует от людей быть в курсе всех событий, происходящих в городе.

Мобильное приложение городских событий является отличным инструментом для того, чтобы оперативно получать информацию о том, что происходит в городе, и не пропускать интересные мероприятия.

Благодаря этому приложению пользователи могут получить актуальную информацию по проходящим мероприятиям в городе. Это позволит пользователям быть в курсе о всех городских событиях, что является весьма удобным для всех интересующихся людей.

Приложение позволяет пользователям быстро получить актуальную информацию про городские мероприятия. Пользователи могут посмотреть информацию о мероприятии, забронировать и купить билет, зарегистрироваться и авторизоваться, производить поиск по названию события.

Таким образом, в приложении должны быть представлены следующие функции:

– регистрация новых пользователей;
– авторизация пользователей;
– просмотр мероприятий;
– поиск мероприятий;
– разделение мероприятий по категориям;
– просмотр свободных мест;
– бронирование билетов;
– отзывы и рейтинги.

1.2 Сравнительный анализ мобильных приложений городских событий

1.2.1 Мобильное приложение Афиша

«Афиша» – это медийно-сервисная платформа о развлечениях и культурном досуге горожанина [1]. Главное меню мобильного приложения представлено на рисунке 1.

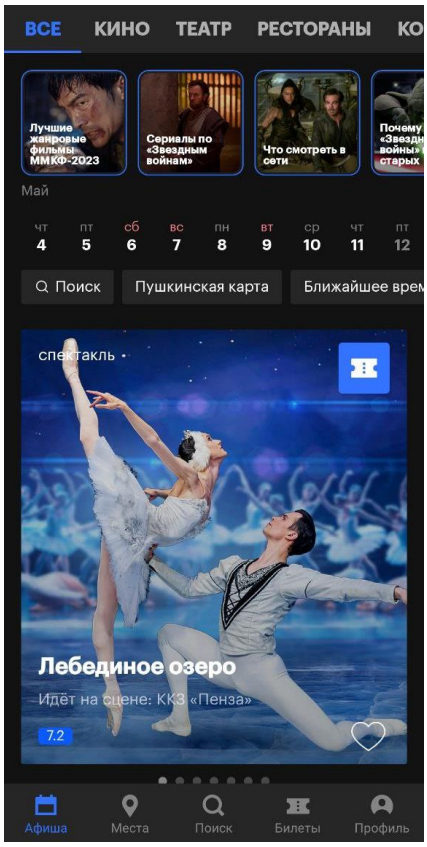


Рисунок 1 – Главное меню приложение «Афиша»

Данная платформа имеет достаточно широкие функциональные возможности, которые позволяют просматривать мероприятия, фильтровать по категориям, оставлять отзывы и оценки. Также есть возможность бронирования и покупки билетов.

Основными недостатками приложения являются загруженность главной страницы информацией, при этом данная информация не всегда актуальна. Не смотря на возможность бронирования и покупки билетов, не на все мероприятия их можно приобрести.

1.2.2 Мобильное приложение «Киноафиша»

«Киноафиша» – это сервис, который содержит информацию о кинотеатрах и фильмах, которые находятся в городском прокате [2]. Главное меню мобильного приложения представлено на рисунке 2.

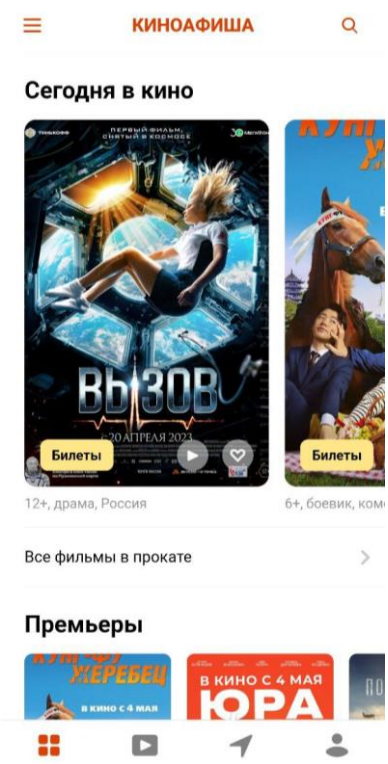


Рисунок 2 – Главное меню приложения «Киноафиша»

Данная платформа имеет достаточно разнообразные функциональные возможности, которые позволяют просматривать афишу в кинотеатрах, оставлять отзывы и оценки, бронировать и покупать билеты.

Несмотря на плюсы, ресурс также содержит и минусы, такие как предоставление информации только о кинотеатрах, не на все сеансы можно приобрести билеты, не всегда обновленная информация.

1.2.3 Мобильное приложение Афиша 7

«Афиша7» – проект о культурных и спортивных событиях России. Он рассчитан на продвижение мероприятий, выставок, концертов, спектаклей, спортивных соревнований, цирковых выступлений, мастер-классов [3].

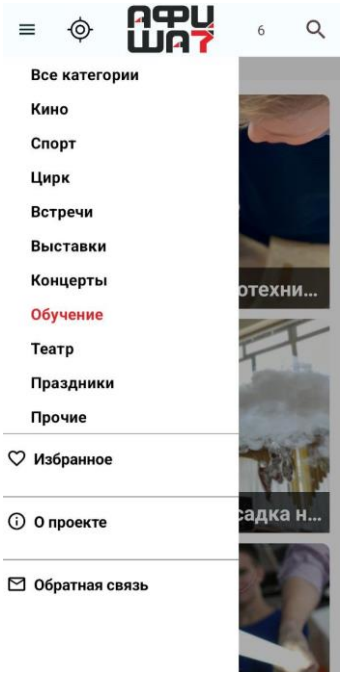


Рисунок 3 – Главное меню приложения Афиша 7

Данный проект имеет достаточно широкий функционал, который позволяют просматривать информацию по мероприятиям города, оставлять отзывы и оценки, бронировать и покупать билеты.

В данном приложении можно отметить и такие минусы, как интерфейс приложение, можно сказать, что он не вполне удобный, не всегда присутствует возможность приобрести билеты на городское событие. А также информация о мероприятиях, размещаемая в приложении, не всегда актуальна.

1.3 Сравнительный анализ продуктов-аналогов

В результате анализа программных продуктов с информацией о городских событиях, которые являются приближенными аналогами разрабатываемого мобильного приложения.

Для проведения сравнительного анализа, в качестве критериев, выбраны наиболее важные возможности, которые напрямую влияют на функциональность сервиса, а именно:

а) функциональные возможности:

1) возможность регистрации и авторизации;

2) возможность поиска;

3) возможность просмотра по категориям;

4) возможность оставлять отзывы;

5) возможность бронирование и покупки билетов;

б) нефункциональные возможности:

1) качество контента;

2) бесплатность продукта;

в) технические возможности:

1) хранение данных в базе данных.

Для оценки описанных выше критериев используется трехбалльная шкала соответствия, 0 – «функция не реализована», 1 – «функция частично реализована», 2 – «функция реализована».

Для анализа были отобраны следующие мобильные приложения «Афиша», «Киноафиша», «Афиша 7».

Заполним таблицу 1, оценив реализованный функционал описанных выше систем. Затем произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{пп} = \sum_{i=1}^{n}(Z_i),$$

где $O_{пп}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия;

Таблица 1 – Критериальная оценка программных продуктов

№	Критерий	Название программного продукта		
		«Афиша»	«Киноафиша»	«Афиша 7»
1	Возможность регистрации и авторизации	2	2	2
2	Возможность поиска	2	2	2
3	Возможность просмотра по категориям	2	0	1

Продолжение таблицы 1

№	Критерий	Название программного продукта		
		«Афиша»	«Киноафиша»	«Афиша 7»
4	Возможность оставлять отзывы	1	2	0
5	Возможность бронирования билетов	1	1	1
6	Качество контента	1	1	1
7	Бесплатность продукта	1	1	2
8	Хранение данных в БД	2	2	2
	Интегральная оценка	12	11	11

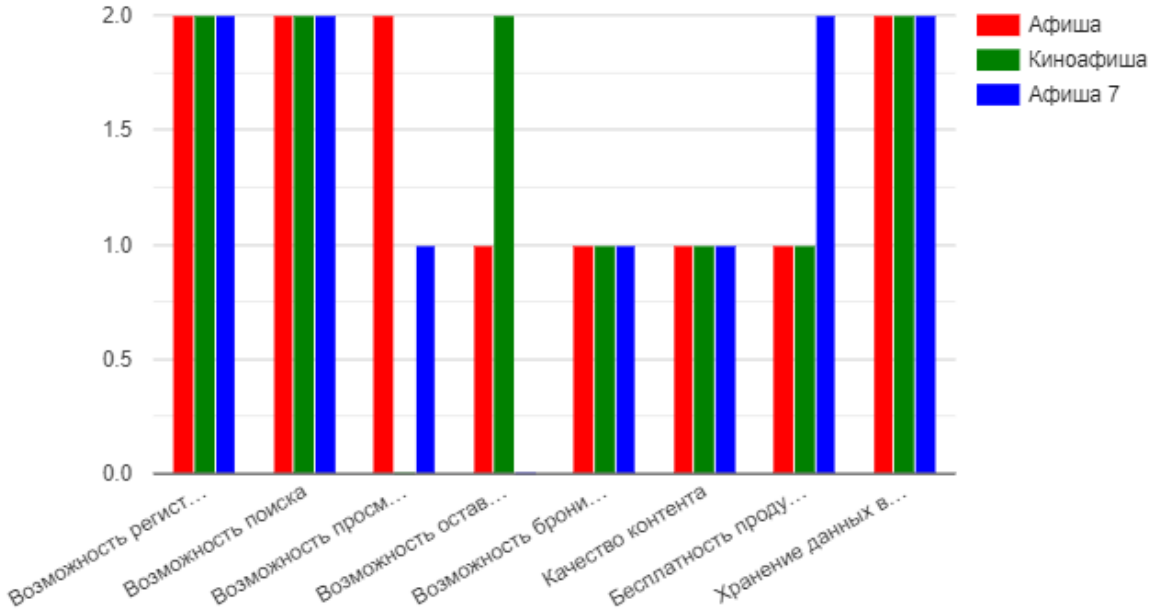


Рисунок 4 – Гистограмма критериальной оценки программных продуктов

Из данной таблицы можно сделать вывод, что лидером является приложение «Афиша». Конечно оно, как и все остальные рассматриваемые варианты имеют недостатки в функционале. С небольшим отрывом «Киноафиша» и «Афиша 7» занимают второе место.

1.4 Анализ функциональных требований

Обобщив сказанное выше, для разрабатываемого приложения выделены следующие основные требования:

– регистрация и авторизация пользователей – пользователи могут зарегистрироваться в приложении, используя свои учетные данные или социальные сети;

– поиск событий – пользователи могут искать события по различным категориям, таким как концерты, спортивные мероприятия, выставки и т. д.;

– фильтры по категориям – пользователи могут перейти в ту категорию, которая его интересует;

– комментарии и оценки – пользователи могут оставлять комментарии и оценки для событий, которые они посетили, чтобы помочь другим пользователям принять решение о посещении;

– бронирование билетов – пользователи могут бронировать билеты на события через приложения.

На рисунке 5 представлена диаграмма вариантов использования.

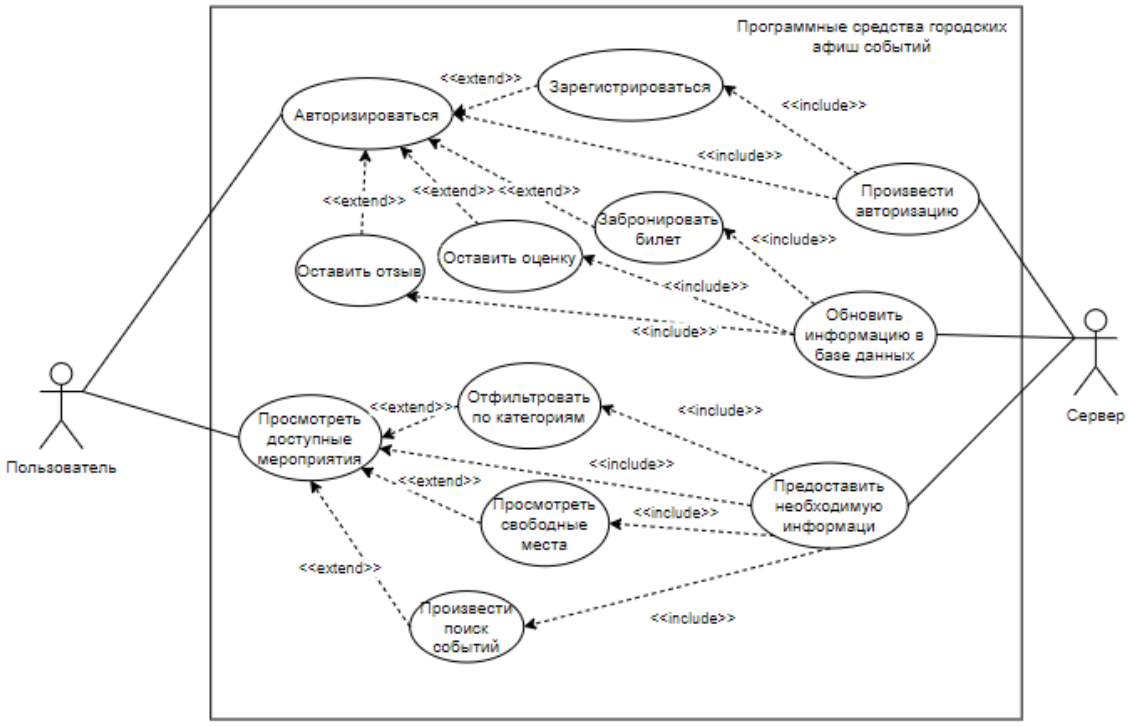


Рисунок 5 – Диаграмма вариантов использования

Сценарии наиболее важных вариантов использования представлены на рисунках 6 – 8.

На рисунке 6 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Посмотреть доступные мероприятия», позволяющий посмотреть список мероприятий, как на главной странице, так и по категориям.

Прецедент: Просмотреть доступные мероприятия
ID: 1
Краткое описание: На главном экране отображаются доступные мероприятий.
Главные актеры: Пользователь, сервер
Второстепенные актеры: Нет.
Предусловие: Сервер запущен.
Основной поток: 1. Пользователю выводится лента с мероприятиями. 2. Пользователь может открыть мероприятий для подробного ознакомления.
Альтернативный потоки: Нет.
Постусловия: Пользователь ознакомлен с мероприятиями.

Рисунок 6 – Спецификация для варианта использования «Посмотреть доступные мероприятия»

На рисунке 7 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Посмотреть свободные места», позволяющий посмотреть список мест на выбранное мероприятие.

Прецедент: Посмотреть свободные места
ID: 2
Краткое описание: Выводится информация со свободными местами.
Главные актеры: Пользователь, сервер
Второстепенные актеры: Нет.
Предусловие: Сервер запущен, на мероприятие должны быть свободные билеты.
Основной поток: 1. Пользователь выбрал мероприятия. 2. Пользователь выбрал дату и время. 3. Пользователь посмотрел свободные места.
Альтернативный потоки: Нет.
Постусловия: Пользователь ознакомлен со свободными местами.

Рисунок 7 – Спецификация для варианта использования «Посмотреть свободные места»

На рисунке 8 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Зарегистрироваться», позволяющий зарегистрировать пользователя в системе.

Прецедент: Зарегистрироваться
ID: 3
Краткое описание: Пользователь регистрируется в системе.
Главные актеры: Пользователь, сервер
Второстепенные актеры: Нет.
Предусловие: Сервер запущен.
Основной поток: 1. Пользователь открыл приложение. 2. Перешел на страницу регистрации 3. Пользователь вводит псевдоним, логин и пароль. 4. Приложение производит валидацию данных 5. Приложение отправляет данные на сервер для регистрации. 6. Сервер проверяет наличие пользователя с аналогичным логином 7. Сервер сохраняет данные в базу данных. 8. Сервер отправляет успешный ответ. 9. Приложение переходит на главную страницу.
Альтернативный потоки: 4а. Если данные не прошли валидацию, приложение выводит сообщение об ошибке и возвращается к пункту 2. 6а. Если пользователь с аналогичным логином уже существует, сервер отправляет сообщение об ошибке и возвращается к пункту 2.
Постусловия: Пользователь зарегистрирован в системе.

Рисунок 8 – Спецификация для варианта использования «Зарегистрироваться»

На рисунке 9 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Авторизоваться», позволяющий зарегистрировать пользователя в системе.

Прецедент: Авторизоваться
ID: 4
Краткое описание: Пользователь пытается войти в свой зарегистрированный аккаунт в мобильном приложении
Главные актеры: Пользователь, сервер
Предусловие: Пользователь должен быть зарегистрирован ранее.
Второстепенные актеры: Нет.
Основной поток: 1. Пользователь открывает страницу авторизации. 2. Пользователь вводит свой логин и пароль. 3. Мобильное приложение проводит валидацию введенных данных на стороне клиента. 4. Мобильное приложение отправляет запрос на сервер для проверки соответствия введенных данных зарегистрированному аккаунту. 5. Сервер проверяет соответствие введенных данных с зарегистрированным аккаунтом. 6. Если данные верны, сервер отправляет подтверждение авторизации мобильному приложению. 7. Мобильное приложение авторизует пользователя и перенаправляет его на главный экран приложения.
Альтернативный потоки: 3а.Если введенные данные не проходят валидацию на стороне клиента, то переход к шагу 2. 6а. Если введенные данные не соответствуют ни одному зарегистрированному аккаунту, то переход к шагу 2.
Постусловия: Пользователь успешно авторизовался в системе.

Рисунок 9 – Спецификация для варианта использования «Авторизоваться»

Для того чтобы раскрыть прецедента «Авторизироваться», я воспользовался диаграммой деятельности, чтобы структурировать процесс и понять логику его выполнения. Благодаря этому инструменту получилось выявить все необходимые шаги и определить зависимости между ними. Диаграмма деятельности представлена на рисунке 10.

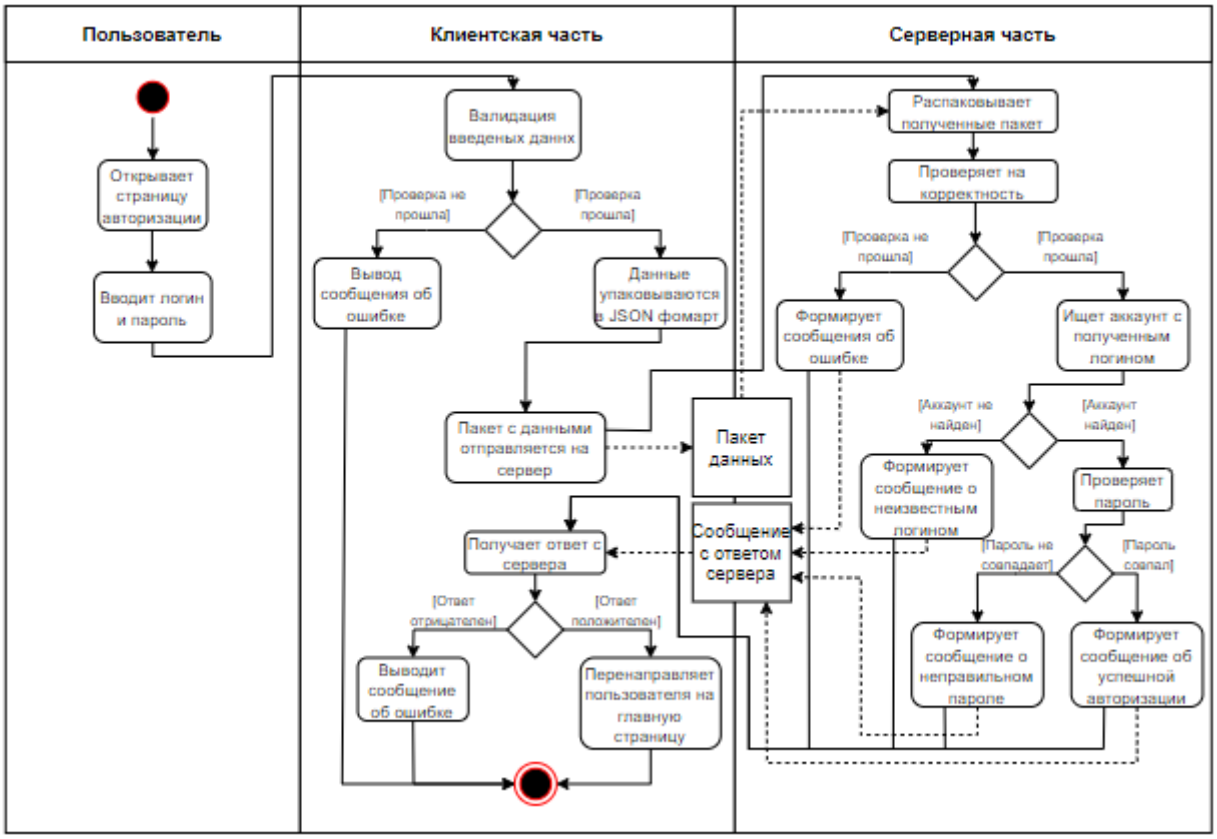


Рисунок 10 – Диаграмма деятельности для прецедента «Авторизироваться»

Путем анализа функциональных требований удалось успешно выявить основные функции, которые должны быть реализованы в разрабатываемом приложении. Для приложения были созданы диаграммы использования и деятельности, которые отражают возможные сценарии использования и порядок выполнения операций. Кроме того, для некоторых определенных вариантов использования были составлены подробные технические характеристики и спецификации.

1.5 Анализ нефункциональных требований

Необходимо написать серверную часть приложения на языке C#, используя Microsoft Visual Studio 2019 в качестве среды разработки. Данные на сервере будут храниться в базе данных, которая будет создана с помощью СУБД Microsoft SQL Server. Для запросов к базе данных следует использовать язык SQL. Серверная часть должна быть совместима с операционной системой Windows 10 и более поздними версиями.

Клиентскую часть приложения следует написать на языке C# с использованием Xamarin Forms. Необходимо разработать пользовательский интерфейс для клиентской части приложения. Интерфейс должен быть удобным и интуитивно понятным для пользователей. Для этого следует использовать современные технологии и подходы, которые обеспечат удобство использования приложения. Кроме того, следует обратить внимание на эстетическую составляющую интерфейса, чтобы он выглядел привлекательно и соответствовал современным тенденциям в дизайне. Приложение должно быть отлажено и успешно работать на устройствах под управлением ОС Android 6.0 и более новых версий.

2 Проектирование мобильного приложения городских событий

2.1 Проектирование архитектуры программного обеспечения

Для данного приложения было принято решение использовать двухуровневую архитектуру «клиент-сервер». Схема представлена на рисунке 11.

Рисунок 11 – Двухуровневая архитектура «клиент-сервер»

В данной архитектуре на сервер содержится база данных и серверная часть приложения, которые взаимодействуют через интерфейс API. Клиентская часть обеспечивает удобный интерфейс для пользователей, который позволяет просматривать и работать с данными, а также удаленно обращаться к серверу.

Взаимодействие между клиентом и сервером осуществляется через HTTP-запросы, при этом сервер создает SQL-запросы для обработки информации в базе данных. Одним из преимуществ такой архитектуры является то, что все данные хранятся на сервере, что позволяет снизить размер клиентского приложения.

2.2 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения. Он включает в себя страницы всех мероприятий, мероприятий по категориям, поиска мероприятия, профиля и с купленными билетами пользователя.

Страница всех мероприятий изображена на рисунке 12. На ней имеются следующие элементы: 1 – переходы на страницы с мероприятиями по

категориям, 2 – отображение мероприятий, 3 – переходы на страницы поиска, профиля и билетов.

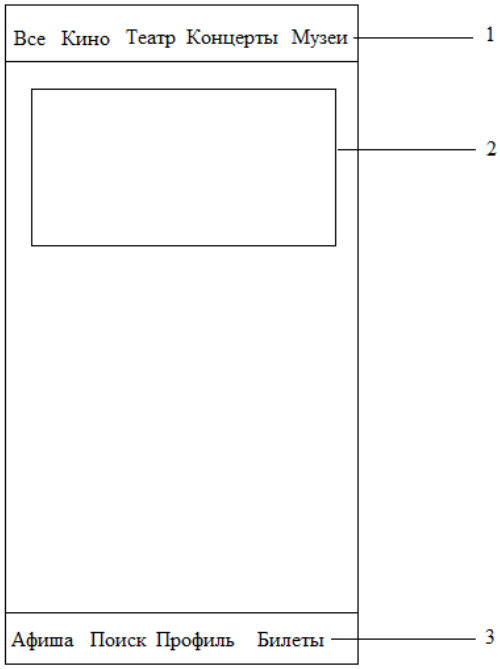


Рисунок 12 – Страница всех мероприятий

Страница профиля не авторизированного пользователя изображена на рисунке 13. Имеются кнопки авторизации и регистрации, также есть возможность перейти на другие страницы.

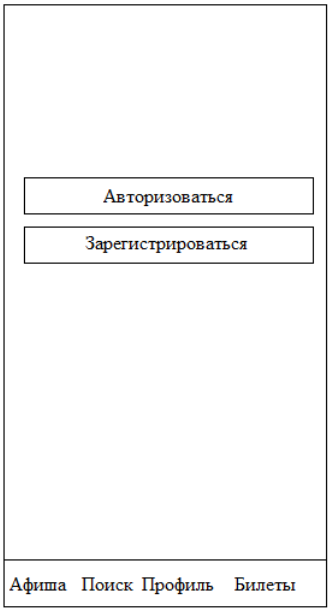


Рисунок 13 – Страница профиля

Страница регистрации пользователя изображена на рисунке 14. Имеет следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля, 3 – поле для ввода псевдонима, 4 – кнопка регистрации. Также присутствует возможность перейти на другие страницы.

Рисунок 14 – Страница регистрации пользователя

Страница авторизации пользователя изображена на рисунке 15. На данной странице присутствуют следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля, 3 – кнопка авторизации, а также есть возможность перейти на другие страницы.

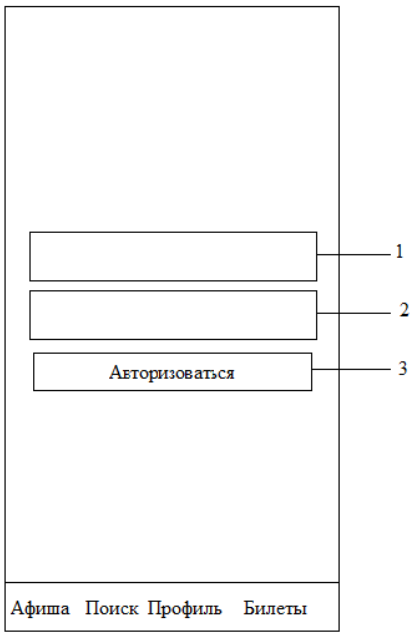


Рисунок 15 – Страница авторизации пользователя

Страница профиля пользователя изображена на рисунке 16. На данной странице присутствуют следующие элементы: 1 – псевдоним пользователя, 2 – фотография пользователя, 3 – вкладки с избранными мероприятиями и оставленными отзывами.

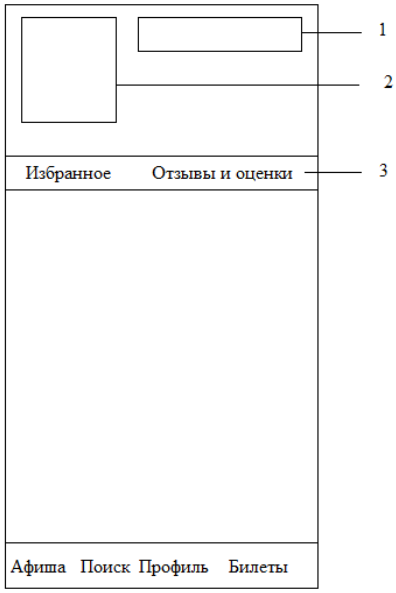


Рисунок 16 – Страница профиля пользователя

Страница поиска изображена на рисунке 17. На странице присутствует строка для поиска мероприятий по названию.

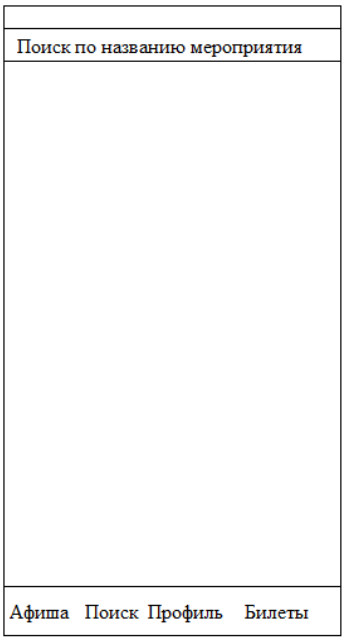


Рисунок 17 – Страница поиска

Страница билетов изображена на рисунке 18. На странице присутствует информация о забронированном билете.

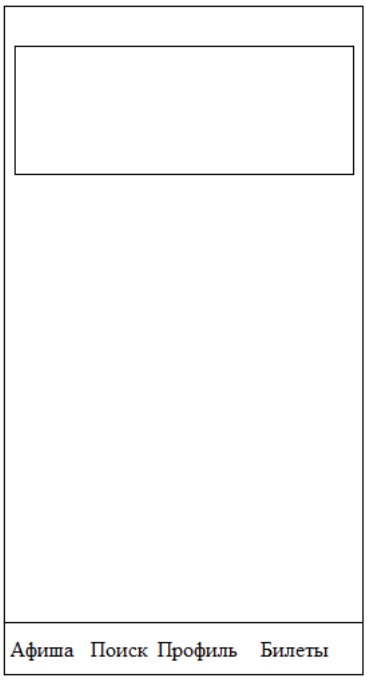


Рисунок 18 – Профиль пользователя

Таким образом, был разработан пользовательский интерфейс, который является важным элементом взаимодействия между пользователем и системой.

2.3 Проектирование структур данных

2.3.1 Концептуальная модель базы данных сервиса

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними независимо от ограничений, вытекающих из моделей данных, поддерживаемая той или иной СУБД [4]. Часто на такой модели отображаются только имена сущностей (таблиц) без указания их атрибутов.

Графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для предметной области информационной системы представлено на рисунке 19.

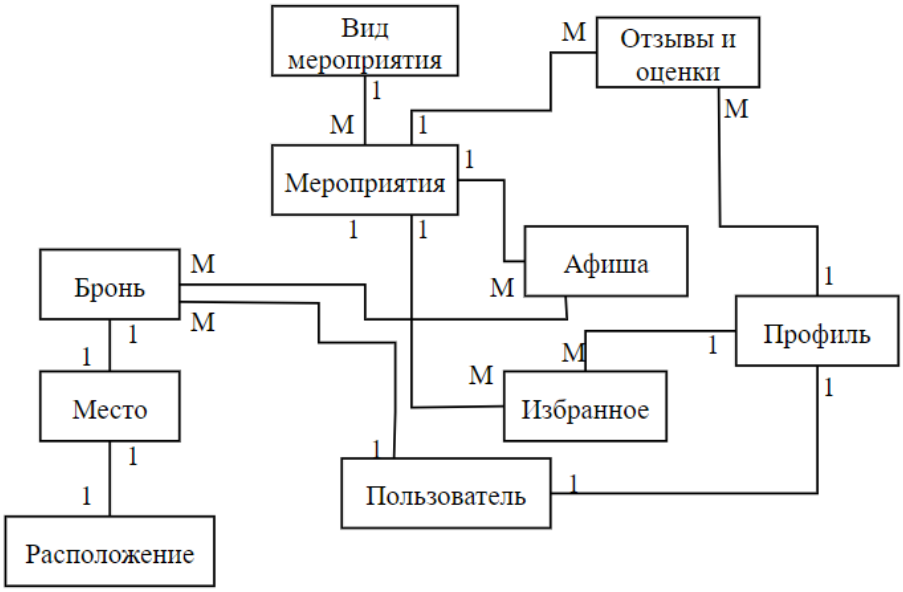


Рисунок 19 – Графическое представление концептуальной модели

В таблице 2 представлены сущности, сформированные на основании входных и выходных данных бизнес-процессов.

Таблица 2 – Сущности предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Список пользователей	Стержневая	500000	В систему добавляются новые пользователи после процедуры регистрации.
2	Профиль	Список профилей пользователей	Стержневая	500000	Добавляется в соответствии с пользователем, отдельно не может существовать.
3	Отзывы и оценки	Список отзывов пользователя	Стержневая	1000	Обновляется, когда пользователь оставляет отзыв или оценку.
4	Избранное	Список избранных мероприятий пользователя	Стержневая	100	Обновляется, когда пользователь отметил мероприятия избранным.
5	Бронь	Список забронированных билетов	Стержневая	10000	Обновляется, когда пользователь бронирует билет.
6	Афиша	Список всех мероприятий	Стержневая	100000	Обновляется при добавлении новых мероприятий.

Продолжение таблицы 2

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
7	Мероприятие	Список мероприятий	Стержневая	10000	Обновляется при добавлении новых мероприятий.
8	Вид мероприятия	Список с видами мероприятий	Стержневая	100	Не обновляется
9	Место	Список с местами	Стержневая	100000	Не обновляется
10	Расположение	Список с расположением мест	Стержневая	100000	Не обновляется

В таблице 3 представлено описание связей между сущностями предметной области. Описание связей между сущностями поможет лучше понять взаимосвязь между элементами предметной области и определить ключевые факторы, влияющие на работу системы или процесса.

Таблица 3 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Пользователь	Профиль	Каждый пользователь имеет один профиль/Каждый профиль относится лишь к одному пользователю	1:1	Обязательная	
2	Пользователь	Бронь	Каждый пользователь может иметь множество забронированных билетов/Забронировать билет на одно место может только один пользователь	1:M	Необязательная (пользователь может не бронировать билеты)	На одно место можно забронировать один билет
3	Профиль	Отзывы и оценки	Каждый пользователь может поставить любое количество отзывов и оценок/Каждая оценка и отзыв ставятся только одним пользователем	1:M	Необязательная (пользователь может не ставить оценки и отзывы)	На одно мероприятие можно поставить одну оценку

Продолжение таблицы 3

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
4	Профиль	Избранное	Один пользователь может добавить много мероприятий в избранное/Каждое мероприятие можно добавить только один раз	1:M	Необязательная (пользователь может не добавлять мероприятия в избранное	Нельзя добавлять несколько раз
5	Мероприятие	Афиша	Одно мероприятие может иметь несколько разных дат в афише /Каждое мероприятие можно добавить только 1 раз	1:M	Обязательная	
6	Афиша	Бронь	На мероприятие в конкретное время и дату может быть много забронированных билетов/Забронированный билет относится к одному времени	1:M	Необязательная	Нельзя Забронировать один и тот же билет
7	Мероприятие	Вид мероприятия	Мероприятие может иметь только один вид/Один вид мероприятия может иметь много мероприятий	M:1	Обязательная	
8	Мероприятие	Отзывы и оценки	Одно мероприятие может иметь много отзывов и оценок/Каждый отзыв или оценка относится к одному мероприятию	1:M	Необязательная (отзывов и оценок может не быть	На одно мероприятие можно поставить одну оценку
9	Мероприятие	Избранное	Одно мероприятие может быть добавлено в избранное много раз/В избранное можно добавить мероприятие только один раз	1:M	Необязательная (мероприятие могут не добавлять в избранное	Нельзя добавлять несколько раз
10	Бронь	Место	Один забронированный билет может иметь только одно место/Место может относится только к одному забронированному билету	1:1	Обязательная	Нельзя забронировать одно и тот же место

Окончание таблицы 3

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
11	Расположение	Место	Расположение имеет только одно такое место/Место только в одном расположении	1:1	Обязательная	Нельзя забронировать одно и тот же место
12	Продажа билетов	Купленные билеты	Один билет можно продать один раз/Один билет можно купить один раз	1:1	Необязательная	Нельзя продавать купленные билеты

На основании концептуальной модели данных в среде Visual Paradigm была построена логическая модель базы данных.

2.3.2 Логическая модель базы данных сервиса

Логическая модель – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила.

На рисунке 20 представлена структура логической модели базы данных.

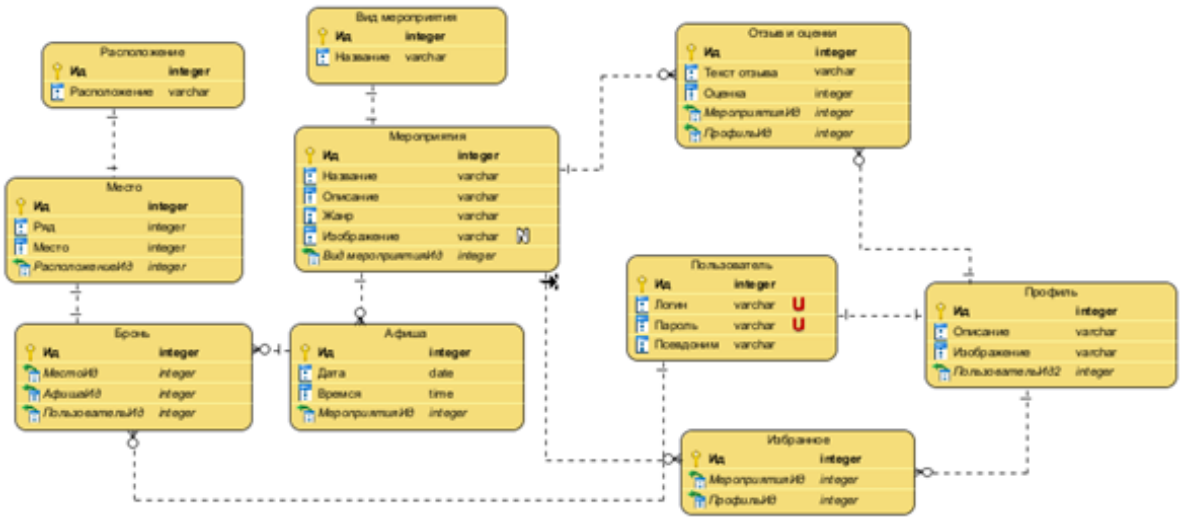


Рисунок 20 – Логическая модель базы данных

В логической модели представлена более подробная структура базы данных. На данной модели четко видно взаимоотношение сущностей между друг другом.

2.4 Проектирование серверной части приложения

Протокол HTTP используется для получения доступа к данным, которые хранятся на серверной части приложения. Для этого используются стандартные методы, такие как GET, POST, PUT и DELETE. Доступ к ресурсам осуществляется посредством URI-идентификаторов.

Архитектура приложения основана на шаблоне проектирования MVC (Model-View-Controller), который состоит из трех компонентов: моделей, контроллеров и представлений. Архитектура представлена на рисунке 21.

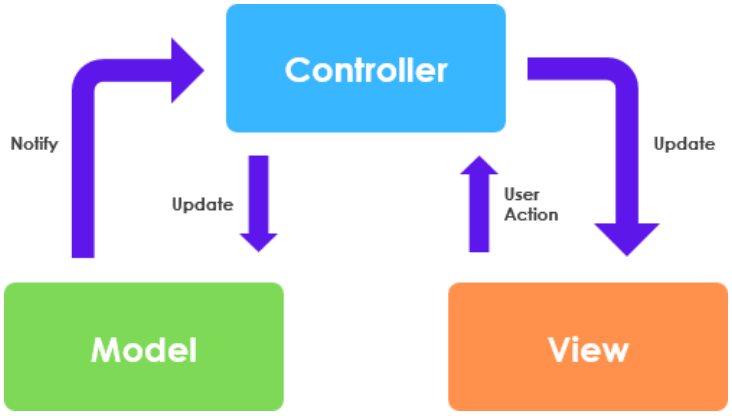


Рисунок 21 – Архитектура MVC

Модели представляют классы, которые обеспечивают доступ к данным и взаимодействуют с базой данных. Контроллеры, в свою очередь, обрабатывают HTTP–запросы и генерируют соответствующие ответы на них. Пользователь взаимодействует только с уровнем Представления, который отвечает за интерфейс и обработку пользовательского ввода.

3 Реализация мобильного приложения городской афиши

3.1 Реализация базы данных

С использованием Visual Paradigm была создана физическая модель базы данных, основанная на предварительно составленной логической модели, учитывающей характеристики СУБД Microsoft SQL Server. Структура этой модели изображена на рисунке 22.

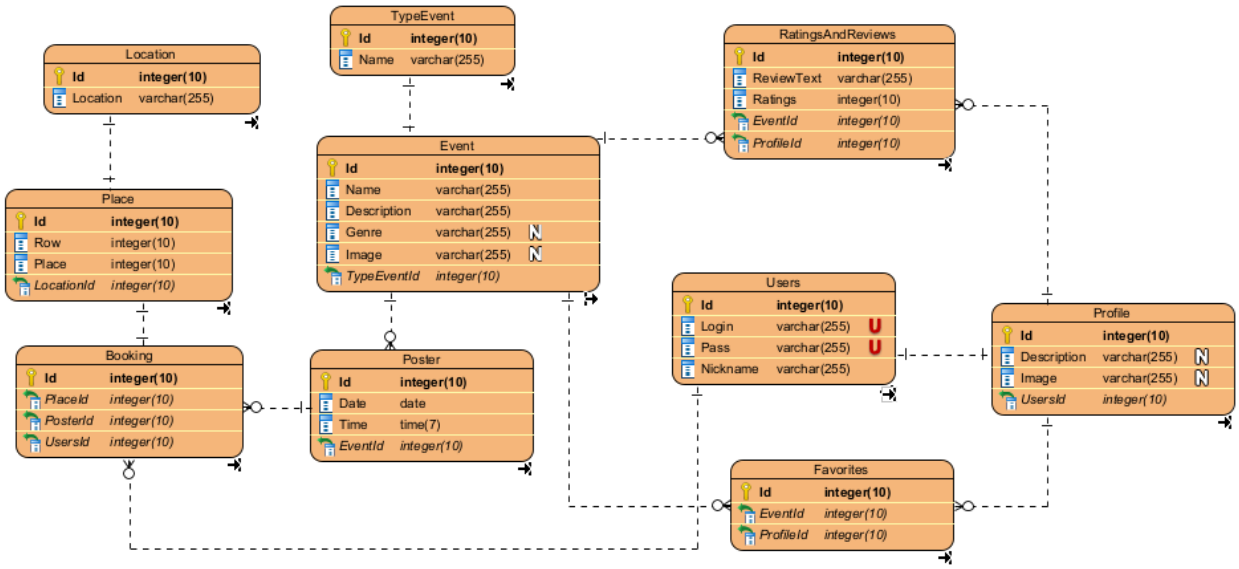


Рисунок 22 – Физическая модель базы данных

Используя сгенерированный на основе физической модели базы данных скрипт на языке SQL, было выполнено развертывание базы данных. Код этого скрипта можно найти в приложении А. Для администрирования баз данных, работающих на СУБД Microsoft SQL Server, была использована утилита SQL Server Management Studio, путем запуска данного скрипта.

3.2 Разработка серверной части

В процессе разработки мобильного приложения, которое работает по клиент-серверной архитектуре, была создана серверная часть с использованием Microsoft Visual Studio 2019 и языка программирования C#. Для реализации серверной части была использована технология Web API [10, 11]. Для взаимодействия с базой данных был выбран Entity Framework [12]. Для

обеспечения безопасности передачи данных между клиентом и сервером был использован протокол HTTPS.

Поскольку ранее была выбрана модель MVC, серверная часть приложения была разделена на два основных слоя: модель и контроллер. Слой модели отвечает за взаимодействие с базой данных и содержит бизнес-логику приложения, в то время как слой контроллеров обрабатывает запросы на сервер и управляет слоем модели. Такой подход позволяет эффективно управлять данными и запросами, а также четко разделить логику приложения от пользовательского интерфейса.

На рисунке 23 представлена диаграмма классов для слоя модели. Классы, которые были созданы, полностью соответствуют таблицам, находящимся в базе данных.

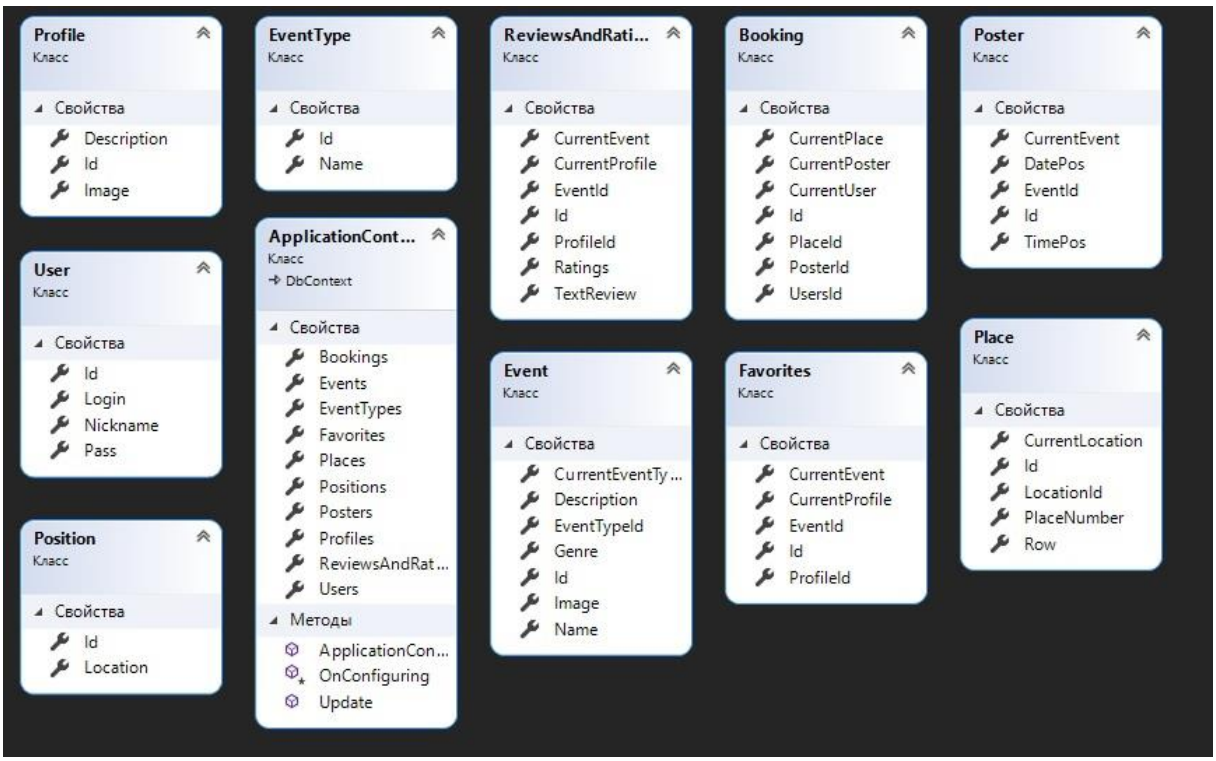


Рисунок 23 – Диаграмма классов серверной части (слой модели)

Диаграмма классов слоя контроллеров представлена на рисунке 24.

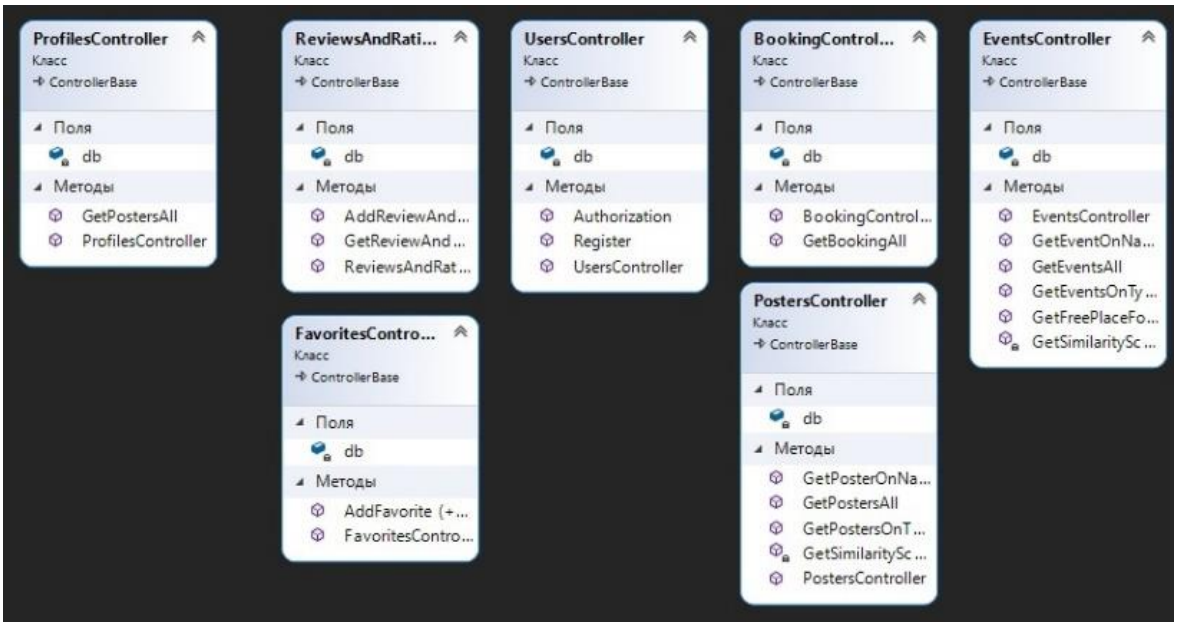


Рисунок 24 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 4.

Таблица 4 – Описание классов слоя контроллеров

Класс	Назначение
BookingController	Контроллер обрабатывает запросы по получению забронированных билетов.
EventsController	Обрабатывает запросы по получению списка мероприятий, а также мероприятий по категориям.
FavoritesController	Контроллер, который обрабатывает запросы по добавлению и получению избранных мероприятий.
PostersController	Контроллер обрабатывает запросы по получению информации о дате и времени проведения мероприятий.

Продолжение таблицы 4

Класс	Назначение
ReviewsAndRatingsController	Данный контроллер обрабатывает запрос на добавление и получение отзывов и оценок на мероприятие.
UsersController	Отвечает за обработку запросов, связанных с пользователями, таких как регистрация, авторизация.

Листинг серверной части представлен в приложении Б.

3.3 Реализация клиентской части приложения

Это мобильное приложение было создано с использованием языка программирования C# и интегрированной среды разработки Visual Studio 2019. Для разработки пользовательского интерфейса и общей логики приложения была использована библиотека Xamarin.Forms. Архитектура приложения была описана с помощью диаграммы компонентов, которая помогла визуализировать взаимодействие различных модулей и компонентов приложения. Диаграмма компонентов представлена на рисунке 25.

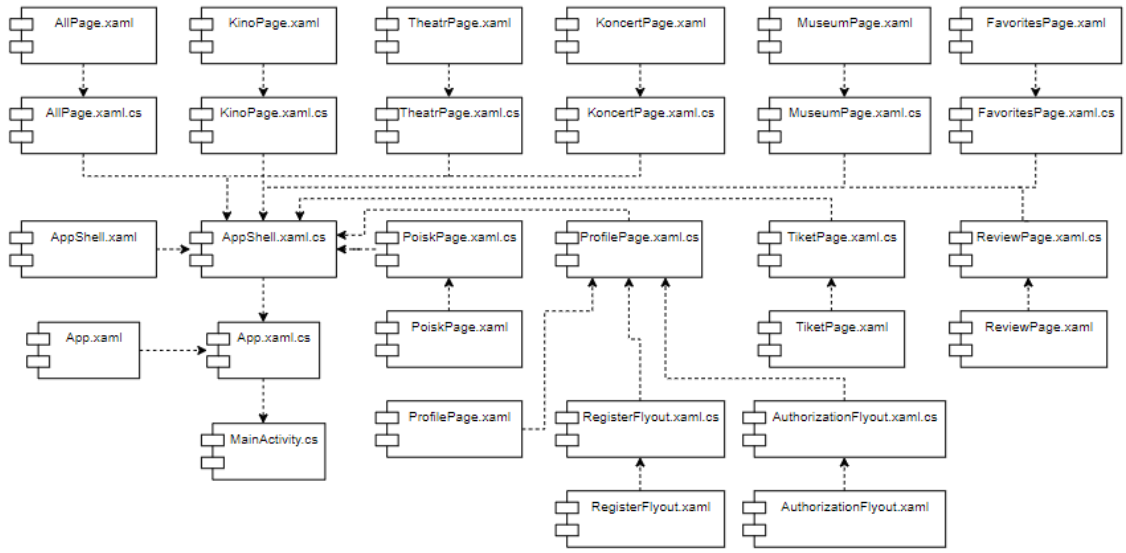


Рисунок 25 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 5.

Таблица 5 – Компоненты клиентского приложения

Компонент	Назначение
MainActivity.cs	Начальный класс, который отвечает за вход в систему
App.xaml.cs	Класс предназначен для отображения интерфейса в системе
App.xaml	Класс, который содержит ресурсы для интерфейса
AppShell.xaml.cs	Класс предназначен для отображения страниц в приложении
AppShell.xaml	Класс, который содержит ресурсы для страниц
AllPage.xaml.cs	Класс предназначен для отображения страницы всех мероприятий
AllPage.xaml	Реализует интерфейс страницы со всеми мероприятиями
KinoPage.xaml.cs	Класс предназначен для отображения страницы мероприятий с категорией кино
KinoPage.xaml	Реализует интерфейс страницы мероприятий с категорией кино
TheatrPage.xaml.cs	Класс предназначен для отображения страницы мероприятий с категорией театр
TheatrPage.xaml	Реализует интерфейс страницы мероприятий с категорией театр
KonsertPage.xaml.cs	Класс предназначен для отображения страницы мероприятий с категорией концерты
KonsertPage.xaml	Реализует интерфейс страницы мероприятий с категорией концерты
MuseumPage.xaml.cs	Класс предназначен для отображения страницы мероприятий с категорией музеи

Продолжение таблицы 5

Компонент	Назначение
MuseumPage.xaml	Реализует интерфейс страницы мероприятий с категорией музеи
PoiskPage.xaml.cs	Класс предназначен для отображения страницы поиска
PoiskPage.xaml	Реализует интерфейс страницы поиска
TicketPage.xaml.cs	Класс предназначен для отображения страницы с забронированными билетами пользователя
TicketPage.xaml	Реализует интерфейс страницы билетов
ProfilePage.xaml.cs	Класс предназначен для отображения страницы профиля пользователя
ProfilePage.xaml	Реализует интерфейс страницы профиля
FavoritesPage.xaml.cs	Класс предназначен для отображения страницы с мероприятиями которые пользователь добавил в избранное
FavoritesPage.xaml	Реализует интерфейс страницы избранного
ReviewPage.xaml.cs	Класс предназначен для отображения страницы с оставленными отзывами пользователя
ReviewPage.xaml	Реализует интерфейс страницы отзывов
RegisterFlyout.xaml.cs	Класс предназначен для отображения страницы регистрации пользователя
RegisterFlyout.xaml	Реализует интерфейс страницы регистрации
AuthorizationFlyout.xaml.cs	Класс предназначен для отображения страницы авторизации пользователя
AuthorizationFlyout.xaml	Реализует интерфейс страницы авторизации

На рисунке 26 представлена диаграмма развертывания, которая показывает, как расположены компоненты системы в физическом отношении.

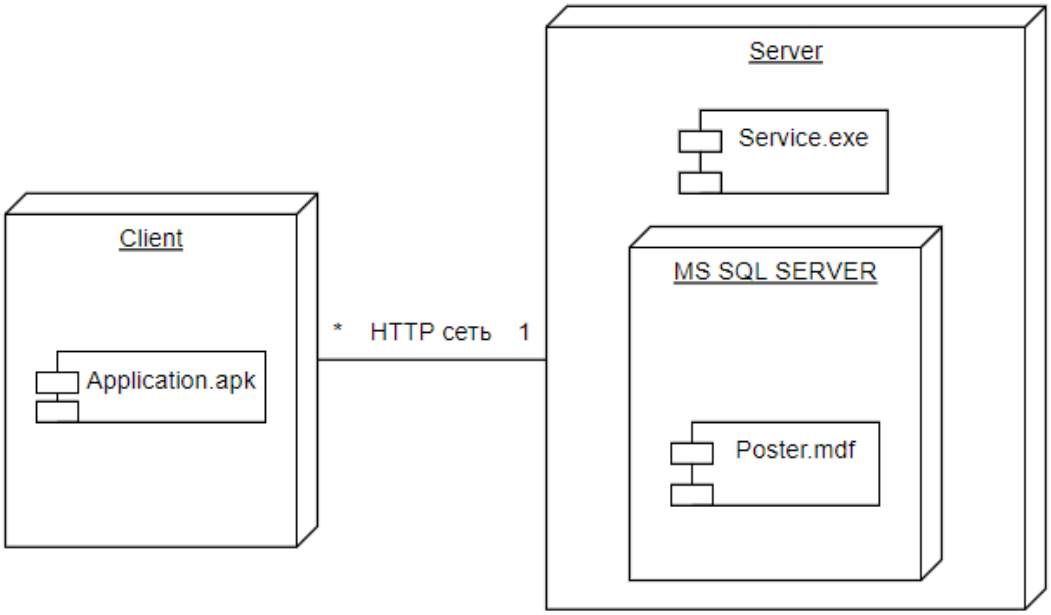


Рисунок 26 – Диаграмма развертывания мобильного приложения

На этой диаграмме развертывания показано, как клиентское приложение на устройстве с операционной системой Android взаимодействует с сервером, который работает на компьютере под управлением операционной системы Windows.

Серверная часть приложения состоит из службы обработки запросов и базы данных, реализованной на платформе Microsoft SQL Server. Сеть HTTP используется для облегчения связи между клиентом и сервером. Этот архитектурный дизайн позволяет пользователям удобно получать доступ к базе данных и обработке запросов, работая удаленно со своего мобильного устройства, используя сетевое подключение. Такая архитектура приложения придает ему большую гибкость и удобство в использовании, что способствует повышению удовлетворенности пользователей и улучшению качества обслуживания. Листинг клиентской части программы представлен в приложении В.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

В процессе тестирования приложения было проведено функциональное тестирование, которое включало проверку работоспособности основных функций приложения. Для тестирования серверной части использовалось приложение Postman. Кроме того, были разработаны тест-кейсы, в которых проверялась корректность обработки различных HTTP-запросов и взаимодействие с базой данных. Тесты для серверной части приложения приведены в таблице 6. Рисунки с отображением результатов тестирования приведены в приложении Г.

Таблица 6 – Результаты тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Регистрация нового пользователя с логином Test, паролем Test и псевдонимом Test	POST	Login = Test Password = Test Nickname = Test	Тест пройден. Пользователь зарегистрирован в системе. Код ответа HTTP – 200. Содержимое ответа – 5 (уникальный идентификатор пользователя в системе) (Рисунок Г.1).
2	Авторизация существующего пользователя с логином Test и паролем Test	POST	Login = Test Password = Test	Тест пройден. Пользователь зарегистрирован в системе. Код ответа HTTP – 200. Содержимое ответа – 1 (идентификатор пользователя в системе) (Рисунок Г.2).
3	Получение информации о всех мероприятиях	GET		Тест пройден. Мероприятия получены. Код ответа HTTP – 200. Содержимое ответа – json список информации по всем мероприятиям (Рисунок Г.3)
4	Получение информации о мероприятии с категорией кино	GET	TypeOfEvent = Кино	Тест пройден. Мероприятия получены. Код ответа HTTP – 200. Содержимое ответа – json список информации по мероприятиям с категорией кино (Рисунок Г.4)

Продолжение таблицы 6

№	Описание теста	Запрос		Результат
		Метод	Параметры	
5	Получение информации о мероприятии с названием слова, рожденные войной	GET	EventName = Слова, рожденные войной	Тест пройден. Мероприятия получены. Код ответа HTTP – 200. Содержимое ответа – json список информации о мероприятиям по названию (Рисунок Г.5)
6	Получение информации об афише мероприятия с категорией кино	GET	TypeOfPoster = Кино	Тест пройден. Афиша получена. Код ответа HTTP – 200. Содержимое ответа – json список информации об афише (Рисунок Г.6)
7	Получение информации о профиле пользователя с идентификатором IDUser=1	GET	IDUser = 1	Тест пройден. Профиль получен. Код ответа HTTP – 200. Содержимое ответа – json список информации об афише (Рисунок Г.7)
8	Добавления нового отзыва от пользователя с текстом отзыва, оценкой, EventId, ProfileId	POST	TextReview = Очень понрафилься фильм, Ratings = 5, EventId = 1, ProfileId = 1	Тест пройден. Комментарий добавлен к мероприятию. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.8)
9	Получение всех комментарий мероприятия с названием переводчик	GET	EventName = Переводчик	Тест пройден. Отзывы получены. Код ответа HTTP – 200. Содержимое ответа – json список отзывов (Рисунок Г.9)
10	Добавление мероприятия в избранное с EventId и ProfileIs	POST	EventId = 2, ProfileId = 1	Тест пройден. Мероприятие добавлен в избранное. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.10)
11	Получение избранных мероприятий с ID = 1	GET	ID = 1	Тест пройден. Избранное мероприятие получено. Код ответа HTTP – 200. Содержимое ответа – json список избранных мероприятий (Рисунок Г.11)
12	Забронировать билет	POST		Тест пройден. Избранное мероприятие получено. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.12)

Приведенные тесты выше были пройдены успешно, следовательно можно сделать вывод, что серверная часть работает корректно.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование

Для проверки клиентской части приложения будет проведено функциональное тестирование. Тестирование должно быть полным, затрагивая все функциональные возможности клиентского приложения. В таблице 7 приведены варианты тестов. Результат тестов приведен в приложении Д.

Таблица 7 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести данные для регистрации правильные. Логин: NewTest Пароль: NewTest Псевдоним: NewTest	Тест пройден. Регистрация прошла успешно. (Рисунок Д.1)
	Использовать в регистрации уже созданный логин. Логин: Test Пароль: Test Псевдоним: Test	Тест пройден. Система вывела сообщение о недопустимости данного логина. (Рисунок Д.2)
	Использовать в регистрации пароль меньше 4-х символов Логин: LoginTest Пароль: pas Псевдоним: Иван	Тест пройден. Система вывела сообщение об ошибке введенных данных. (Рисунок Д.3)
	Использовать запрещенные символы при вводе данных Логин: LoginTest!? Пароль: $Pass% Псевдоним: Дмитрий	Тест пройден. Система вывела сообщение об ошибке введенных данных. (Рисунок Д.4)
Авторизация	Ввести корректные данные для авторизации. Логин: Test Пароль: Test	Тест пройден. Авторизация прошла успешно, пользователь получил доступ к приложению. (Рисунок Д.5)
	Авторизировать под несуществующими данными. Логин: DoesntExistLogin Пароль: DoesntExistPassword	Тест пройден. Выведено сообщение о невозможности авторизации. (Рисунок Д.6)
Показать все мероприятие	Вывести список всех мероприятий	Тест пройден. Выведен список всех мероприятий (Рисунок Д.7)
Показать мероприятие по категории	Вывести список всех мероприятий с категорией кино	Тест пройден. Выведен список всех мероприятий с категорией кино. (Рисунок Д.8)

Продолжение таблицы 7

Вариант использования	Тест и параметры	Результат
Оставить отзыв	Написать комментарий и поставить оценку Тест комментария: очень красивый музей Оценка: 5	Тест пройден. Отзыв оставлен под новостью. (Рисунок Д.9)
Добавить в избранное	Добавить мероприятие в избранное	Тест пройден. Мероприятие добавлено в избранное. (Рисунок Д.10)
Изменить изображение в профили	Установить новое изображение для профиля из локального хранилища	Тест пройден. Система установила новое изображение для профиля пользователя. (Рисунок Д.11)
Изменить описание в профиле	Установить новое описание в профили в профили	Тест пройден. Система изменила текстовое описание профиля пользователя. (Рисунок Д.12)
Показать список свободных мест	Открыть расположение мест	Тест пройден. Система отобразила список свободных мест. (Рисунок Д.13)
Забронировать билет	Забронировать билет на мероприятие	Тест пройден. Место забронировано. (Рисунок Д.14)

Функциональное тестирование показало корректность работы клиентской части.

4.2.2 Тестирование приложения методом черного ящика

Метод тестирования черным ящиком заключается в проверке функциональности приложения без доступа к исходному коду. Такое тестирование позволяет быстро выявлять проблемы в работе приложения, однако не позволяет определить их источник.

Для примера будут приведены тесты для метода по созданию логина при регистрации. Табличное представление классов эквивалентности приведены в таблице 8.

Таблица 8 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина имени	4-32 1)	0-3 (2) 33- ∞ (3)

Продолжение таблицы 8

	Правильные КЭ	Неправильные КЭ
Структура	Начинается с буквы (4)	Начинается не с буквы (5)
Содержание	Буквы Латинские (6) Цифры (7) Дефис (8) Хотя бы одна большая латинская буква (10)	Остальные символы (9) Нет большой буквы (11)

Графическое представление классов эквивалентности представлено на рисунке 27.

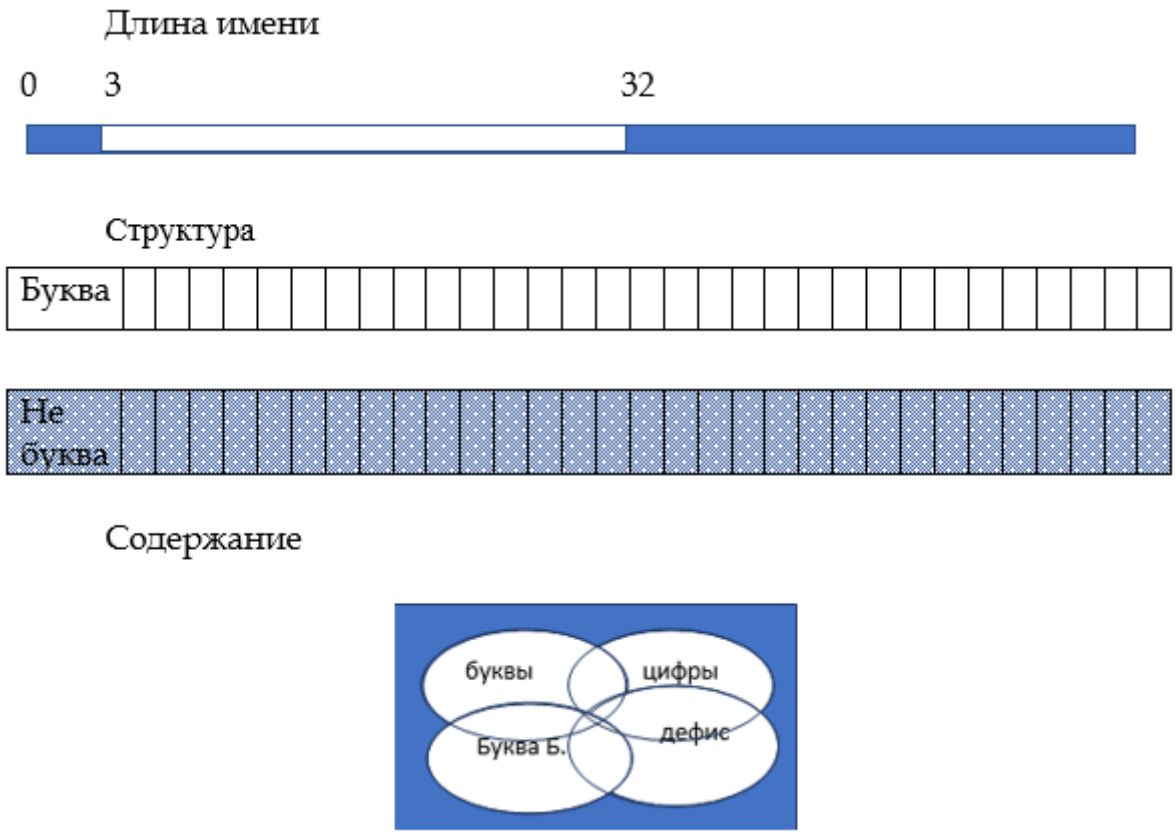


Рисунок 27 – Графическое представление классов эквивалентности

После определения структуры для логина необходимо привести примеры, которые будут содержать уникальную последовательность классов эквивалентности. Данные примеры в полной мере будут покрывать метод по созданию логина. Примеры приведены в таблице 9.

Таблица 9 – Тест–кейсы для создания логина

№	КЭ	Входные данные	Выходные данные
1	Все правильные 1,4,6,7,8,10	TestLogin-23	Успешно
2	**2**,4,6,7,8,10	P-1	Ошибка: меньше 4 символов
3	**3**,4,6,7,8,10	TheBestLoginNumberOneTheBestLoginNumberOneTheBestLoginNumberOne	Ошибка: больше 32 символов
4	1,**5**,6,7,8,10	-Dark-Pro-1232	Ошибка: первый символ не является буквой
5	1,4,7,8,**9,**10	TheBestLogin!?	Ошибка: символ не входящий в список разрешенных
6	1,4,7,8,**11**	badlogin-2	Ошибка: нет большой буквы

Из таблицы видно, что примеры полностью удовлетворяют условиям для создания логина. Эти тест-кейсы могут быть использованы в будущем для разработки модульных тестов.

4.2.3 Модульное тестирование

Модульные тесты представляют собой инструмент, который помогает автоматизировать процесс тестирования программного обеспечения. Они играют важную роль в обеспечении непрерывного контроля за правильностью работы программного обеспечения. Основная цель модульных тестов заключается в получении и сравнении фактического результата с ожидаемым, используя предварительно определенные функции и данные.

В данной задаче были разработаны модульные тесты для проверки данных, используемых при регистрации, таких как логин и пароль. Эти тесты позволяют убедиться, что введенные пользователем данные соответствуют определенным требованиям и работают корректно. Например, модульные тесты могут

проверять, что логин содержит только допустимые символы и что пароль соответствует определенным критериям безопасности.

Написание модульных тестов для проверки данных при регистрации помогает обнаружить и исправить потенциальные проблемы заранее, еще до того, как программа будет запущена в реальной среде. Такой подход способствует улучшению качества программного обеспечения и повышению удовлетворенности пользователей.

Модульные тесты были реализованы в классе ValidationRegisterTests, код которого представлен в листинге 1.

Листинг 1 – Код модульных тестов

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using UnitTest;

namespace StorageOfTest
{
    [TestClass]
    public class ValidationRegisterTests
    {
        [TestMethod]
        public void RegisterCorrectDataLogin_admin_Password_admin()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "test", Password = "test" };

            bool result = validation.SuitForm(user);

            Assert.IsTrue(result);
        }
        [TestMethod]
        public void RegisterLogin_log_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "log", Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_123Login_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "123Test", Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void
RegisterLogin_TooMuchLoginSymbolsCountitdoesntSuitimsureaboutit_Password_passwor
d()
```

```
        {
            Validation validation = new Validation();
            User user = new User() { Login =
"TooMuchLoginSymbolsCountitdoesntSuitimsureaboutitplusseveralsymbols", Password
= "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_EmptyLogin_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "", Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_UseDash_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "Login-With-Dash", Password =
"password" };

            bool result = validation.SuitForm(user);

            Assert.IsTrue(result);
        }
        [TestMethod]
        public void RegisterLogin_IncorrectSymbol_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "MyLoginHave!?Incorrect symbols",
Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_OnlyNumbers_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "12345678909876", Password =
"password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);}}}
```

Результат, полученный при запуске модульных тестов представлен на рисунке 28.

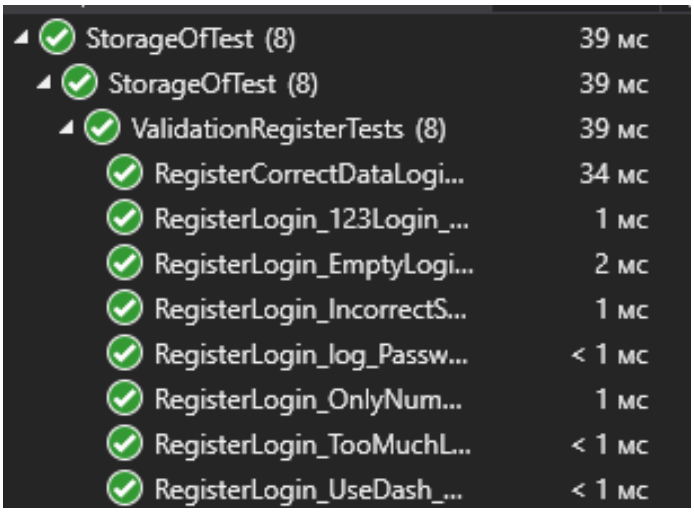


Рисунок 28 – Результаты модульного тестирования

Тесты, приведенные выше были успешно выполнены, что свидетельствует о корректной работоспособности проверяемого участка кода.

4.3 Метрики кода

Метрики кода представляют собой числовые значения, которые позволяют разработчику оценить различные аспекты программного обеспечения, такие как эффективность, оптимизация и масштабируемость написанного кода приложения.

Существует множество методов для получения метрик кода, которые различаются по алгоритмам и информации, полученной в результате. Для разработанного приложения будут использоваться следующие группы метрик:

– метрики размера программы (например, SLOC-метрики), которые измеряют объем кода, например, количество строк кода или операторов;

– Метрики сложности потока управления программы, такие как метрики Холстеда и цикломатическая сложность.

Эти метрики помогают разработчикам оценить характеристики кода, такие как его размер, сложность и структуру. Это позволяет выявить потенциальные проблемы и улучшить качество кода, а также облегчает его поддержку и дальнейшую разработку.

Для оценки возьмём метод Register класса UserController, который ответственен за регистрацию нового пользователя в системе. Код для метода регистрации приведен ниже в листинге 2.

Листинг 2 – Регистрация нового пользователя

```
public async Task<ActionResult> Register([FromBody] User user)
            {
                try
                {
                    //Проверяем корректность полученных данных
                    if (user == null || user.Login == null || user.Login == ""
|| user.Password == null || user.Password == "")
                        return BadRequest("Не все нужные данные были введены");
                    //Находим пользователя в базе данных по логину
                    var user = db.users.FirstOrDefault(x => x.Login ==
user.Login);
                    //Если пользователь существует с таким логинов, то заявляем,
что это ошибка
                    if (user != null)
                        return BadRequest("Пользователь с таким логином уже
есть");
                    //Создаём и добавляем нового пользователя
                    User userForAdded = new User { Login = user.Login, Pass =
user.Password, Nickname = user.Nickname};
                    db.users.Add(userForAdded);
                    db.SaveChanges();
                    //Создаём для нового пользователя профиль
                    Profile profile = new Profile { ID = userForAdded.ID,
Description = null, ImageProfile = null };
                    db.profiles.Add(profile);
                    db.SaveChanges();
                    return Ok(userForAdded.ID.ToString());}
                catch
                {
                    return BadRequest();}}
```

Метод, который обрабатывает попытки пользователей создать новую учетную запись в системе, является серверным. Он принимает логин, пароль и псевдоним в качестве входных данных. Задача метода состоит в проверке наличия уже существующей учетной записи с таким же логином.

Для оценки данного метода можно использовать две группы метрик: метрики размера программы и метрики Холстеда. Метрики размера программы позволяют оценить объем кода, например, количество строк или функций, что может быть полезно для определения сложности метода и его поддержки в будущем.

Метрики Холстеда, оценивает сложность программы на основе количества операторов и операндов, а также их взаимодействия. Они могут помочь в

определении сложности алгоритма, эффективности использования ресурсов и возможных проблем в коде.

Применение этих метрик позволит оценить качество и сложность данного метода, а также выявить потенциальные улучшения и проблемы в коде.

4.3.1 Метрики размера программы

В рамках первой проведенной метрики будет выполнено измерение размера программы, включающее подсчет количества строк программного кода и числа комментариев. Для этого будет использован код из листинга 2. Результаты измерений представлены в таблице 10.

Таблица 10 – Метрики размера программы

Метрика	Количество
Физических строк кода	24
Логических строк кода	15
Строки-комментарии	5

Предоставленные числовые значения позволяют вычислить процент комментариев для данного участка кода. Процент комментариев можно рассчитать, как (5/24) *100=20. Полученный результат указывает на достаточное количество комментариев в коде.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы

– число уникальных операторов;
– число уникальных операндов;
– общее число операторов;
– общее число операндов.

Для вычисления характеристик были составлены словарь операторов (таблица 11) и словарь операндов (таблица 12).

Таблица 11 – Словарь операторов

Оператор	Число операторов
Try catch	1
if	2
==	6
return	4
=	9
;	11
!=	1
new	2
{}	5
.	20
=>	1
()	12

Таблица 12 – Словарь операндов

Операнд	Число операндов
users	9
null	6
user	2
db	5
userForAdded	4
profile	2

Таким образом, число уникальных операторов программы $n1 = 12$, число уникальных операндов программы $n2 = 6$, общее число операторов в программе $N1 = 74$, общее число операндов в программе $N2 = 28$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

– словарь программы (n):

$$n = n1 + n2;$$

– длина программы(N):

$$N = N1 + N2;$$

– объем программы (V):

$$V = N * log_2 n;$$

– сложность программы (D):

$$D = (n1/2) * (N2/n2);$$

– оценка интеллектуальных усилий (E):

$$E = D * V.$$

Рассчитаем словарь программы: n = 12+6 = 18.

Рассчитаем длину программы N = 74+28=102.

Рассчитаем объем программы: V = 102 * $\log_2$18 = 425.

Рассчитаем сложность программы: D = (12/2) * (28/6) = 28.

Рассчитаем оценку интеллектуальных усилий: E=28 * 425 = 11900.

Подсчитанные значения предоставляют информацию о том, что в проекте используется небольшое количество переменных, что делает его более легким для поддержки в будущем. Кроме того, длина программы и ее объем указывают на то, что код является обширным, но при этом он разбит на логические функциональные блоки, которые легко масштабировать.

Сложность программы и оценка интеллектуальных усилий демонстрируют наличие множества высокоуровневых связей. Полученные характеристики указывают на то, что программный код обладает высоким уровнем качества и удобства для дальнейшей поддержки.

4.4 Планирование разработки и оценки бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project [5, 6, 7]. Проект был начат 12 апреля 2023 года.

При планировании разработки было выделено 29 задач. Список задач и их описание представлено на рисунке 29.

Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
Начало проекта	0 дней	Пн 03.04.23	Пн 03.04.23		
Анализ требований	**9 дней**	**Пн 03.04.23**	**Чт 13.04.23**		
Изучение предметной области	5 дней	Пн 03.04.23	Пт 07.04.23	1	Аналитик;Интернет;Компьютер[1]
Анализ аналогичных приложений	4 дней	Пн 10.04.23	Чт 13.04.23	3	Аналитик;Интернет;Компьютер[1]
Анализ требований завершен	0 дней	Чт 13.04.23	Чт 13.04.23	4	
Выбор средств	**5 дней**	**Пн 10.04.23**	**Пт 14.04.23**		
Выбор программных средств	3 дней	Пн 10.04.23	Ср 12.04.23	3	Программист ;Компьютер[1];Интернет
Выбор аппаратных средств	2 дней	Чт 13.04.23	Пт 14.04.23	7	Программист ;Интернет;Компьютер[1]
Завершение выбора средств	0 дней	Пт 14.04.23	Пт 14.04.23	8	
Проектирование приложение	**9 дней**	**Пн 17.04.23**	**Чт 27.04.23**		
Проектирование пользовательского интерфейса	3 дней	Пн 17.04.23	Ср 19.04.23	9	Программист ;Интернет;Компьютер[1]
Проектирование базы данных	2 дней	Чт 20.04.23	Пт 21.04.23	11	Программист ;Компьютер[1];Интернет
Проектирование бизнес-логики	2 дней	Чт 20.04.23	Пт 21.04.23	11	Программист ;Компьютер[1];Интернет
Проектирование клиентской части	1 день	Пн 24.04.23	Вт 25.04.23	13	Программист [50%];Компьютер[1];Инте
Проектирование серверной части	2 дней	Ср 26.04.23	Чт 27.04.23	14	Программист ;Компьютер[1];Интернет
Завершение проектирование приложения	0 дней	Чт 27.04.23	Чт 27.04.23	15	
Реализация приложения	**21 дней**	**Пт 28.04.23**	**Пт 26.05.23**		
Разработка пользовательского интерфейса	4 дней	Пт 28.04.23	Ср 03.05.23	16	Программист ;Компьютер[1];Мобильно
Разработка базы данных	4 дней	Чт 04.05.23	Вт 09.05.23	18	Программист [75%];Компьютер[1];Инте
Разработка бизнес-логики приложения	4 дней	Ср 10.05.23	Пн 15.05.23	19	Программист ;Аналитик;Компьютер[1];
Разработка клиенской части	5 дней	Вт 16.05.23	Пн 22.05.23	20	Программист ;Мобильно устройство[1]
Разработка серверной части	4 дней	Вт 23.05.23	Пт 26.05.23	21	Программист ;Компьютер[1];Интернет
Завершение реализации приложения	0 дней	Пт 26.05.23	Пт 26.05.23	22	
Тестирование	**4 дней**	**Пн 29.05.23**	**Чт 01.06.23**		
Модульное тестирвоание	2 дней	Пн 29.05.23	Вт 30.05.23	23	Программист ;Тестировщик;Компьютер
Функциональное тестирование	2 дней	Пн 29.05.23	Вт 30.05.23	23	Тестировщик;Мобильно устройство[1];
Нагрузочное тестирование	2 дней	Ср 31.05.23	Чт 01.06.23	25;26	Тестировщик;Интернет;Мобильно устр
Тестирование завершено	0 дней	Чт 01.06.23	Чт 01.06.23	27	
Конец проекта	0 дней	Чт 01.06.23	Чт 01.06.23	28	

Рисунок 29 – Список задач проекта

Для реализации проекта необходимо выделить основные ресурсы, их типа и стоимость. Описание ресурсов представлено на рисунке 30.

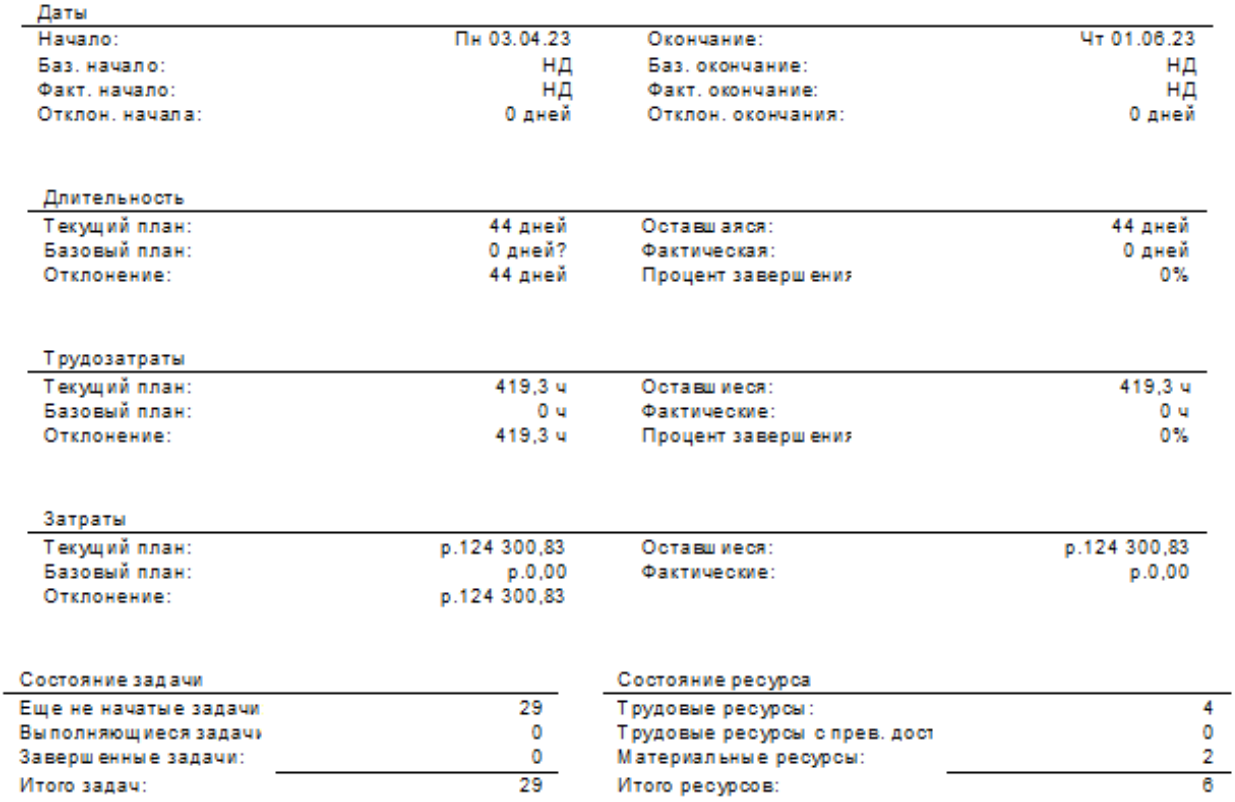
Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь	Код
Программист	Трудовой		П		100%	р.50 000,00/мес	р.300,00/ч	р.0,00	Пропорциональное	Стандартный	
Аналитик	Трудовой		А		100%	р.45 000,00/мес	р.250,00/ч	р.0,00	Пропорциональное	Стандартный	
Тестировщик	Трудовой		Т		100%	р.35 000,00/мес	р.200,00/ч	р.0,00	Пропорциональное	Стандартный	
Компьютер	Материальный		К			р.0,00		р.0,00	Пропорциональное		
Мобильно устройство	Материальный		М			р.0,00		р.0,00	Пропорциональное		
Интернет	Затраты		И						Пропорциональное		

Рисунок 30 – Список ресурсов проекта

В результате выделения задач и ресурсов была построена диаграмма Ганта, изображена на рисунке 31. К ней представлены продолжительность задач и взаимосвязи их друг с другом

Рисунок 31 – Диаграмма Ганта

На рисунке 32 изображен обзор проекта. Из графика можно увидеть, что продолжительность проекта составляет 44 дня, а предполагаемые затраты – 124 300 рублей.



Даты

Начало:	Пн 03.04.23	Окончание:	Чт 01.06.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность

Текущий план:	44 дней	Оставшаяся:	44 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	44 дней	Процент завершения	0%

Трудозатраты

Текущий план:	419,3 ч	Оставшиеся:	419,3 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	419,3 ч	Процент завершения	0%

Затраты

Текущий план:	р.124 300,83	Оставшиеся:	р.124 300,83
Базовый план:	р.0,00	Фактические:	р.0,00
Отклонение:	р.124 300,83		

Состояние задачи

Еще не начатые задачи	29
Выполняющиеся задачи	0
Завершенные задачи:	0
Итого задач:	29

Состояние ресурса

Трудовые ресурсы:	4
Трудовые ресурсы с прев. дост	0
Материальные ресурсы:	2
Итого ресурсов:	6

Рисунок 32 – Обзорный отчет проекта

На основании проведенной оценки затрат и временных рамок проекта можно сделать вывод о его экономической целесообразности.

Заключение

В результате выполнения выпускной квалификационной работы разработано мобильное приложение для городских событий.

При анализе требований и предметной области были проанализированы программные продукты, выполняющие аналогичные функции, с целью выявления их преимуществ и недостатков. Данный анализ позволил определить функциональные и нефункциональные требования к создаваемому приложению.

Произведено составление плана разработки приложения и проведен расчет бюджета. Из результатов планирования следует, что проект займет 44 дня на реализацию, а расчетные затраты составят 124 300 рублей, что говорит о том, что программа является экономически выгодным проектом.

В процессе проектирования приложения была разработана логическая модель базы данных и создана архитектура мобильного приложения. База данных была создана при помощи SQL, а для ее управления была выбрана СУБД Microsoft SQL Server 2018. Серверная часть приложения была создана на языке программирования C# с использованием среды разработки Microsoft Visual Studio 2019. Для клиентской части был выбран язык программирования C#, а среда разработки – Visual Studio 2019. Для разработки клиентской части была использована библиотека Xamarin Forms.

После завершения разработки приложения было проведено тестирование функциональности как серверной, так и клиентской частей, которое показало, что они работают корректно. Для оценки качества программного кода были рассчитаны соответствующие метрики.

При выполнении выпускной квалификационной работы были успешно выполнены все поставленные задачи, что позволило достичь поставленных целей и получить желаемый результат. Результатом работы стало разработанное мобильное приложение для городских событий.

Список использованных источников

1. Об «Афише» [Электронный ресурс]. – URL: https://www.afisha.ru/article/about/ (дата обращения: 15.05.2023).

2. О сайте «Киноафиша» [Электронный ресурс]. – URL: https://www.kinoafisha.info/about/ (дата обращения: 15.05.2023).

3. О проекте [Электронный ресурс]. – URL: https://afisha7.ru/page/about/ (дата обращения: 15.05.2023).

4. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт ; пер. с англ. – 8-е изд. – Москва: Издательский дом «Вильяме», 2005. – 1328 с.

5. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.

6. Джонсон, Т. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.

7. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.

8. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб.: Питер, 2007. – 544 с.

9. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт ; пер. с англ. – 8-е изд. – Москва: Издательский дом «Вильяме», 2005. – 1328 с.

10. Учебник. Создание веб-API с помощью ASP.NET Core // Microsoft Learn. [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/ASPNET/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio (дата обращения: 26.05.2023)

11. Руководство по ASP.NET Web API 2 // METANIT.COM. [Электронный ресурс]. URL: https://metanit.com/sharp/aspnet_webapi/ (дата обращения: 26.05.2023)

12. Смит Дж. П. Entity Framework Core в действии / пер. с англ. Д. А. Беликова. – Москва: ДМК Пресс, 2022. – 690 с.

Приложение А

ИСХОДНЫЙ КОД СКРИПТОВ БАЗЫ ДАННЫХ

(рекомендуемое)

```
CREATE DATABASE Poster

create table Useres(
IdUs int IDENTITY not null,
LoginUs varchar(32) not null UNIQUE,
PassUs varchar(20) not null UNIQUE,
NicknameUs varchar(20) not null,
primary key (IdUs));

create table Profiles (
UseresId INT primary key not null,
Forigen key(UsersId) REFERENCES Useres(IdUs),
DescriptionProf varchar(1000) null,
Image varchar(100) null);

create table EventTypes (
IdEvT int IDENTITY not null,
NameEvT varchar(100) not null,
primary key (IdEvT));

create table Events (
IdEv int IDENTITY not null,
NameEv varchar(1000) not null,
DescriptionEv varchar(1000) not null,
GenreEv varchar(255) null,
ImageEv varchar(1000) null,
EventTypesId int REFERENCES EventTypes (IdEvT),
primary key (IdEv));

create table Poster(
IdPos int IDENTITY not null,
DatePos date not null,
TimePos time not null,
EventsId INT REFERENCES Event (IdEv),
primary key (IdPos));

create table Favorites(
IdFav int IDENTITY NOT NULL,
EventsIdFav int REFERENCES Events (IdEv),
ProfilesIdFav int REFERENCES Profiles (UseresId),
primary key (IdFav));

create table ReviewsAndRatinges (
IdReview int IDENTITY not null,
TextReview varchar (1000) not null,
Ratinges int NOT NULL,
EventsIdrev int REFERENCES Event (IdEv),
ProfilesIdRev int REFERENCES Profiles (UsersId),
primary key (IdReview));

create table Location(
IdLocation int IDENTITY not null,
Location varchar(255) not null,
primary key (IdLocation));

create table Place(
IdPlace int IDENTITY not null,
RowPlace int not null,
Place int not null,
LocationIdPlace int REFERENCES Location (Id),
primary key (IdPlace));

create table Booking(
```

```
IdBooking int IDENTITY NOT NULL,
UsersIdBooking int REFERENCES Users (Id),
PosterIdBooking int REFERENCES Poster (Id),
PlaceIdBooking int REFERENCES Place (Id),
primary key (IdBooking));
```

Приложение Б

ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ

(рекомедуемое)

Листинг Б.1 – Файл ApplicationsContexts.cs

```
namespace PostServer.Model
{
    public class ApplicationsContexts : DbContext
    {
        public DbSet<Bookings> Bookings { get; set; }
        public DbSet<Events> Events { get; set; }
        public DbSet<EventsTypes> EventTypes { get; set; }
        public DbSet<Favorites> Favorites { get; set; }
        public DbSet<Positions> Positions { get; set; }
        public DbSet<Places> Places { get; set; }
        public DbSet<Posters> Posters { get; set; }
        public DbSet<Profiles> Profiles { get; set; }
        public DbSet<ReviewsAndRatings> ReviewsAndRatings { get; set; }
        public DbSet<Useres> Users { get; set; }
        public ApplicationsContexts(DbContextOptions<ApplicationsContexts>
optionsBuilder) : base(optionsBuilder)
        {
            Update();
        }
        public void Update()
        {
            Useres.Load();
            Profiles.Load();
            EventsTypes.Load();
            Events.Load();
            Posters.Load();
            Favorites.Load();
            ReviewsAndRatings.Load();
            Positions.Load();
            Places.Load();
            Bookings.Load();
        }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=DESKTOP-
J9N17RD\SQLEXPRESS;Initial Catalog=Poster;User Id=Иван; Integrated
Security=True; TrustServerCertificate=True;
MultipleActiveResultSets=True;Trusted_Connection=True;");
        }
    }
}
```

Листинг Б.2 – Файл Bookings.cs

```
namespace PostServer.Model
{
    [Table("Bookings")]
    public class Booking
    {
        [Column("Id")]
        public int Id { get; set; }
        [Column("UsersId"), ForeignKey("CurrentUser")]
        public int UsersId { get; set; }
        public virtual User CurrentUser { get; set; }
        [Column("PosterId"), ForeignKey("CurrentPoster")]
        public int PosterId { get; set; }
        public virtual Poster CurrentPoster { get; set; }
```

```
        [Column("PlaceId"), ForeignKey("CurrentPlace")]
        public int PlaceId { get; set; }
        public virtual Place CurrentPlace { get; set; }
    }
}
```

Листинг Б.3 – Файл Events.cs

```
namespace PostServer.Model
{
    [Table("Events")]
    public class Events
    {
        [Column("IdEv")]
        public int Id { get; set; }
        [Column("NameEv")]
        public string Name { get; set; }
        [Column("DescriptionEv")]
        public string Description { get; set; }
        [Column("GenreEv")]
        public string? Genre { get; set; }
        [Column("ImageEv")]
        public string? Image { get; set; }
        [Column("EventsTypesId"), ForeignKey("CurrentEventType")]
        public int EventsTypesId { get; set; }
        public virtual EventType CurrentEventType { get; set; }
    }
}
```

Листинг Б.4 – Файл EventsTypes.cs

```
namespace PostServer.Model
{
    [Table("EventsTypes")]
    public class EventType
    {
        [Column("IdEvT")]
        public int Id { get; set; }
        [Column("NameEvT")]
        public string Name { get; set; }

    }
}
```

Листинг Б.5 – Файл Favorites.cs

```
namespace PostServer.Model
{
    [Table("Favorites")]
    public class Favorites
    {
        [Column("IdFav")]
        public int Id { get; set; }
        [Column("EventIdFav"), ForeignKey("CurrentEvent")]
        public int EventId { get; set; }
        public virtual Event CurrentEvent { get; set; }
        [Column("ProfileIdFav"), ForeignKey("CurrentProfile")]
        public int ProfileId { get; set; }
        public virtual Profile CurrentProfile { get; set; }
```

```
    }
}
```

Листинг Б.6 – Файл Locations.cs

```
namespace PostServer.Model
{
    [Table("Locations")]
    public class Position
    {
        [Column("IdLocations")]
        public int Id { get; set; }
        [Column("Location")]
        public string Location { get; set; }

    }
}
```

Листинг Б.7 – Файл Places.cs

```
namespace PostServer.Model
{
    [Table("Places")]
    public class Place
    {
        [Column("IdPlaces ")]
        public int Id { get; set; }
        [Column("RowPlaces ")]
        public int Row { get; set; }
        [Column("Place")]
        public int PlaceNumber { get; set; }
        [Column("LocationId"), ForeignKey("CurrentLocation")]
        public int LocationId { get; set; }
        public virtual Position CurrentLocation { get; set; }
    }
}
```

Листинг Б.8 – Файл Posters.cs

```
namespace PostServer.Model
{
    [Table("Posters")]
    public class Poster
    {
        [Column("IdPos")]
        public int Id { get; set; }
        [Column("DatePos")]
        public DateTime? DatePos { get; set; }
        [Column("TimePos")]
        public TimeSpan? TimePos { get; set; }
        [Column("EventsId"), ForeignKey("CurrentEvent")]
        public int EventsId { get; set; }
        public virtual Event CurrentEvent { get; set; }
    }
}
```

Листинг Б.9 – Файл Profiles.cs

```
namespace PostServer.Model
{
    [Table("Profiles")]
    public class Profile
    {
        [Column("UsersId")]
        public int Id { get; set; }
        [Column("DescriptionProf")]
        public string Description { get; set; }
        [Column("ImageProf")]
        public string Image { get; set; }
    }
}
```

Листинг Б.10 – Файл ReviewesAndRatinges.cs

```
namespace PostServer.Model
{
    [Table("ReviewesAndRatinges")]
    public class ReviewsAndRatings
    {
        [Column("IdReview")]
        public int Id { get; set; }
        [Column("TextReview")]
        public string TextReview { get; set; }
        [Column("Ratings")]
        public int Ratings { get; set; }
        [Column("EventIdRev"), ForeignKey("CurrentEvent")]
        public int EventId { get; set; }
        public virtual Event CurrentEvent { get; set; }
        [Column("ProfileIdRev"), ForeignKey("CurrentProfile")]
        public int ProfileId { get; set; }
        public virtual Profile CurrentProfile { get; set; }
    }
}
```

Листинг Б.11 – Файл Useres.cs

```
namespace PostServer.Model
{
    [Table("Useres")]
    public class Users
    {
        [Column("IdUs")]
        public int Id { get; set; }
        [Column("LoginUs")]
        public string Login { get; set; }
        [Column("PassUs")]
        public string Pass { get; set; }
        [Column("NicknameUs")]
        public string Nickname { get; set; }
    }
}
```

Листинг Б.12 – Файл BookingsControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
```

```
    public class BookingsControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public BookingsControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetBookingsAll")]
        public async Task<ActionResult> GetBookingsAll([FromQuery(Name =
"NameEvent")] string NameEvent, [FromQuery(Name = "CurrentDateTime")] DateTime
CurrentDateTime)
        {
            try
            {
                var Bookings =
db.Bookings.Where(x=>x.CurrentPoster.CurrentEvent.Name==NameEvent &&
x.CurrentPoster.DatePos==CurrentDateTime).ToList();
                var JsonString = JsonConvert.SerializeObject(Bookings);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpPost]
        [Route("Buy")]
        public async Task<ActionResult> Buy([FromBody] Booking booking)
        {
            try
            {
                if (db.Bookings.FirstOrDefault(x => x.PosterId ==
booking.PosterId && x.CurrentPlace.LocationId == booking.CurrentPlace.LocationId
&&
                 x.CurrentPlace.PlaceNumber == booking.CurrentPlace.PlaceNumber
&& x.CurrentPlace.Row == booking.CurrentPlace.Row) != null)
                    return BadRequest("Данный билет уже куплен на данное
мероприятие");

                db.Bookings.Add(booking);
                db.SaveChanges();

                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }
}
```

Листинг Б.13 – Файл EventsControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
    public class EventsControllers: ControllerBase
    {
        private readonly ApplicationsContexts db;
```

```
        public EventsControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetEventsAll")]
        public async Task<ActionResult> GetEventsAll()
        {
            try
            {
                var Events = db.Events.ToList();
                var JsonString = JsonConvert.SerializeObject(Events);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetEventsOnType")]
        public async Task<ActionResult> GetEventsOnType([FromQuery(Name =
"TypeOfEvent")] string TypeOfEvent)
        {
            try
            {
                var Events = db.Events.Where(x =>
x.CurrentEventType.Name.ToLower() == TypeOfEvent.ToLower()).ToList();
                var JsonString = JsonConvert.SerializeObject(Events);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetEventOnName")]
        public async Task<ActionResult> GetEventOnName([FromQuery(Name =
"NameEvent")] string NameEvent)
        {
            try
            {
                var Events = db.Events.OrderByDescending(x => x.Name).ToList();
                Events = Events.OrderByDescending(x =>
GetSimilarityScore(x.Name, NameEvent)).Take(30).ToList();
                var JsonString = JsonConvert.SerializeObject(Events);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        static int GetSimilarityScore(string str1, string str2)
        {
            if (str1.ToLower() == str2.ToLower())
                return str2.Length * 10;
            var count = str1.Count(ch => str2.Contains(ch));
            return count;
        }

        [HttpGet]
```

```
        [Route("GetFreePlaceForEvents")]
        public async Task<ActionResult> GetFreePlaceForEvents([FromQuery(Name =
"NameEvent")] string NameEvent, [FromQuery(Name = "CurrentDateTime")] DateTime
CurrentDateTime)
        {
            try
            {
                int CountFreePlace = 100 - db.Bookings.
                    Where(x=>x.CurrentPoster.DatePos==CurrentDateTime &&
x.CurrentPoster.CurrentEvent.Name==NameEvent).
                    Count();
                var JsonString = JsonConvert.SerializeObject(CountFreePlace);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }
}
```

Листинг Б.14 – Файл FavoritesControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
    public class FavoritesControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public FavoritesControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpPost]
        [Route("AddFavorites")]
        public async Task<ActionResult> AddFavorites([FromBody] Favorites
Favorite)
        {
            try
            {
                if (db.Users.FirstOrDefault(x => x.Id == Favorites.ProfileId) ==
null)
                    return BadRequest();

                db.Favorites.Add(Favorite);
                db.SaveChanges();

                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetFavorites")]
        public async Task<ActionResult> AddFavorite([FromQuery(Name = "ID")] int
ID)
        {
            try
            {
```

```
                var Favorities = db.Favorites.Where(x => x.CurrentProfile.Id ==
ID).ToList();
                var JsonString = JsonConvert.SerializeObject(Favorities);

                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }
}
```

Листинг Б.15 – Файл PostersControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
    public class PostersControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public PostersControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetPostersAll")]
        public async Task<ActionResult> GetPostersAll()
        {
            try
            {
                var Posters = db.Posters.ToList();
                var JsonString = JsonConvert.SerializeObject(Posters);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetPostersOnTypes")]
        public async Task<ActionResult> GetPostersOnTypes([FromQuery(Name =
"TypeOfPoster")] string TypeOfPoster)
        {
            try
            {
                var Posters = db.Posters.Where(x =>
x.CurrentEvent.CurrentEventType.Name.ToLower() ==
TypeOfPoster.ToLower()).ToList();
                var JsonString = JsonConvert.SerializeObject(Posters);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetPosterOnName")]
```

```
        public async Task<ActionResult> GetPosterOnNames([FromQuery(Name =
"NameEvent")] string NamePoster)
        {
            try
            {
                var Posters = db.Events.OrderByDescending(x => x.Name).ToList();
                Posters = Posters.OrderByDescending(x =>
GetSimilarityScore(x.Name, NamePoster)).Take(30).ToList();
                var JsonString = JsonConvert.SerializeObject(Posters);
                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        static int GetSimilarityScore(string str1, string str2)
        {
            if (str1.ToLower() == str2.ToLower())
                return str2.Length * 10;
            var count = str1.Count(ch => str2.Contains(ch));
            return count;
        }

    }
}
```

Листинг Б.16 – Файл ProfilesControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
    public class ProfilesControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public ProfilesControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetProfiles")]
        public async Task<ActionResult> GetProfiles([FromQuery(Name = "IDUser")]
int IDUser)
        {
            try
            {
                var profile = db.Profiles.FirstOrDefault(x => x.Id == IDUser);
                if (profile == null)
                    return BadRequest();
                return Ok(profile);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }
}
```

Листинг Б.17 – Файл ReviewsAndRatingsControllers.cs

```
namespace PostServer.Controllers
{

    [ApiController]
    public class ReviewsAndRatingsControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public ReviewsAndRatingssControllers(ApplicationsContexts context)
        {
            db = context;
        }
        [HttpPost]
        [Route("AddReviewAndRatings")]
        public async Task<ActionResult> AddReviewAndRatings([FromBody]
ReviewsAndRatings ReviewAndRating)
        {
            try
            {
                if (db.Users.FirstOrDefault(x => x.Id ==
ReviewAndRating.ProfileId) == null)
                    return BadRequest();

                db.ReviewsAndRatings.Add(ReviewAndRating);
                db.SaveChanges();

                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
        [HttpGet]
        [Route("GetReviewAndRatings")]
        public async Task<ActionResult> GetReviewAndRatings([FromQuery(Name =
"NameEvent")] string NameEvent)
        {
            try
            {
                var ReviewsAndRatings = db.ReviewsAndRatings.Where(x =>
x.CurrentEvent.Name== NameEvent).ToList();
                var JsonString = JsonConvert.SerializeObject(ReviewsAndRatings);

                return Ok(JsonString);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }
}
```

Листинг Б.18 – Файл UsersControllers.cs

```
namespace PostServer.Controllers
{
    [ApiController]
    public class UsersControllers : ControllerBase
    {
        private readonly ApplicationsContexts db;
        public UsersControllers(ApplicationsContexts context)
```

```
        {
            db = context;
        }
[HttpPost]
[Route("Registering")]
public async Task<ActionResult> Registering([FromBody] User users)
         {
           try
                {
                    if (users == null || users.Login == null || users.Login ==
"" || users.Password == null || users.Password == "")
                        return BadRequest("Не все нужные данные были введены");
                    var user = db.Users.FirstOrDefault(x => x.Login ==
users.Login);
                    if (user != null)
                        return BadRequest("Пользователь с таким логином уже
есть");
                    User userForAdded = new User { Login = users.Login, Pass =
users.Password, Nickname = users.Nickname};
                    db.Users.Add(userForAdded);
                    db.SaveChanges();
                    Profile profile = new Profile { ID = userForAdded.ID,
Description = null, ImageProfile = null };
                    db.Profiles.Add(profile);
                    db.SaveChanges();
                    return Ok(userForAdded.ID.ToString());}
                catch
                {
                    return BadRequest();}}


        [HttpPost]
        [Route("Authorizationes")]
        public async Task<ActionResult> Authorizationes([FromBody] User user)
        {
            try
            {
                if (db.Users.FirstOrDefault(x => x.Login == user.Login && x.Pass
== user.Pass) != null)
                    return Ok();

                return BadRequest(user.Id);
            }
            catch (Exception ex)
            {
                return BadRequest();
            }
        }
    }}
```

Приложение В

ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ

(рекомендуемое)

Листинг В.1 – Файл MainActivity.cs

```
namespace Poster.Droid
{
    [Activity(Label = "Афиша", Icon = "@mipmap/ic_launcher", Theme =
"@style/MainTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode |
ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize )]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            LoadApplication(new App());
        }
        protected override void OnActivityResult(int requestCode, Result
resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);
            if ((resultCode == Result.Ok) && (data != null))
            {
                Android.Net.Uri uri = data.Data;
                System.IO.Stream Mystream =
this.ContentResolver.OpenInputStream(uri);

                ExifInterface exif = new ExifInterface(Mystream);
                var orientation =
exif.GetAttributeInt(ExifInterface.TagOrientation,
(int)Android.Media.Orientation.Normal);
                Bitmap bitmap =
MediaStore.Images.Media.GetBitmap(this.ContentResolver, data.Data);
                Matrix matrix = new Matrix();
                switch (orientation)
                {
                    case 6:
                        matrix.PostRotate(90);
                        bitmap = Bitmap.CreateBitmap(bitmap, 0, 0, bitmap.Width,
bitmap.Height, matrix, true);
                        break;
                    case 8:
                        matrix.PostRotate(270);
                        bitmap = Bitmap.CreateBitmap(bitmap, 0, 0, bitmap.Width,
bitmap.Height, matrix, true);
                        break;
                    case 3:
                        matrix.PostRotate(180);
                        bitmap = Bitmap.CreateBitmap(bitmap, 0, 0, bitmap.Width,
bitmap.Height, matrix, true);
                        break;
                    default:
                        break;
                }

                var imgsrc = ImageSource.FromStream(() =>
                {
                    MemoryStream ms = new MemoryStream();
                    bitmap.Compress(Bitmap.CompressFormat.Jpeg, 100, ms);
```

```
                    bitmap.Recycle();
                    ms.Seek(0L, SeekOrigin.Begin);
                    return ms;
                });

                if (requestCode == 1)
                    Profile.currentObject.UpdatePhoto(imgsrc);
            }
        }
        public override void OnRequestPermissionsResult(int requestCode,
string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[]
grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

            base.OnRequestPermissionsResult(requestCode, permissions,
grantResults);
        }
    }
}
```

Листинг В.2 – Файл App.xaml.cs

```
namespace Poster
{
    public partial class App : Xamarin.Forms.Application
    {
        internal static Context ActivityContexts { get; private set; }
        internal static Activity ActivityCurrents { get; private set; }
        internal static int IDCurrentUser { get; set; }
        public App(Context context, Activity activity)
        {
            InitializeComponent();
            ActivityContexts = context;
            ActivityCurrents = activity;
            MainPage = new NavigationPage(new Authorization());
            string login = Preferences.Get("Login", string.Empty);
            string password = Preferences.Get("Password", string.Empty);
            if(login!=string.Empty && password!=string.Empty)
            {
                ModelView.ModelViewAuthorization modelView = new
ModelView.ModelViewAuthorization(AppShell);
                modelView.TextLogins = login;
                modelView.TextPasswords = password;
                modelView.ExecuteProcedure();
            }
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
```

```
}
```

Листинг В.3 – Файл App.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

Листинг В.4 – Файл AppShell.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AppShell : Shell
    {

        public AppShell()
        {
            InitializeComponent();
        }

    }
}
```

Листинг В.5 – Файл AppShell.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:poisk="clr-namespace:Poster"
             xmlns:all="clr-namespace:Poster"
             xmlns:kino="clr-namespace:Poster"
             xmlns:theatr="clr-namespace:Poster"
             xmlns:koncert="clr-namespace:Poster"
             xmlns:museum="clr-namespace:Poster"
             xmlns:profile="clr-namespace:Poster"
             xmlns:ticket="clr-namespace:Poster"
             x:Class="Poster.AppShell"
       Shell.BackgroundColor="#30302e" Shell.TabBarTitleColor="White">

    <TabBar >
        <Tab Title="Афиша" Icon="playbill.png">
            <ShellContent Title="Все" ContentTemplate="{DataTemplate
all:AllPage}"/>

            <ShellContent Title="Кино" ContentTemplate="{DataTemplate
kino:KinoPage}"/>

            <ShellContent Title="Театр" ContentTemplate="{DataTemplate
theatr:TheatrPage}"/>

            <ShellContent Title="Концерты" ContentTemplate="{DataTemplate
koncert:KoncertPage}"/>
```

```
            <ShellContent Title="Музеи" ContentTemplate="{DataTemplate
museum:MuseumPage}"/>

        </Tab>
        <Tab Title="Поиск" Icon="search.png">
            <ShellContent ContentTemplate="{DataTemplate poisk:PoiskPage}" />

        </Tab>
        <Tab Title="Профиль" Icon="user.png">
            <ShellContent Title="Профиль" ContentTemplate="{DataTemplate
profile:ProfilePage}"/>

            <ShellContent Title="Отзывы" ContentTemplate="{DataTemplate
kino:ReviewPage}"/>

            <ShellContent Title="Избранное" ContentTemplate="{DataTemplate
theatr:FavoritesPage}"/>

        </Tab>
        <Tab Title="Билеты" Icon="ticket.png">
            <ShellContent ContentTemplate="{DataTemplate ticket:TiketPage}" />
        </Tab>
    </TabBar>
</Shell>
```

Листинг В.6 – Файл AllPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AllPage : ContentPage
    {
        public AllPage()
        {
            InitializeComponent();
        }


         string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
         {
            PropertyNameCaseInsensitive = true,
         };
         private HttpClient GetClient()
        {
             HttpClient clients = new HttpClient();
             client.DefaultRequestHeaders.Add("Accept", "application/json");
             return clients;
        }

        public async Task<ActionResult> GetEventsAll()
         {
             url = "http://192.168.1.4:5002/GetEventsAll/";
             HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.7 – Файл AllPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.AllPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventesList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.8 – Файл KinoPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class KinoPage : ContentPage
    {
        public KinoPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetEventsOnTypes()
        {
            url = "http://192.168.1.4:5002/GetEventsOnTypes?TypeOfEvent=Кино";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
```

```
        }
    }
}
```

Листинг В.9 – Файл KinoPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.KinoPage"
             BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.10 – Файл TheatrPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TheatrPage : ContentPage
    {
        public TheatrPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetEventsOnTypes()
        {
```

```
            url = "http://192.168.1.4:5002/GetEventsOnType?TypeOfEvent=Кино";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.11 – Файл TheatrPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.TheatrPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.12 – Файл KoncertPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class KoncertPage : ContentPage
    {
        public KoncertPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
```

```
        }

        public async Task<ActionResult> GetEventsOnTypes()
        {
            url = "http://192.168.1.4:5002/GetEventsOnType?TypeOfEvent=Концерт";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
            }
      }
}
```

Листинг В.13 – Файл KoncertPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.KoncertPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.14 – Файл MuseumPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MuseumPage : ContentPage
    {
        public MuseumPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
```

```
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetEventsOnTypes()
        {
            url = "http://192.168.1.4:5002/GetEventsOnType?TypeOfEvent=Кино";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.15 – Файл MuseumPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.MuseumPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.16 – Файл FavoritesPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FavoritesPage : ContentPage
    {
        public FavoritesPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
```

```
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetFavorites()
        {
            url = "http://192.168.1.4:5002/GetFavorites?{ID}";
            HttpClient client = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.17 – Файл FavoritesPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.FavoritesPage" BackgroundColor="#141412">

    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.18 – Файл PoiskPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PoiskPage : ContentPage
    {
        public PoiskPage()
```

```
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetEventsOnNames()
        {
            url = "http://192.168.1.4:5002/GetEventOnName?{EventName}";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.19 – Файл PoiskPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.PoiskPage" BackgroundColor="#141412" >


    <StackLayout>
        <StackLayout>
            <SearchBar Placeholder="Введите название мероприятия"
PlaceholderColor="White" CancelButtonColor="White" TextColor="White"  />
        </StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.20 – Файл ProfilesPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Profiles : ContentPage
    {
        //ffimageloading:CachedImage
        public static Profiles currentObject { get; private set; }
        private Page BeforePage;
        private int IDUseres { get; set; }
        public Profiles(int IDUser, Page page = null)
        {
            BeforePage = page;
            InitializeComponent();
            this.IDUser = IDUseres;
            currentObject = this;
        }

        protected override bool OnBackButtonPressed()
        {
            if (BeforePage != null)
            {
                MainPage.Current.Detail = new NavigationPage(BeforePage);
                App.Current.MainPage = MainPage.Current;
                return true;
            }
            return base.OnBackButtonPressed();
        }
        private async void GetArticlesProfile()
        {
            var response = await GetArticlesProfileResponse();
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var ArticlesJson = await response.Content.ReadAsStringAsync();
                var Articles =
JsonConvert.DeserializeObject<List<PlaceHelper>>(ArticlesJson);
                CreateArticlesProfile(Articles);
            }
        }

        private void CreateArticlesProfile(List<PlaceHelper> articles)
        {
            foreach (var article in articles)
            {
                Frame frame = new Frame();
                frame.CornerRadius = 20;
                frame.BackgroundColor = Color.LightGray;

                Label TitleArticle = new Label();
                TitleArticle.Text = article.Name;
                TitleArticle.TextColor = Color.Black;
                TitleArticle.FontSize = 18;

                Label AdressArticle = new Label();
                AdressArticle.Text = article.Adress;
                AdressArticle.TextColor = Color.DarkGray;
                AdressArticle.FontSize = 16;


                StackLayout stackLayout = new StackLayout();
```

```
                if (IDUser == App.IDCurrentUser)
                {
                    Grid gridButtonsForArticle = new Grid();

                    var columnFirst = new ColumnDefinition();
                    columnFirst.Width = new GridLength(5, GridUnitType.Star);

                    var columnSecond = new ColumnDefinition();
                    columnSecond.Width = new GridLength(5, GridUnitType.Star);

                    gridButtonsForArticle.ColumnDefinitions.Add(columnFirst);
                    gridButtonsForArticle.ColumnDefinitions.Add(columnSecond);

                    Button button = new Button();
                    button.Text = "Удалить статью";
                    button.BackgroundColor = Color.DarkGray;
                    button.FontSize = 8;
                    button.CornerRadius = 25;
                    button.Clicked += async (s, e) =>
                    {
                        var result = await ConfirmDialog();
                        if (result)
                        {
                            ArticlesProfile.Children.Remove(frame);
                            DeleteArticle(article.IDArticle);
                        }
                    };
                    Button buttonEditArticle = new Button();
                    buttonEditArticle.Text = "Редактировать статью";
                    buttonEditArticle.BackgroundColor = Color.DarkGray;
                    buttonEditArticle.FontSize = 8;
                    buttonEditArticle.CornerRadius = 25;
                    buttonEditArticle.Clicked += async (s, e) =>
                    {
                        App.Current.MainPage = new NavigationPage(new
AddNews(this, article.IDArticle));
                    };
                    gridButtonsForArticle.Children.Add(button);
                    gridButtonsForArticle.Children.Add(buttonEditArticle);

                    Grid.SetColumn(button, 0);
                    Grid.SetColumn(buttonEditArticle, 1);

                    stackLayout.Children.Add(gridButtonsForArticle);
                }

                stackLayout.Children.Add(TitleArticle);
                stackLayout.Children.Add(AdressArticle);

                frame.Content = stackLayout;

                var tapGestureRecognizer = new TapGestureRecognizer();
                tapGestureRecognizer.NumberOfTapsRequired = 1;
                tapGestureRecognizer.Tapped += (s, e) =>
                {
                    ShowArticle(article.IDArticle);
                };
                frame.GestureRecognizers.Add(tapGestureRecognizer);

                ArticlesProfile.Children.Add(frame);
            }
        }
```

```
        private async void GetDataProfile()
        {
            var response = await GetProfileData();
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var CurrentProfileDataJson = await
response.Content.ReadAsStringAsync();
                var CurrentProfileData =
JsonConvert.DeserializeObject<ProfileData>(CurrentProfileDataJson);
                SetProfileView(CurrentProfileData);
                //await DisplayNews(News);
            }
        }

        private void SetProfileView(ProfileData currentProfileData)
        {
            if (currentProfileData.ImageProfile != null)
            {
                //Stream stream = new
MemoryStream(currentProfileData.ImageProfile);
                ImageProfile.Source = ImageSource.FromStream(() => new
MemoryStream(currentProfileData.ImageProfile));
            }
            LabelArticlesCount.Text =
currentProfileData.CountArticles.ToString();
            LabelNickname.Text = currentProfileData.Nickname;
            LabelSubscribers.Text = currentProfileData.Subscribers.ToString();
            //LabelStatusCount.Text = currentProfileData.Status.ToString();
            if (currentProfileData.Description != null)
                LabelDescription.Text =
currentProfileData.Description.ToString();
        }

        private async Task<HttpResponseMessage> GetProfileData()
        {
            var client = new HttpClient();
            HttpResponseMessage responseMessage = null;
            try
            {
                responseMessage = await
client.GetAsync($"http://192.168.0.29:5000/GetProfile/{IDUser}");
            }
            catch (Exception ex)
            {
                if (responseMessage == null)
                {
                    responseMessage = new HttpResponseMessage();
                }
                responseMessage.StatusCode = HttpStatusCode.InternalServerError;
                await DisplayAlert("Ошибка", "Подключение отсутствует",
"Закрыть");
            }
            return responseMessage;
        }
        private void SetProfileView()
        {
            LineChoice.TranslateTo(0, 0);
            ScrollViewArticles.IsVisible = false;
            ScrollViewSubscribers.IsVisible = false;
            ScrollViewProfile.IsVisible = true;
        }
        private void SetArticlesView()
```

```
        {
            LineChoice.TranslateTo(LineChoice.Width + GridMenu.ColumnSpacing,
0);
            ScrollViewProfile.IsVisible = false;
            ScrollViewSubscribers.IsVisible = false;
            ScrollViewArticles.IsVisible = true;
        }
        private void SetSubscribersView()
        {
            LineChoice.TranslateTo((LineChoice.Width + GridMenu.ColumnSpacing) *
2, 0);
            ScrollViewProfile.IsVisible = false;
            ScrollViewArticles.IsVisible = false;
            ScrollViewSubscribers.IsVisible = true;
        }
        private void Button_Clicked(object sender, EventArgs e)
        {
            SetProfileView();
        }

        private void TapGestureRecognizer_Tapped(object sender, EventArgs e)
        {
            if (IDUser != App.IDCurrentUser)
                return;
            if (SettingImageProfile.IsVisible)
            {
                SettingImageProfile.IsVisible = false;
                return;
            }
            SettingImageProfile.IsVisible = true;
        }

        private void ImageButton_Clicked_S(object sender, EventArgs e)
        {
            SettingImageProfile.IsVisible = false;
        }
        public async void UpdatePhoto(ImageSource imagesrc)
        {
            //var Image = new CachedImage { Source = imagesrc, Aspect =
Aspect.AspectFill, DownsampleToViewSize = true };
            ImageProfile.Source = imagesrc;
            var response = await UpdatePhotoResponse(imagesrc);
            if (response.StatusCode != HttpStatusCode.OK)
            {
                return;
            }
        }
        private void Button_Clicked_I(object sender, EventArgs e)
        {
            SettingImageProfile.IsVisible = false;
            Intent intent = new Intent(Intent.ActionSend);
            intent.SetType("image/*");
            intent.SetAction(Intent.ActionGetContent);

App.ActivityCurrent.StartActivityForResult(Intent.CreateChooser(intent, "Select
Picture"), 1);
        }
        private byte[] ImageSourceToBytes(ImageSource imageSource)
        {
            if (imageSource == null)
                return null;
            StreamImageSource streamImageSource =
(StreamImageSource)imageSource;
```

```
            System.Threading.CancellationToken cancellationToken =
            System.Threading.CancellationToken.None;
            Task<Stream> task = streamImageSource.Stream(cancellationToken);
            Stream stream = task.Result;
            byte[] bytesAvailable = new byte[stream.Length];
            stream.Read(bytesAvailable, 0, bytesAvailable.Length);
            return bytesAvailable;
        }
        private async Task<HttpResponseMessage> UpdatePhotoResponse(ImageSource
image)
        {
            var client = new HttpClient();
            HttpResponseMessage responseMessage = null;
            var imageBytes = ImageSourceToBytes(image);
            JsonContent content = JsonContent.Create(new ProfileImage { ID =
App.IDCurrentUser, Image = imageBytes });
            try
            {
                responseMessage = await
client.PutAsync($"http://192.168.0.29:5000/UpdateImageProfile/", content);
            }
            catch (Exception ex)
            {
                if (responseMessage == null)
                {
                    responseMessage = new HttpResponseMessage();
                }
                responseMessage.StatusCode = HttpStatusCode.InternalServerError;
                await DisplayAlert("Ошибка", "Подключение отсутствует",
"Закрыть");
            }
            return responseMessage;
        }

        private void TapGestureRecognizer_Tapped_1(object sender, EventArgs e)
        {
            if (App.IDCurrentUser != IDUser)
                return;
            LabelDescription.IsVisible = false;
            EditorDescription.Text = LabelDescription.Text;
            EditorDescription.IsVisible = true;
            EditorDescription.Focus();
        }

        private void EditorDescription_Unfocused(object sender, FocusEventArgs
e)
        {
            EditorDescription.IsVisible = false;
            LabelDescription.Text = EditorDescription.Text;
            LabelDescription.IsVisible = true;
        }

        private async void Button_Clicked_4(object sender, EventArgs e)
        {
            ImageProfile.Source = null;
            SettingImageProfile.IsVisible = false;
            await UpdatePhotoResponse(null);
        }
    }
    public class ProfileData
    {
        public string Nickname { get; set; }
```

```
        public string Description { get; set; }
        public string ImageProfile { get; set; }
    }
    public class ProfileImage
    {
        public int ID { get; set; }
        public byte[] Image { get; set; }
    }
}
```

Листинг В.21 – Файл ProfilePage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.ProfilePage" BackgroundColor="#141412">
    <ContentPage.Content>
        <Grid x:Name="MainGrid" RowSpacing="0" ColumnSpacing="0"
BackgroundColor="#30302e">
            <Grid.RowDefinitions>
                <RowDefinition Height="200*"></RowDefinition>
                <RowDefinition Height="90*"></RowDefinition>
                <RowDefinition Height="300*"></RowDefinition>
            </Grid.RowDefinitions>

            <yummy:PancakeView x:Name="FrameImage" BackgroundColor="Coral"
CornerRadius="0,0,40,40" Grid.Row="0" Padding="0" IsClippedToBounds="True">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                    </Grid.ColumnDefinitions>
                    <ffimageloading:CachedImage x:Name="ImageProfile"
BackgroundColor="LightGray" Aspect="AspectFill" Grid.ColumnSpan="5"
Grid.RowSpan="5">
                        <ffimageloading:CachedImage.GestureRecognizers>
                            <TapGestureRecognizer NumberOfTapsRequired="1"
Tapped="TapGestureRecognizer_Tapped"></TapGestureRecognizer>
                        </ffimageloading:CachedImage.GestureRecognizers>
                    </ffimageloading:CachedImage>
                    <Frame x:Name="SettingImageProfile" IsVisible="false"
BackgroundColor="White" CornerRadius="15" Grid.Column="1" Grid.Row="1"
Grid.RowSpan="3" Grid.ColumnSpan="3" IsClippedToBounds="True">
                        <ScrollView>
                            <StackLayout IsClippedToBounds="True" Spacing="1">
                                <ImageButton Clicked="ImageButton_Clicked"
Source="confirmIcon.png" WidthRequest="20" HeightRequest="20"
BackgroundColor="White" HorizontalOptions="End"></ImageButton>
                                <Button BackgroundColor="White"
HeightRequest="38" Text="Редактировать фото" Clicked="Button_Clicked_3"
FontSize="14" TextColor="Black"></Button>
```

```
                                <Button BackgroundColor="White"
HeightRequest="38" ContentLayout="Top,0" Text="Удалить фото"
Clicked="Button_Clicked_4" FontSize="14" TextColor="Black"></Button>
                            </StackLayout>
                        </ScrollView>
                    </Frame>
                </Grid>
            </yummy:PancakeView>
            <Frame x:Name="FrameInformation" Background="Black" Grid.Row="1"
Padding="20" IsClippedToBounds="True">
                <Grid >
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="100*"></ColumnDefinition>
                        <ColumnDefinition Width="100*"></ColumnDefinition>
                    </Grid.ColumnDefinitions>
                    <Grid Grid.Column="0" RowSpacing="0">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="50*"></RowDefinition>
                            <RowDefinition Height="50*"></RowDefinition>
                        </Grid.RowDefinitions>
                </Grid>
            </Frame>
            <yummy:PancakeView x:Name="FrameData" BackgroundColor="White"
CornerRadius="40,40,0,0" IsClippedToBounds="True" Grid.Row="2">
                <Grid IsClippedToBounds="True">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="100*"></RowDefinition>
                        <RowDefinition Height="500*"></RowDefinition>
                    </Grid.RowDefinitions>
                    <Grid Grid.Row="0" BackgroundColor="LightGray"
Padding="15,5,15,0" RowSpacing="0" x:Name="GridMenu">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="100*"></ColumnDefinition>
                            <ColumnDefinition Width="100*"></ColumnDefinition>
                            <ColumnDefinition Width="100*"></ColumnDefinition>
                        </Grid.ColumnDefinitions>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="150*"></RowDefinition>
                            <RowDefinition Height="15"></RowDefinition>
                        </Grid.RowDefinitions>
                        <Line x:Name="LineChoice" Stroke="Gray" Grid.Row="1"
X1="0" X2="150" StrokeThickness="5"></Line>
                    </Grid>
                    <Grid x:Name="GridBar" Grid.Row="1">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="*"></ColumnDefinition>
                        </Grid.ColumnDefinitions>
                        <ScrollView x:Name="ScrollViewProfile">
                            <Grid RowSpacing="0">
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="55"></RowDefinition>
                                    <RowDefinition Height="55"></RowDefinition>
                                    <RowDefinition Height="33*"></RowDefinition>
                                </Grid.RowDefinitions>
                                <Label x:Name="LabelNickname" Text="Nickname"
Margin="20,20,0,0" HorizontalOptions="Start" FontSize="20" TextColor="Black"
FontAttributes="Bold" Grid.Row="0"></Label>
                                <Label Text="About" Margin="20,20,0,0"
HorizontalOptions="Start" FontSize="20" TextColor="Black" FontAttributes="Bold"
Grid.Row="1"></Label>
                                <Label x:Name="LabelDescription" Text="Описание
отсутствует" Margin="20,0,0,0" HorizontalOptions="Start" FontSize="20"
TextColor="Black" Grid.Row="2">
```

```
                                    <Label.GestureRecognizers>
                                        <TapGestureRecognizer
NumberOfTapsRequired="1"
Tapped="TapGestureRecognizer_Tapped_1"></TapGestureRecognizer>
                                    </Label.GestureRecognizers>
                                </Label>
                                <Editor x:Name="EditorDescription"
Margin="20,0,0,0" Unfocused="EditorDescription_Unfocused" Grid.Row="2"
FontSize="20" TextColor="Black" IsVisible="false"></Editor>
                            </Grid>
                        </ScrollView>
                        <ScrollView x:Name="ScrollViewArticles"
IsVisible="false" IsClippedToBounds="True" Margin="0">
                        </ScrollView>
                    </Grid>
                </Grid>
            </yummy:PancakeView>
        </Grid>
    </ContentPage.Content>
</ContentPage>
```

Листинг В.22 – Файл RegisterFlyout.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegisterFlyout : ContentPage
    {
        public ListView ListView;

        public RegisterFlyout()
        {
            InitializeComponent();
        }
        private IUserUnitWorks UserUnitWorks;
        private ValidationRegisters validationRegisters;
        private Page Views;
        private bool IsWaitings;
        public ModelViewRegister(Page view)
        {
            Views = views;
            UserUnitWorks = new ServiceUsers();
            IsWaitings = false;
            validationRegisters = new ValidationRegisters();
        }
        private string _TextLogins { get; set; }
        public string TextLogins
        {
            get
            {
                return _TextLogins;
            }
            set
            {
                _TextLogins = value;
                OnPropertyChanged("TextLogin");
            }
        }

        private string _TextPasswords { get; set; }
        public string TextPasswords
        {
```

```
            get
            {
                return _TextPasswords;
            }
            set
            {
                _TextPasswords = value;
                OnPropertyChanged("TextPassword");
            }
        }

        private string _TextNicknames { get; set; }
        public string TextNicknames
        {
            get
            {
                return _TextNicknames;
            }
            set
            {
                _TextNicknames = value;
                OnPropertyChanged("TextNickname");
            }
        }
        private ICommand _clickCommands;
        public ICommand ClickCommands
        {
            get
            {
                return _clickCommand s?? (_clickCommands = new DelegateCommand(
                    (parameters) => ExecuteProcedure(parameters)));
            }
        }
        private ICommand _CommandChangePages;
        public ICommand CommandChangePages
        {
            get
            {
                return _CommandChangePages ?? (_CommandChangePages = new
DelegateCommand(
                    (parameters) => ChangePageProcedures(parameters)));
            }
        }

        private void ChangePageProcedures(object parameters)
        {
            App.Current.MainPage = new NavigationPage(new Authorization());
        }

        private async void ExecuteProcedures(object parameters)
        {
            if (IsWaitings)
                return;
            IsWaitings = true;
            User users = new User { Login = _TextLogins, Password =
_TextPasswords, Nickname = _TextNicknames };
            if (!validationRegister.SuitForm(users))
            {
                await View.DisplayAlert("Результат", "Некорректные данные,
введите заново", "Закрыть");
                IsWaitings = false;
                return;
            }
```

```
            var response = await UserUnitWork.Register(users);
            var stream = response.Content;
            string str = await stream.ReadAsStringAsync();
            if (response.IsSuccessStatusCode)
            {
                if (int.TryParse(str, out int IDUser))
                {
                    App.IDCurrentUser = IDUser;
                    App.Current.MainPage = new MainPage();
                    return;
                }
            }
            await View.DisplayAlert("Результат", str, "Закрыть");
            IsWaitings = false;
        }

        private void OnPropertyChangeds(string prop = "")
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));
        }
        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

Листинг В.23 – Файл RegisterFlyout.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.RegisterFlyout"
             Title="Регистрация" BackgroundColor="#141412" >
    <ContentPage.Content>
        <StackLayout Spacing="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="CenterAndExpand">
            <Editor x:Name="EditorNickname" Text="{Binding TextNickname}"
Placeholder="Имя" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorLogin" Text="{Binding TextLogin}"
Placeholder="Логин" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorPassword" Text="{Binding TextPassword}"
Placeholder="Пароль" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Button Text="Зарегистрировать" Command="{Binding
ClickCommand}"></Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Листинг В.24 – Файл AuthorizationFlyout.xaml.cs

```
namespace Poster
    {
        class ModelViewAuthorization : INotifyPropertyChanged
        {
            private IUserUnitWork UserUnitWorks;
            private Page Views;
            private bool IsWaitings;
            public ModelViewAuthorizations(Page views)
            {
```

```
            Views = views;
            UserUnitWorks = new ServiceUsers();
            IsWaitings = false;
        }

        private string _TextLogins { get; set; }
        public string TextLogins
        {
            get
            {
                return _TextLogins;
            }
            set
            {
                _TextLogins = value;
                OnPropertyChanged("TextLogin");
            }
        }

        private string _TextPasswords { get; set; }
        public string TextPasswords
        {
            get
            {
                return _TextPasswords;
            }
            set
            {
                _TextPasswords = value;
                OnPropertyChanged("TextPassword");
            }
        }

        private ICommand _clickCommands;
        public ICommand ClickCommansd
        {
            get
            {
                return _clickCommands ?? (_clickCommands = new
DelegateCommand(
                    (parameters) => ExecuteProcedure()));
            }
        }
        private ICommand _CommandChangePages;
        public ICommand CommandChangePages
        {
            get
            {
                return _CommandChangePages ?? (_CommandChangePages = new
DelegateCommand(
                    (parameters) => ChangePageProcedures(parameters)));
            }
        }
        private async void ChangePageProcedures(object parameters)
        {
            App.Current.MainPage = new NavigationPage(new Register());
        }
        public async void ExecuteProcedure()
        {
            if (IsWaitings)
                return;
            IsWaitings = true;
```

```
                User user = new User { Login = _TextLogins, Password =
_TextPasswords };
                var response = await UserUnitWorks.Authorization(user);
                var stream = response.Content;
                string str = await stream.ReadAsStringAsync();
                if (response.IsSuccessStatusCode)
                {
                    if (int.TryParse(str, out int IDUser))
                    {
                        App.IDCurrentUser = IDUser;
                        App.Current.MainPage = new MainPage();
                        Preferences.Set("Login", user.Login);
                        Preferences.Set("Password", user.Password);
                        return;
                    }
                }
                await View.DisplayAlert("Результат", str, "Закрыть");
                IsWaiting = false;
            }

            private void OnPropertyChangeds(string prop = "")
            {
                PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));
            }
            public event PropertyChangedEventHandler PropertyChanged;
        }
    }
}
```

Листинг В.25 – Файл AuthorizationFlyout.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.AuthorizationFlyout"
             Title="Авторизация"  BackgroundColor="#141412" >
    <ContentPage.Content>
        <StackLayout Spacing="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="CenterAndExpand">
            <Label Text="Авторизация"
                VerticalOptions="Center" HorizontalOptions="Center"
FontSize="18"/>
            <Editor x:Name="EditorLogin" Text="{Binding TextLogin}"
Placeholder="Логин" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorPassword" Text="{Binding TextPassword}"
Placeholder="Пароль" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Button Text="Авторизоваться" Command="{Binding
ClickCommand}"></Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Листинг В.26 – Файл TicketPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
```

```
    public partial class TiketPage : ContentPage
    {
        public TiketPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetTicket()
        {
            url = "http://192.168.1.4:5002/GetTicket";
            HttpClient client = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.27 – Файл TicketPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.TiketPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="eventsList" ItemsSource="{Binding Events}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Image Source="{Binding Image}"/>
                                    <Label Text="{Binding EventId}" />
                                    <Label Text="{Binding Genre}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Листинг В.28 – Файл ReviewPage.xaml.cs

```
namespace Poster
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ReviewPage : ContentPage
    {
        public ReviewPage()
        {
            InitializeComponent();
        }
        string url;
        JsonSerializerOptions optiones = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };
        private HttpClient GetClient()
        {
            HttpClient clients = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return clients;
        }

        public async Task<ActionResult> GetReview()
        {
            url = "http://192.168.1.4:5002/GetTicket";
            HttpClient clients = GetClient();
            string results = await client.GetStringAsync(url);
            return
System.Text.Json.JsonSerializer.Deserialize<ActionResult>(results, optiones);
        }
    }
}
```

Листинг В.29 – Файл ReviewPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Poster.ReviewPage" BackgroundColor="#141412">
    <StackLayout>
        <ListView x:Name="reviewList" ItemsSource="{Binding Review}"
              SelectedItem="{Binding SelectedEvent}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout Margin="0,0,0,30"
BackgroundColor="#30302e">
                                <StackLayout>
                                    <Label Text="{Binding Name}" />
                                    <Label Text="{Binding Discription}" />
                                    <Label Text="{Binding Ratings}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Приложение Г

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ

(рекомендуемое)

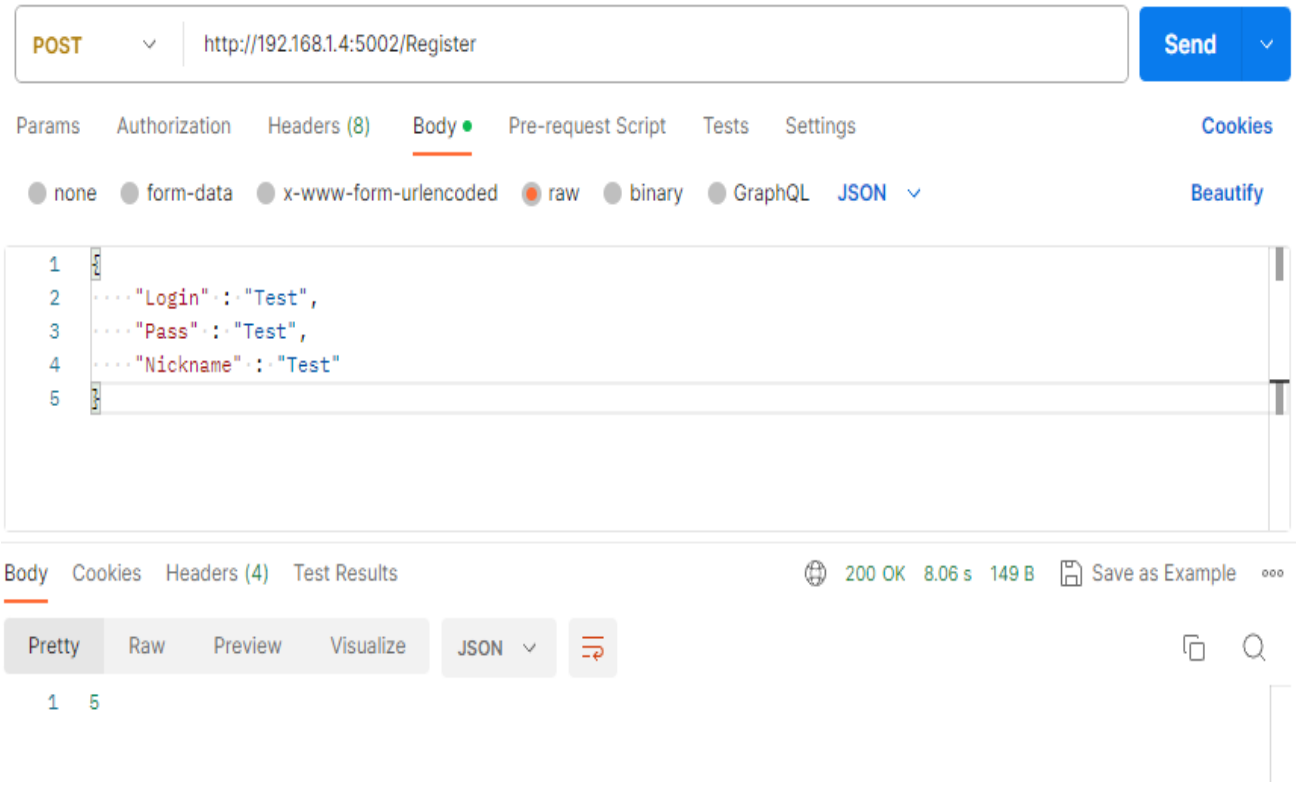


Рисунок Г.1 – Регистрация нового пользователя

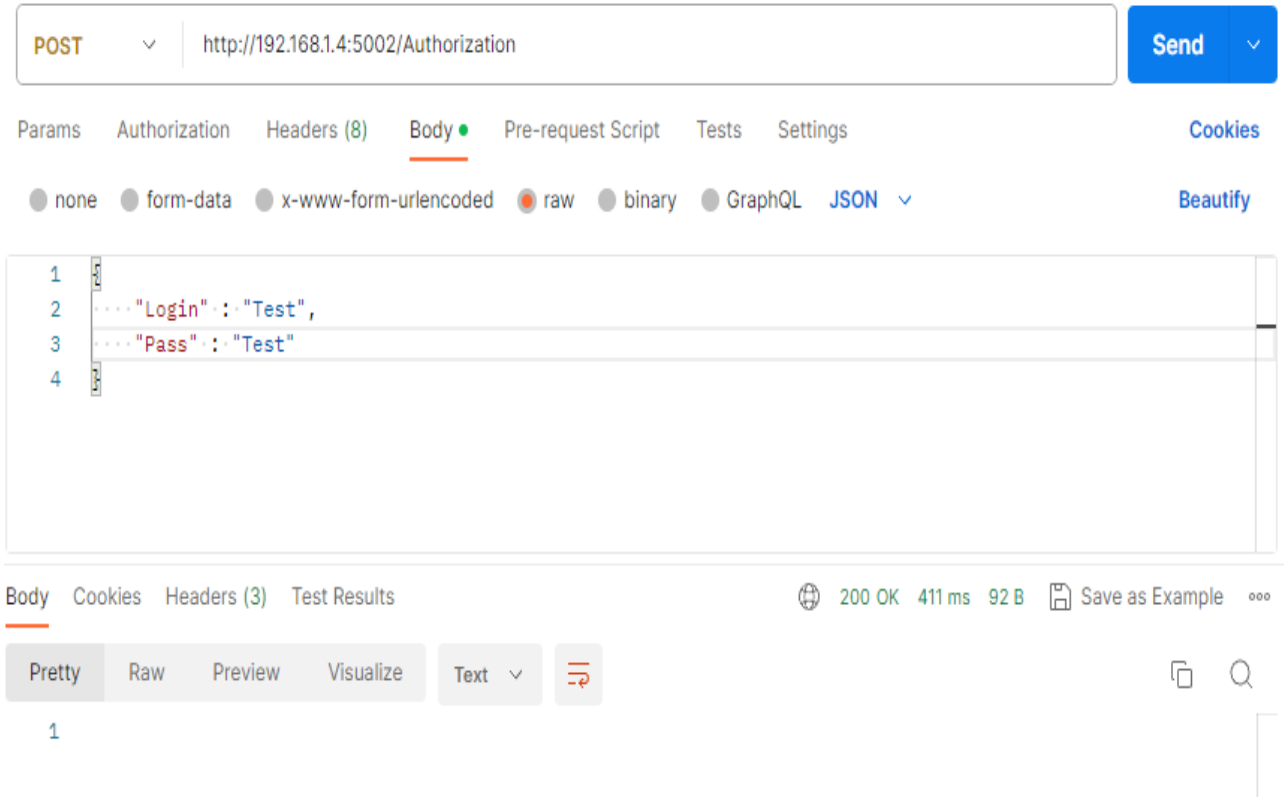


Рисунок Г.2 – Авторизация пользователя

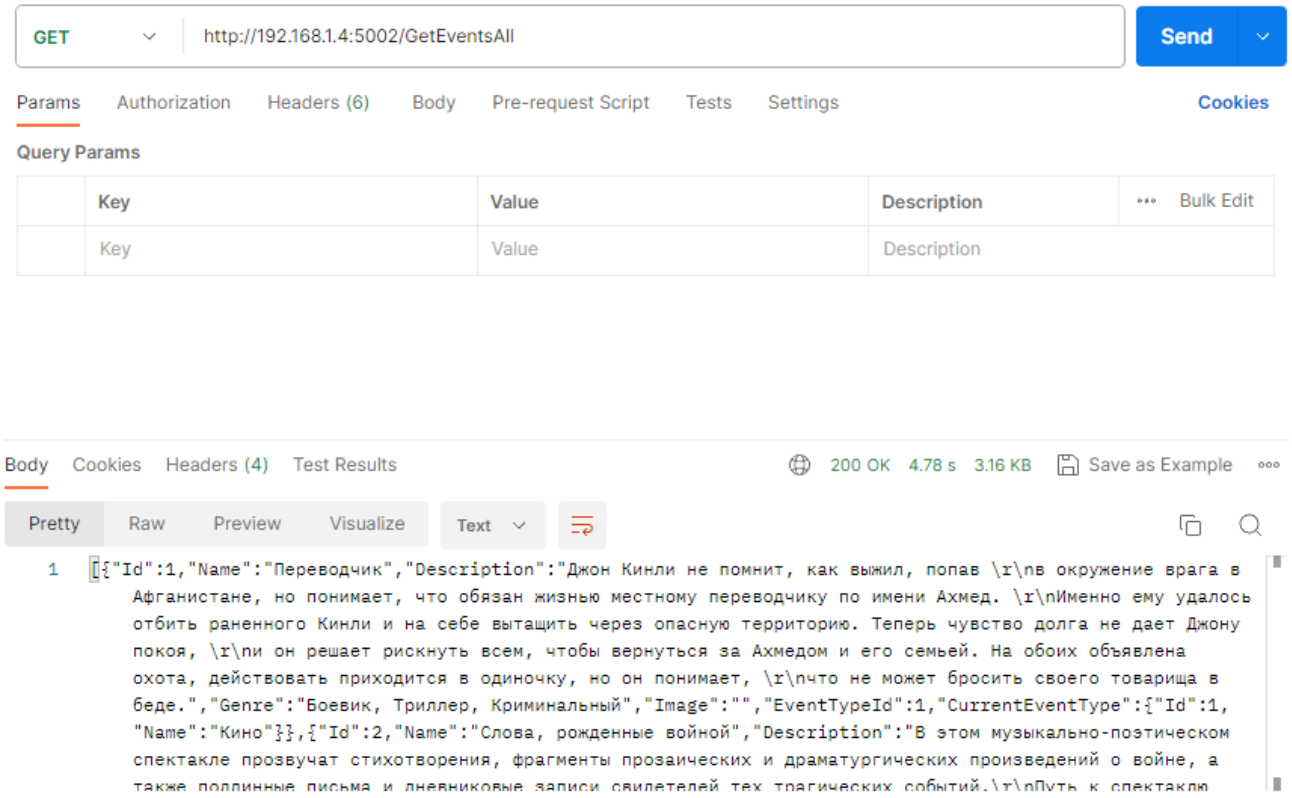


Рисунок Г.3 – Получения информации о всех мероприятиях

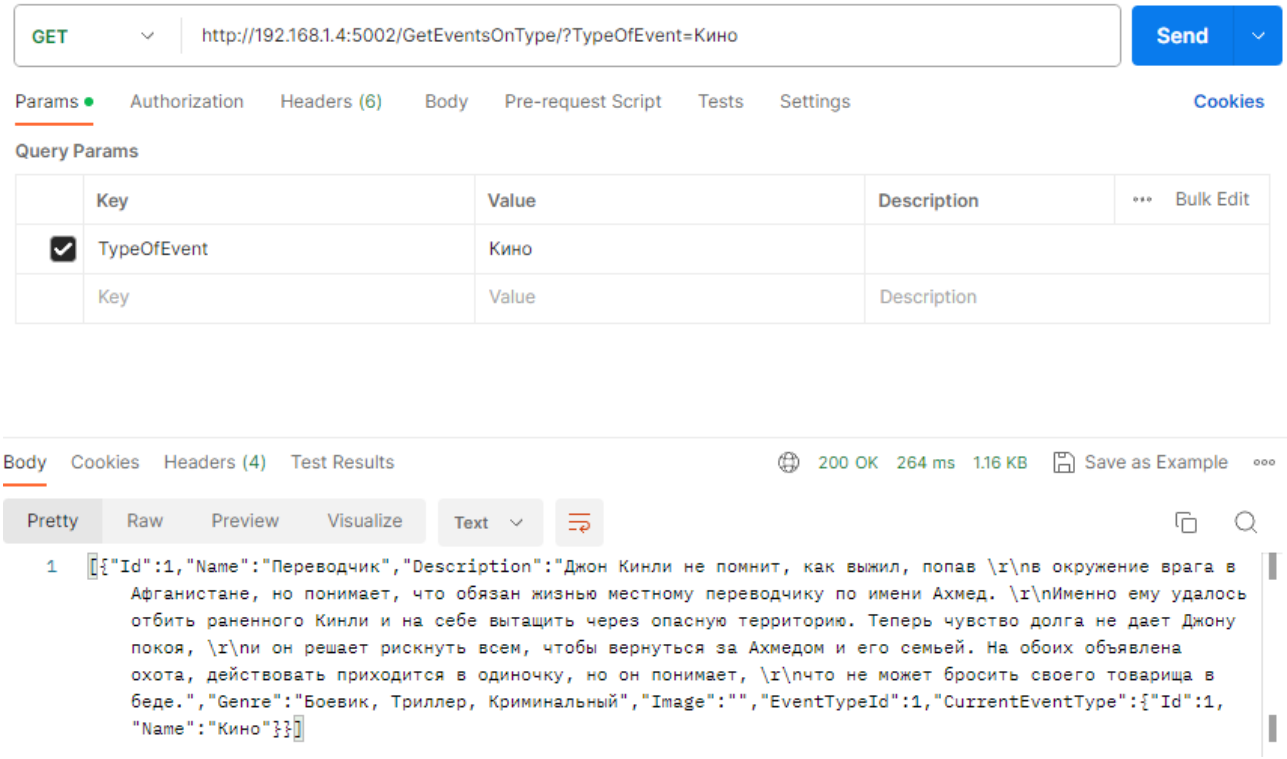


Рисунок Г.4 – Получения информации о мероприятии с категорией кино

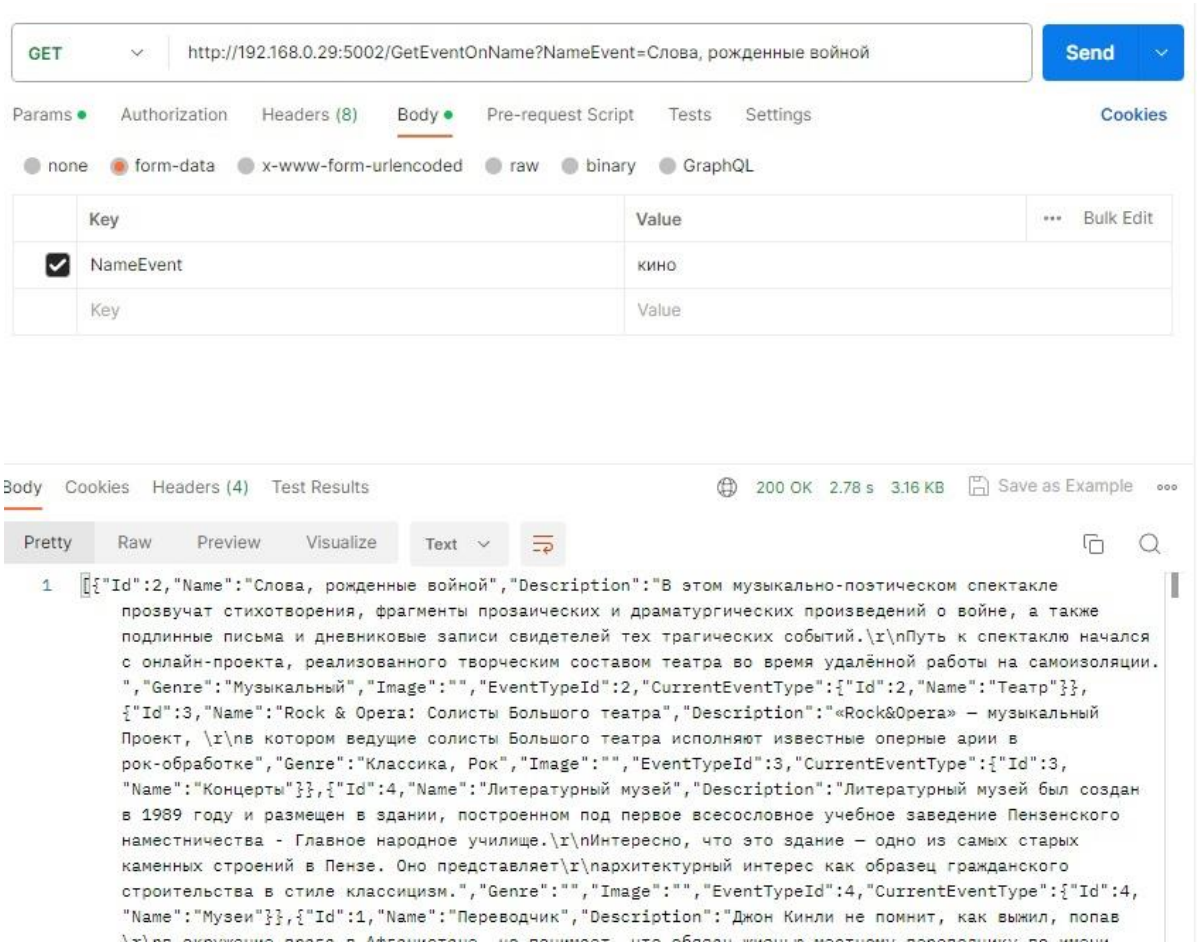


Рисунок Г.5 – Получение информации о мероприятии по названию

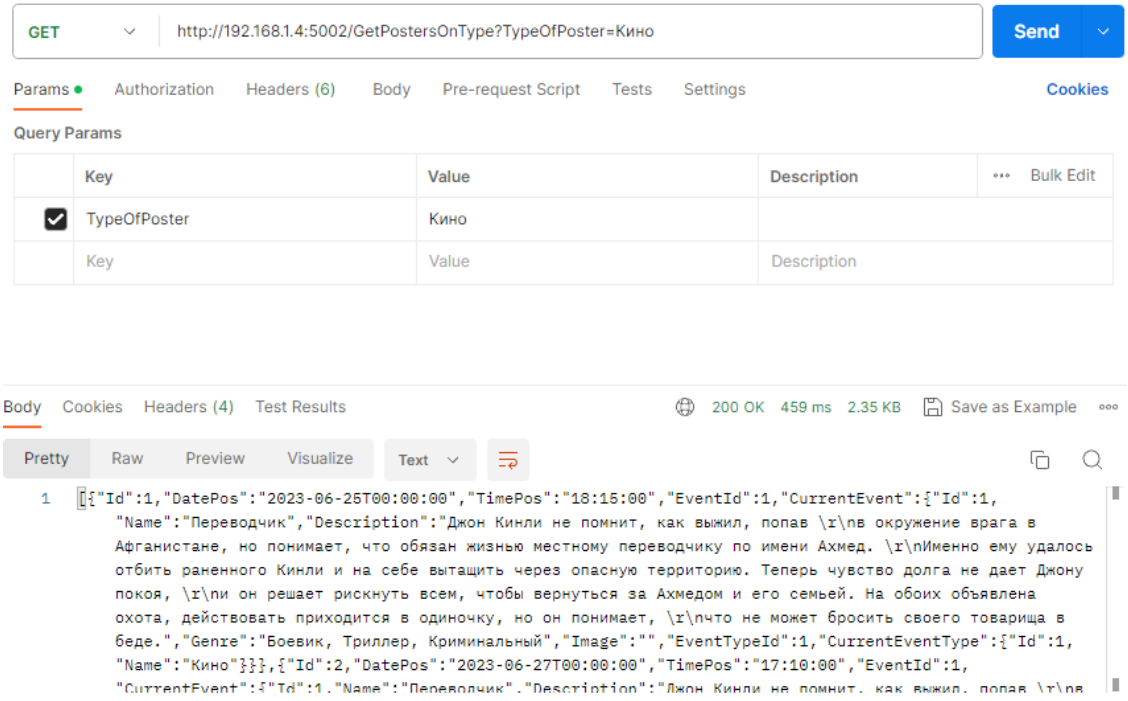


Рисунок Г.6 – Получение информации об афише мероприятия

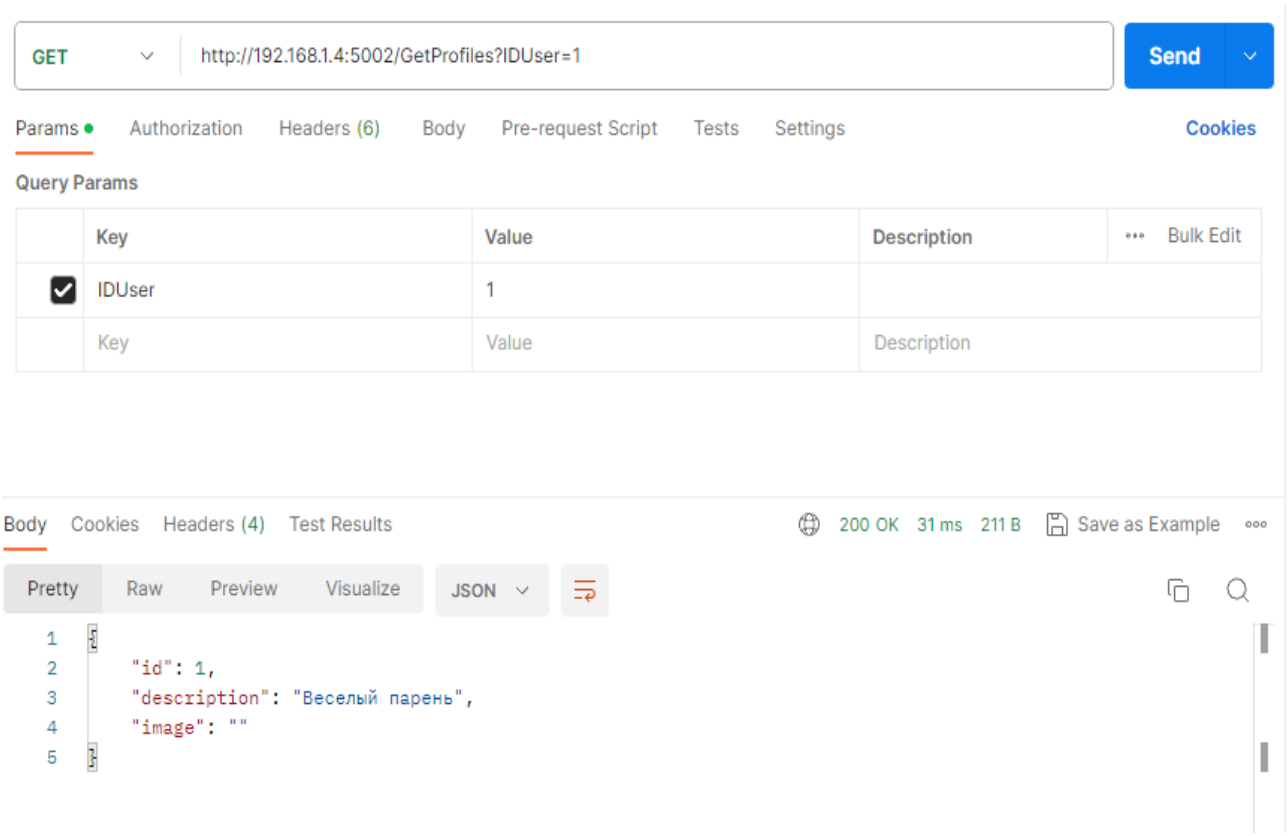


Рисунок Г.7 – Получение информации о профиле пользователя

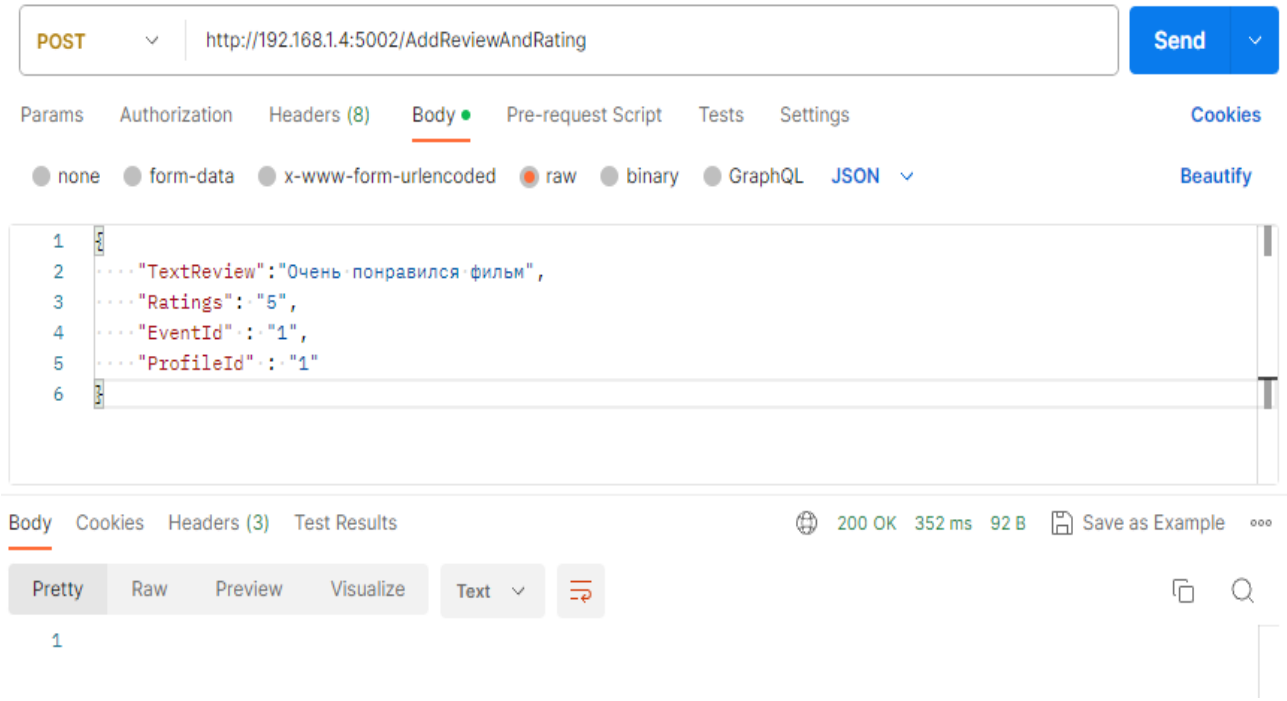


Рисунок Г.8 – Добавления отзыва

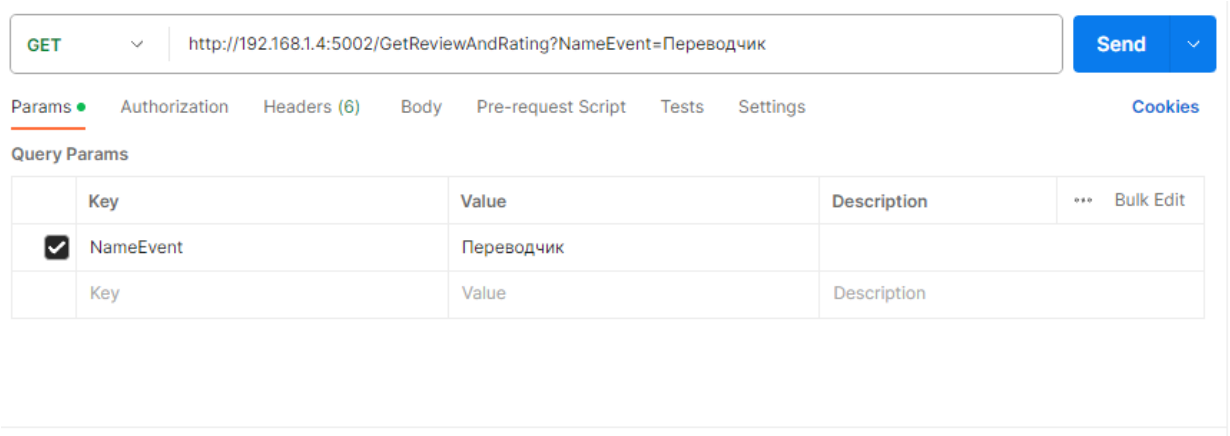


Рисунок Г.9 – Получение всех отзывов на мероприятия

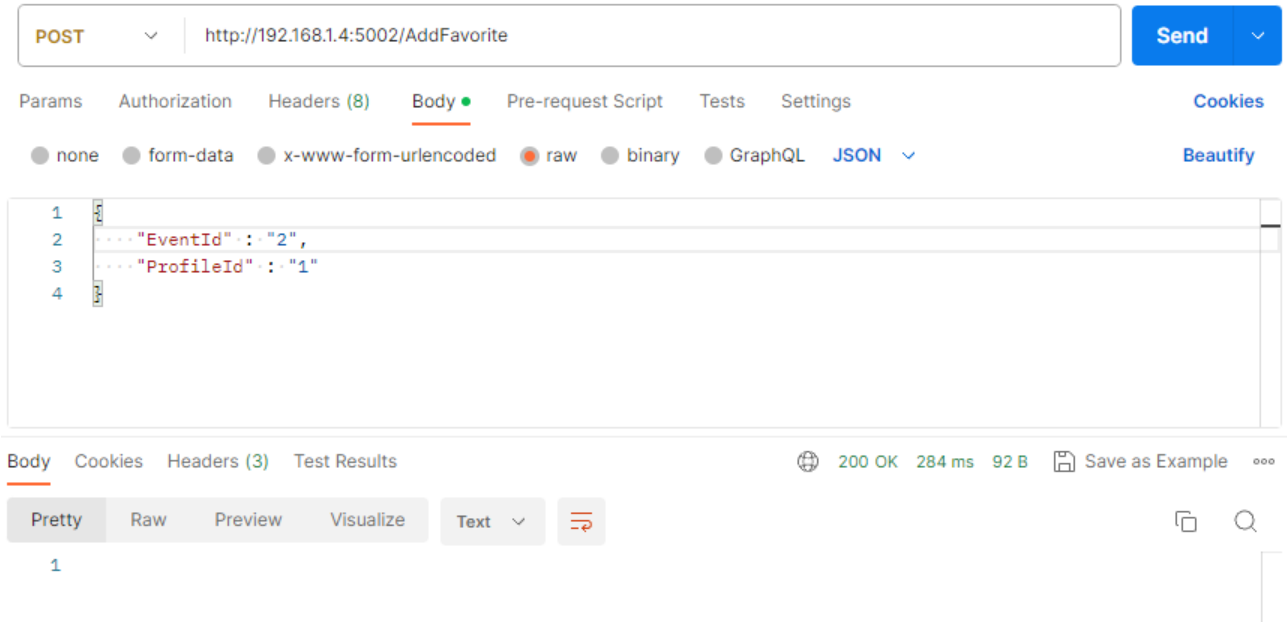


Рисунок Г.10 – Добавления мероприятия в избранное

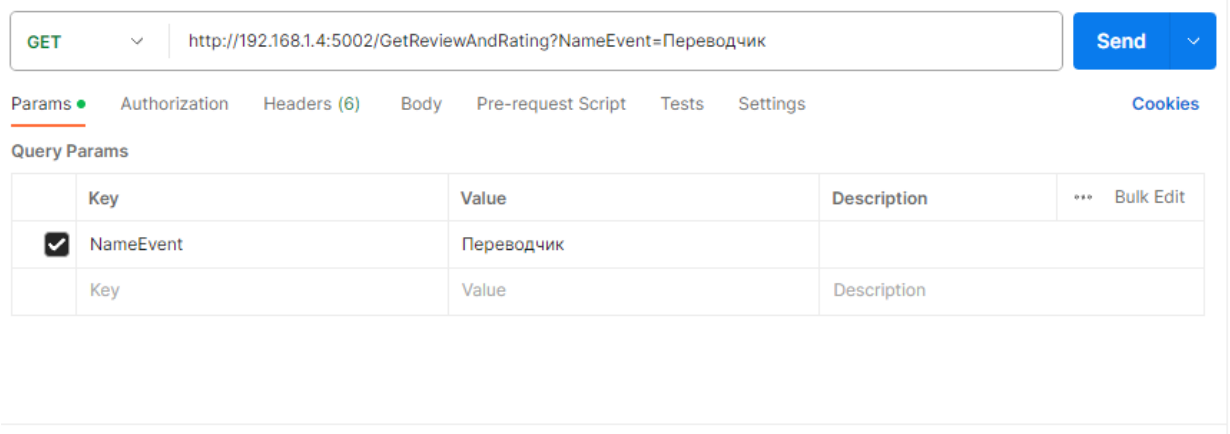


Рисунок Г.11 – Получения избранных мероприятия пользователя

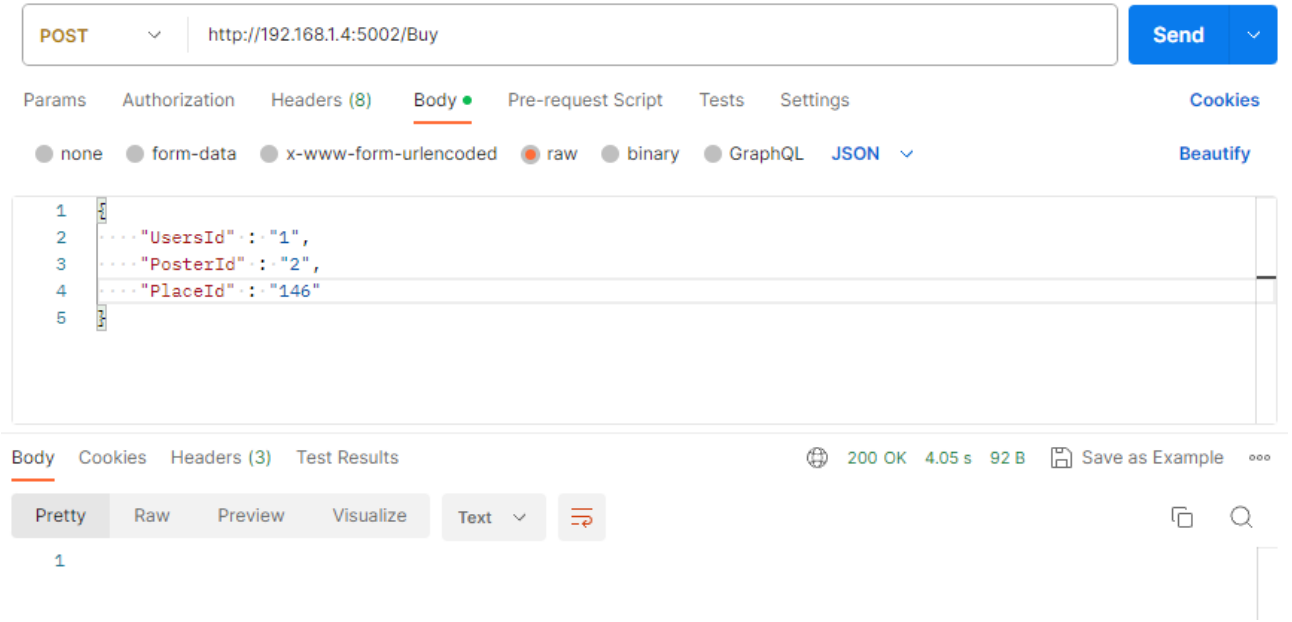


Рисунок Г.12 – Бронирование билета

Приложение Д

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

(рекомендуемое)

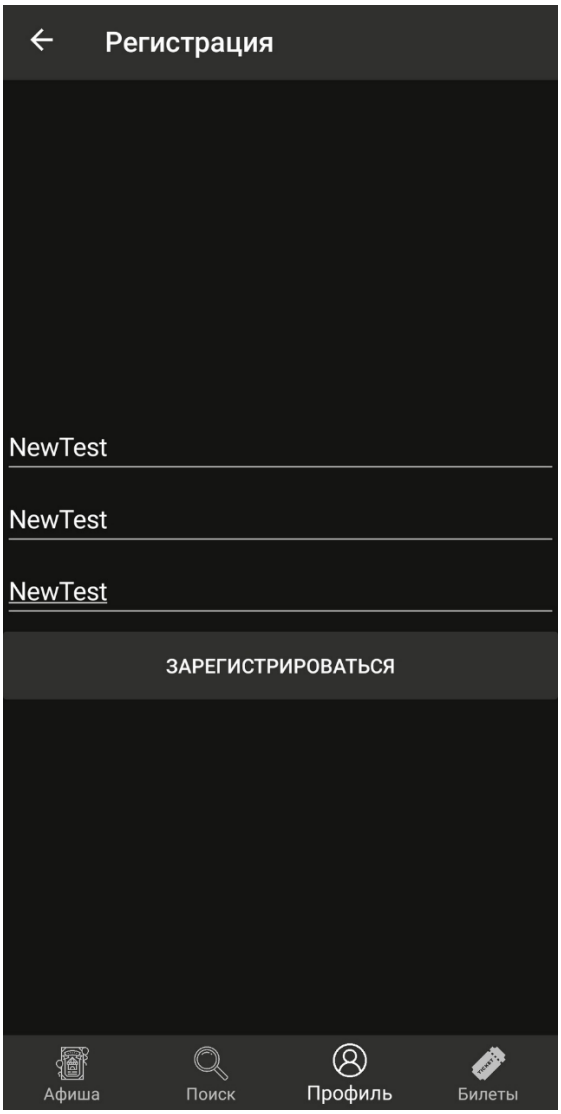


Рисунок Д.1 – Введенные данные для успешной регистрации пользователя

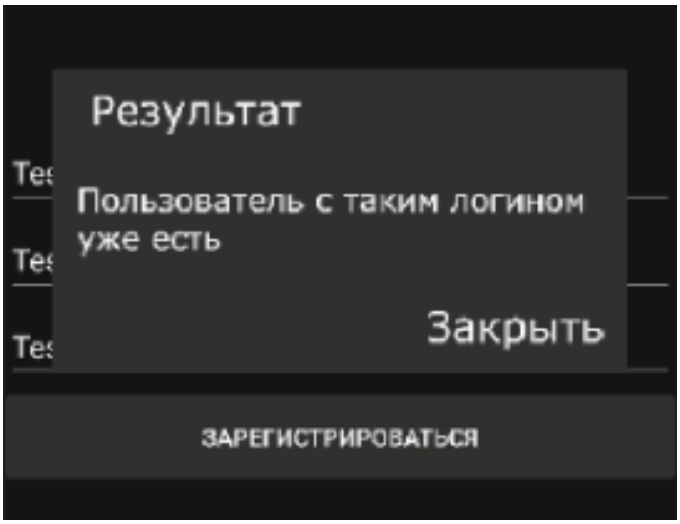


Рисунок Д.2 – Попытка регистрации под существующим логином

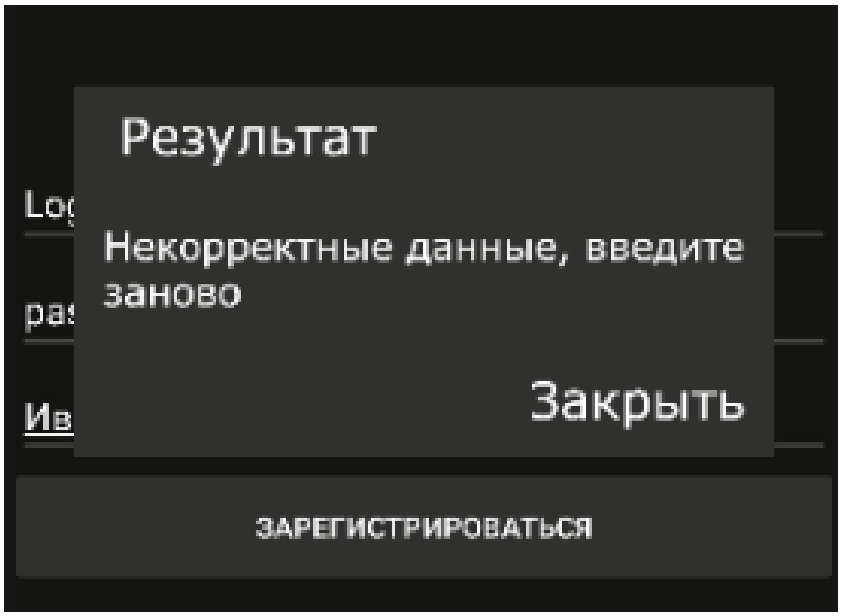


Рисунок Д.3 – Попытка регистрации с использованием пароля или логина кароче 4-х символов

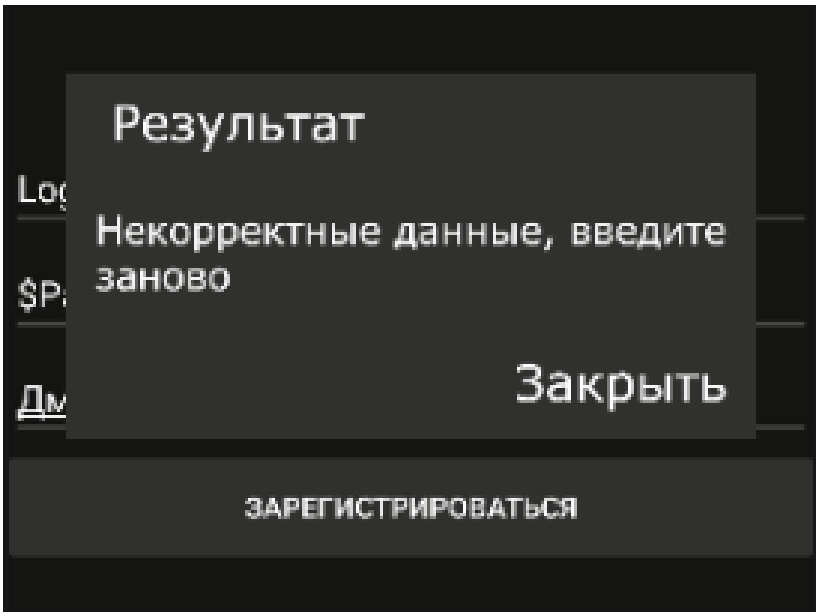


Рисунок Д.4 – Попытка регистрации при использование некорректных символов

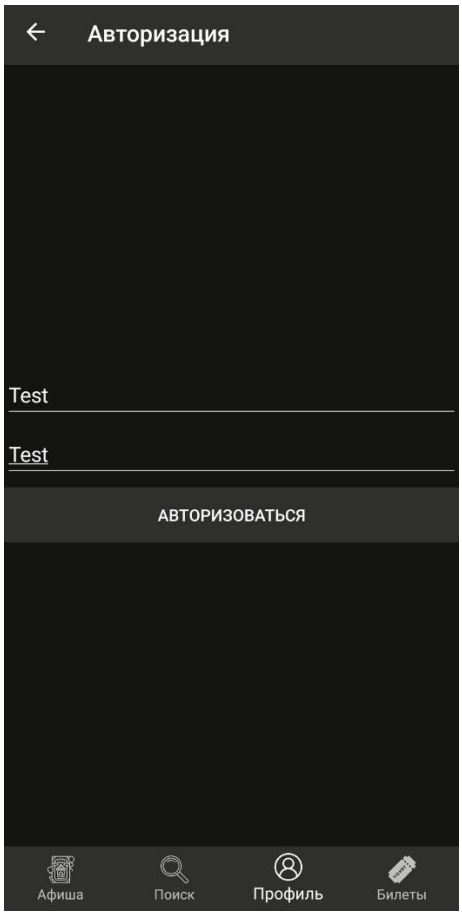


Рисунок Д.5 – Авторизации под существующим логином и паролем

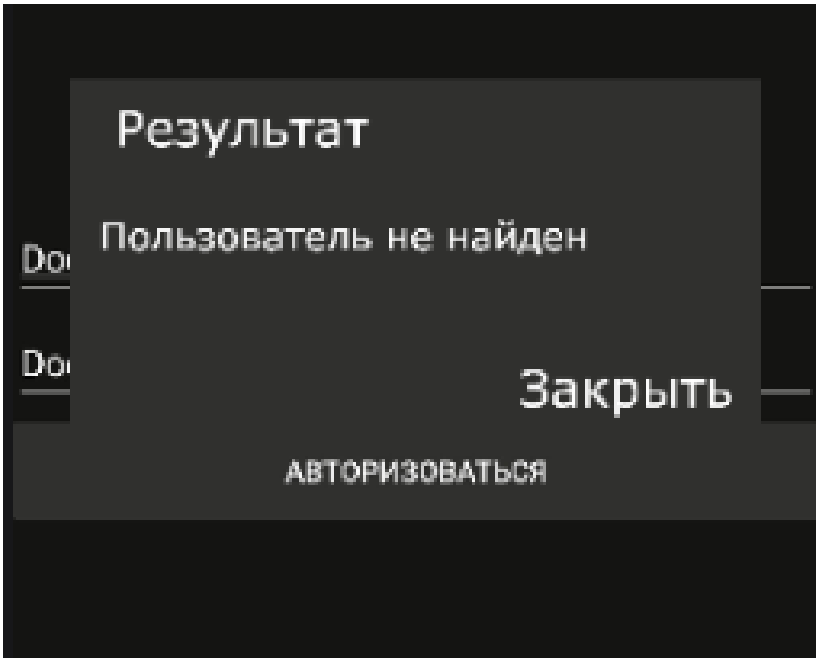


Рисунок Д.6 – Авторизация под несуществующим логином

Рисунок Д.7 – Просмотр всех мероприятий

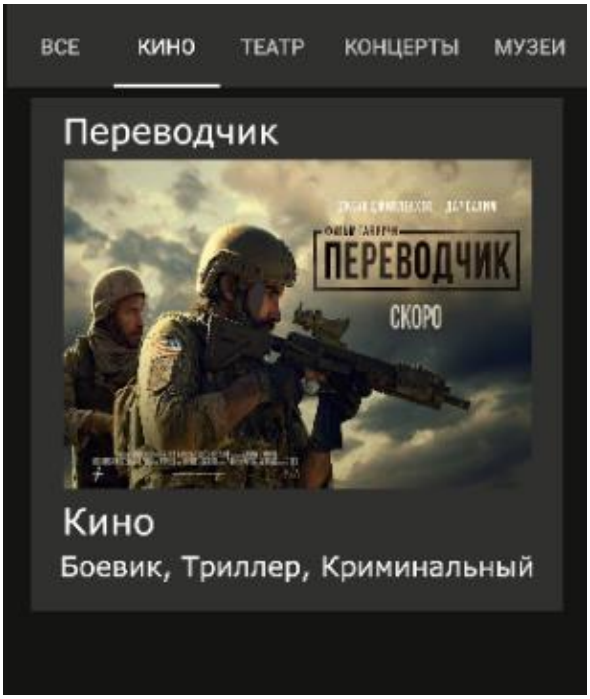


Рисунок Д.8 – Просмотр мероприятий по категории

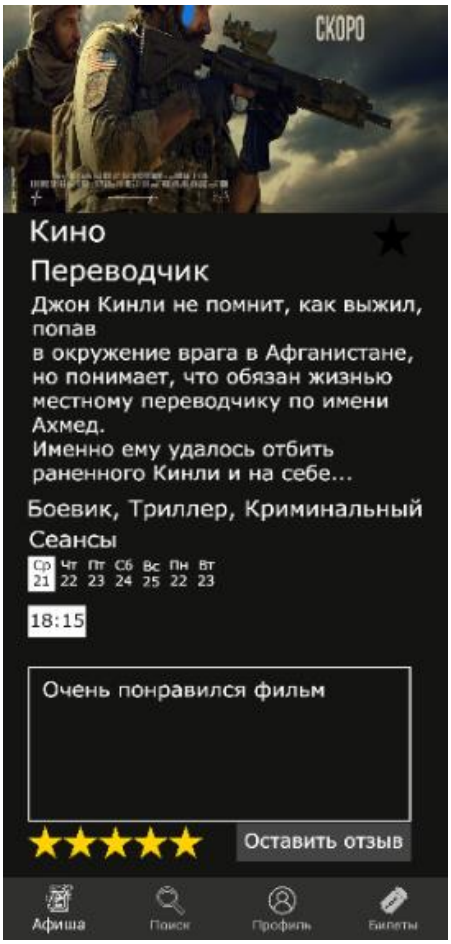


Рисунок Д.9 – Оставить отзыв

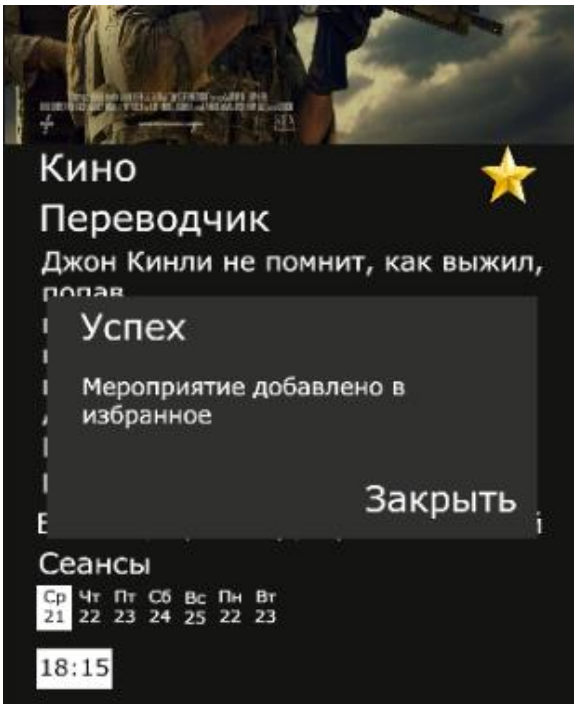


Рисунок Д.10 – Добавить в избранное

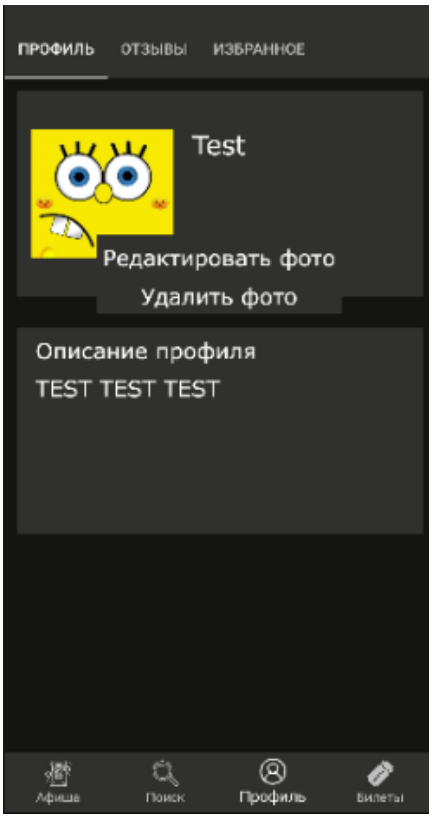


Рисунок Д.11 – Изменить изображение в профиле

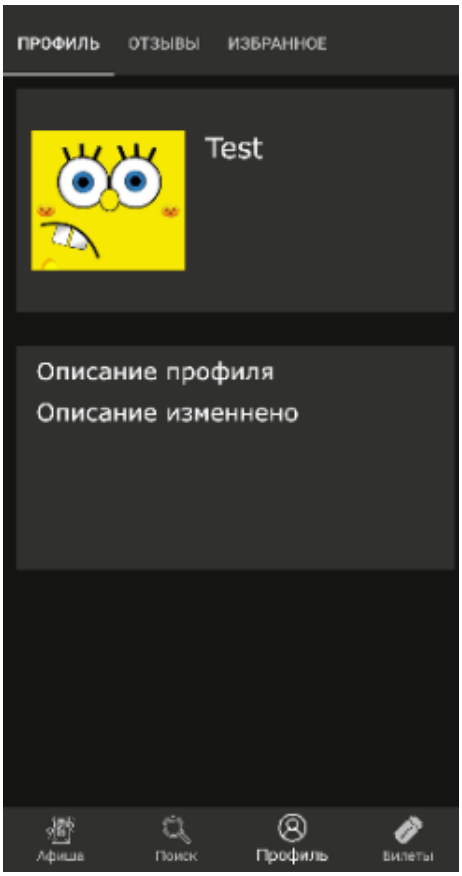


Рисунок Д.12 – Изменить описание в профиле

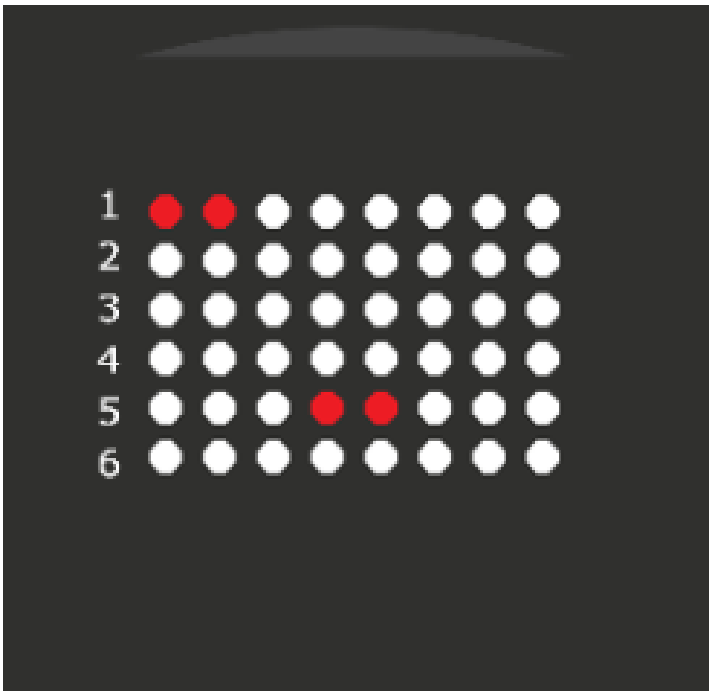


Рисунок Д.13 – Показать список свободных мест

Рисунок Д.14 – Забронировать место