

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

Факультет
«Вычислительная техника»
(наименование)

Кафедра
«Системы автоматизированного проектирования»
(наименование)

правление подготовки

02.03.03 Математическое обеспечение и АИС
(код и наименование)

офиль

Администрирование информационных систем
(наименование)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему:

Разработка мобильного приложения
для организации социальных событий и встреч
на основе интересов пользователей

удент

(подпись, дата)

Шульмин А.Н.
(фамилия, инициалы)

ководитель

(подпись, дата)

Бождай А.С.
(фамилия, инициалы)

рмоконтролёр

проф. каф. САПР
(должность, место работы)

(подпись, дата)

Бождай А.С.
(фамилия, инициалы)

бота допущена к защите

ведующий
федрой

(подпись)

Бурукина И.П.
(фамилия, инициалы)

бота защищена с отметкой отлично (протокол заседания ГЭК от 17.06.24 № 1)

кретарь ГЭК

(подпись)

Финогеев А.А.
(фамилия, инициалы)

Пенза 2024

“Утверждаю”

И.П. Бурукина/
Заведующая кафедрой САПР
“10” 06 2024г.

ЗАДАНИЕ

на выполнение бакалаврской работы

Студент гр. 20ВА1 факультета ФВТ специальности Математическое обеспечение и администрирование информационных систем

Шульмин Андрей Николаевич

(фамилия, имя, отчество)

Руководитель ВКР профессор кафедры САПР Бождай Александр Сергеевич

Время дипломного проектирования с _____ 2024 г. по _____ 2024 г.

Место преддипломной практики кафедра САПР ФВТ ПГУ

Тема бакалаврской работы: «Разработка мобильного приложения для организации социальных событий и встреч на основе интересов пользователей»

Тема утверждена приказом ПГУ № 44410 от “26” 04 2024 г.

Техническое задание на бакалаврскую работу (назначение устройства, условия применения, внешние взаимодействия, специальные требования т.п.)

6.1. Назначение разработки: Разрабатываемый программный продукт предназначен для улучшения процесса создания и организации социальных событий и встреч на основе интересов пользователей

6.2. Требования к функциональности системы:

- 1) Регистрация/авторизация пользователя в системе;
- 2) Возможность указать интересы пользователя и время его доступности;
- 3) Возможность создавать собственные встречи и события;
- 4) Возможность просмотра имеющихся встреч и событий, а также возможность изъять желание участия в них;
- 5) Создание и автоматическое объединение пользователей в группы

6.3. Требования к ПО:

Система должна содержать следующие подсистемы:

- 1) Подсистема пользователей
- 2) Подсистема групп
- 3) Подсистема локаций
- 4) Подсистема мест
- 5) Подсистема интересов

6.4. Требования к аппаратному обеспечению: Персональный компьютер; Операционная система Linux Ubuntu 20.04 или выше.

7. Содержание пояснительной записки бакалаврской работы (перечень вопросов, подлежащих разработке, расчетов, обоснований, описаний)

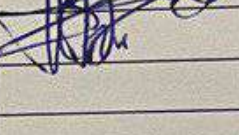
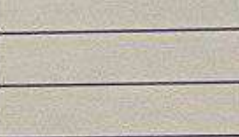
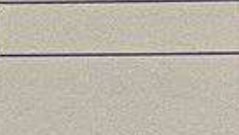
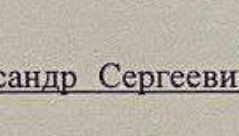
7.1. Анализ предметной области

7.2. Проектирование системы

7.3. Разработка программного обеспечения

7.4. Тестирование приложения

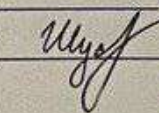
8. Календарный график работ по выполнению проекта.

Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя, консультанта
Анализ ТЗ	5 %	15.03-20.03	
Разработка структуры системы	15 %	20.03-05.04	
Разработка ИО	15 %	05.04-15.04	
Разработка ПО	30%	15.04-10.05	
Тестирование	30%	10.05-15.05	
Оформление ПЗ	5%	15.05-30.05	
Нормоконтроль		31.05	

Дата выдачи задания "15" 03 2024 г.

Руководитель дипломного проекта профессор кафедры САПР Бождай Александр Сергеевич

Задание к исполнению принял "15" 03 2024 г.

Студент 20ВА1 Шульмин Андрей Николаевич / 

Бакалаврскую работу к защите допустить

Декан ФВТ  / Л.Р. Фионова

10.06.2024

Реферат

Пояснительная записка содержит 64 страниц, 18 рисунков, 2 таблицы, 15 использованных источников и 2 приложения.

ОГРАНИЗАЦИЯ СОБЫТИЙ, СОЦИАЛЬНАЯ КОММУНИКАЦИЯ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, NODE.JS, POSTGRESQL, DART/FLUTTER.

Объект разработки – мобильное приложения для организации социальных событий и встреч на основе интересов пользователей.

Цель разработки: повысить эффективность процесса поиска единомышленников и организации оффлайн событий и встреч, учитывая индивидуальные интересы и предпочтения участников, способствуя более глубоким и качественным социальным взаимодействиям.

Задачами, которые требовалось решить для достижения цели, являются: изучение предметной области, определение требований к функциональности, проектирование и разработка базы данных, разработка дизайна интерфейса, разработка программного обеспечения, тестирование, исправление ошибок.

Для запуска клиентского приложения необходимо следующее окружение: смартфон с операционной системой Android 13(или выше) или IOS 15(или выше), а также стабильное подключение к сети интернет.

Приложение разработано с помощью средств проектирования DBeaver, СУБД PostgresSQL, платформы Node.js и фреймворка Nest.js, для создания серверной части, а также фреймворка Flutter для Dart для создания клиентской части приложения.

					ВКР.020303.12 24 81 01			
Изм.	Лист	№ докум.	Подпись	Дата	Разработка веб-приложения для улучшения системы бронирования отеля для животных. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Шульмин А.Н.						
Провер.		Бождай А.С.		05.06.24			3	
Реценз.						гр. 20BA1		
Н. Контр.		Бождай А.С.		05.06.24				
Утверд.		Бурукина И.П.		10.06.24				

Содержание

Обозначения и сокращения	6
Введение	7
1. Анализ предметной области.....	9
1.1. Методы и средства разработки.....	9
1.2. Сравнительный анализ аналогичных разработок.....	12
2. Проектирование системы	16
2.1. Инфологическое проектирование.....	17
2.1.1. Инфологическая модель данных.....	17
2.1.2. Связи.....	20
2.2. Проектирование структуры системы	20
2.3. Разработка алгоритма работы системы	28
3. Процесс разработки программного кода	35
3.1. Иерархия классов.....	35
3.2. Разработка запросов	35
Запрос на регистрацию пользователя.....	37
Запрос на создание встречи	37
Запрос приглашений на встречи.....	38
Запрос на принятие приглашения на встречу	38
4. Основные сценарии работы с приложением.....	39
Авторизация и регистрация пользователя	39
Введение пользователя	39
Список приглашений	40
Список групп пользователя.....	42
Профиль пользователя.....	43
Создания встречи	44
Админ-панель	45
Заключение.....	47
Список используемых источников	49
Приложение А. Разработанные запросы.....	51
Приложение Б. Листинг кода.....	60

Обозначения и сокращения

БД – база данных;

ПО – программное обеспечение;

СУБД – система управления базами данных;

UML (англ. Unified Modeling Language) – унифицированный язык моделирования;

JavaScript – мультипарадигмальный динамический интерпретируемый язык программирования;

Node.js – программная платформа для разработки серверных приложений на основе языка JavaScript;

Dart – язык программирования общего назначения, предназначенный прежде всего для разработки прикладных приложений;

Flutter – фреймворк с открытым исходным кодом создания мобильных приложений под Android и IOS;

SQL (англ. Structured Query Language) – это декларативный язык программирования;

Frontend – пользовательский интерфейс и его функции, направленные на взаимодействие с пользователем;

Backend – серверное приложение, предназначенное для хранения в выполнения доменной логики приложения;

Матчинг (англ. Matching) – процесс объединения пользователей, на основе одного или более критериев.

Введение

В современном мире, где социальные связи и взаимодействия играют все более важную роль в жизни людей, разработка программного продукта для организации оффлайн встреч и событий на основе интересов пользователей становится актуальной и востребованной задачей.

Развитие современных сообществ в интернете оставляет без внимания личный опыт общения между людьми. Тем не менее, существует человеческая предрасположенность к живому, офлайн, общению. Решением этой проблемы могло бы послужить приложение для организации встреч и мероприятий на основе интересов пользователей [1].

В выпускной квалификационной работе рассматривается архитектура, возможные методы организации потоков данных, хранение и управление ими в рамках решаемой задачи.

Кроме того, существующие аналоги подобных решений не всегда предлагают удобный способ для решения возникающих проблем, в частности:

- сложность поиска группы единомышленников, а также отсутствие функционала по автоматическому объединению пользователей со схожими интересами;
- отсутствие возможности по созданию собственного мероприятия, с последующим сбором группы пользователей на это мероприятие;
- отсутствие возможности интеграции с публичными заведениями в локации пользователя;
- большинство подобных приложений не реализуют функционал для создания группы пользователей более чем из 2х человек;

Исходя из вышесказанного, приложение должно реализовывать следующие функции:

- автоматизация процесса формирования групп единомышленников;
- возможность создания встреч, инициированных пользователем;
- интеграция с популярными публичными заведениями;
- охватывать наиболее популярные случаи использования;

Для достижения поставленных целей необходимо решить следующие задачи:

1. Проектирование модели предметной области.
2. Проектирование уровня сохранения данных.
 - 2.1. Логическое проектирование.
 - 2.2. Физическое проектирование.
3. Проектирование проверок целостности на уровне моделей доменной области.
4. Разработка серверной и клиентской частей приложения.

1. Анализ предметной области

Большинство сообществ в интернете не выходит за рамки онлайн общения, а наличие сообществ по интересам, как правило, направлено на общение в узкой сфере (бизнес, видео игры, романтические отношения и т.д.), не затрагивая широкий спектр иных областей и сфер человеческих интересов. Таким образом, сложность поиска единомышленников для офлайн общения неизбежно растет, а формирование так называемых информационных пузырей становится серьезной проблемой нашего времени.

Реализация социальной платформы, предоставляющей возможности гибкого поиска единомышленников и организации любых сообществ пользователей существенно бы улучшило пользовательский опыт в сфере социальных коммуникаций.

Ожидаемым результатом разработки такого продукта могло бы стать мобильное приложение, в котором пользователи могли бы легко объединяться в группы по общим интересам, встречаться в офлайн-формате, находить единомышленников, а также новые сферы собственных интересов.

Кроме того, платформа могла бы предоставлять возможность партнерства для коммерческих организаций, таких как: бары, рестораны и кафе. Их владельцы могут указать список интересов, для людей, которым было бы комфортно собраться в неформальной обстановке в их заведении.

1.1. Методы и средства разработки

Для реализации данного проекта были рассмотрены и выбраны следующие технологии для создания клиент-серверных приложений, а также проанализирована литература, посвященная данным технологиям:

Технологии для реализации серверной части приложения:

1. Node.js [8] – кроссплатформенный рантайм для JavaScript, зарекомендовавший себя как хороший инструмент для проектирования серверных приложений. Большое количество обучающих материалов обеспечивают качественную кривую обучения и приятный опыт разработки. Решение, доказавшее свою зрелость на протяжении долгого

времени, легко масштабируется и при правильном использовании показывает впечатляющую производительность, за счет использования асинхронной неблокирующей модели ввода-вывода.

2. Nest.js [9] – веб-фреймворк, разработанный поверх платформы Node.js [8]. Имеет богатый инструментарий для разработки качественных и современных серверных приложений, пропагандирует чистую и масштабируемую архитектуру. В сравнение с конкурентами показывает высокую производительность, а широкий набор внешних модулей позволяет реализовывать логику любой сложности, не прибегая к большим усилиям при разработке.
3. PostgreSQL [10] – бесплатная и open-source СУБД, известная своей стабильностью, масштабируемостью и надежностью, что очень важно для проектируемой системы. Имеет развитое сообщество и большое количество обучающих материалов. Производительность данной СУБД также является немаловажным фактором для выбора именно данной технологии.
4. Prisma ORM [13] – популярным ORM (Object-Relational Mapping) для node.js с широкими возможностями и удобным API. Полностью поддерживает TypeScript, предоставляет удобные инструменты для написания и применения миграций.

Технологии для реализации клиентской части приложения:

1. Dart [15] – язык общего назначения, применяющийся наиболее часто в сфере разработки кроссплатформенных мобильных приложений. Имеет просто для изучения синтаксис, высокую производительность и отличную документацию, позволяющую быстро разобраться со всеми особенностями данной технологии.
2. Flutter [11] – стек технологий для разработки кроссплатформенных мобильных приложений. Официальная документация Flutter [11] описывает самые лучшие практики по разработке кроссплатформенных приложений и предоставляет наиболее ценную информацию об его

возможностях. Часто приложения, написанные с помощью данной технологии, предоставляют не менее качественный опыт использования, неотличимый от нативно-разработанного приложения.

Прочие технологии:

1. JWT [12] – это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Хорошо подходит для мобильных клиентов. Прост в масштабируемости и очень хорошо защищен с точки зрения безопасности. В данном проекте эта технология необходима для реализации авторизации и регистрации пользователей.
2. Nginx [14] – веб-сервер, работающий на Unix-системах. Имеет открытую лицензию и является проектом с открытым исходным кодом. Использует асинхронную режим работы, что повышает скорость обработки запросов. Имеет встроенную систему кэширования данных, что так же позволяет повысить производительность системы.

В ходе проектирования и разработки данного проекта использовался архитектурный стиль DDD (Domain Driven Design), позволяющий разрабатывать гибкие и масштабируемые приложения. Книга “Реализация методов предметно-ориентированного проектирования” [2] описывает основную суть данного подхода, а также принципы его реализации непосредственно в коде проектируемого приложения.

Кроме всего прочего, для разработки приложения были использованы некоторые паттерны и шаблоны проектирования, описанные в книге “Чистая архитектура. Искусство разработки программного обеспечения” [3], что прежде всего положительно сказывается на чистоте кода, программных абстракций и общей архитектуре приложения.

1.2. Сравнительный анализ аналогичных разработок

В рамках разработки данной системы были рассмотрены следующие аналоги, предоставляющие похожий функционал, но не имеющие ряд критически важных функций, необходимых для реализации качественного пользовательского опыта:

InParty [4] – приложение для поиска друзей, подруг и попутчиков поблизости, в вашем городе. Приложение построено на основе поиска людей поблизости и предложений от пользователей заняться чем-то вместе. Не имеет интеграции с публичными заведениями в текущей локации, а также не используются возможности по автоматизации матчинга пользователей на основе их интересов, хотя анкета пользователя требует заполнения данного поля.

LetzTalk [5] – сервис для поиска друзей. Основан на просмотре и оценке анкет других пользователей. Не предоставляет возможность создания групп пользователей на основе их интересов, и как следствие – отсутствует автоматизированный процесс создания групп на основе пользовательских интересов. Нет интеграции с публичными заведениями в текущей локации.

Random Coffee [6] – сервис дружеских и профессиональных знакомств. Основан на чат-боте в Telegram. Так же не имеет ряд ключевых и достаточно важных функциональных особенностей, но основным минусом является отсутствие самостоятельного пользовательского интерфейса. Использование мессенджера Telegram как пользовательского интерфейса сильно ограничивает сервис и сильно уменьшает его удобство, а также доступность, в случаях, если потенциальный пользователь не имеет аккаунта в Telegram.

VK Dating [7] – сервис знакомств. Основан на просмотре и оценке анкет других пользователей. Очень сильно фокусируется на романтических знакомствах и фактически не дает возможности для иного варианта использования. Имеет функционал по автоматического матчинга пользователей, но фокусируется на романтических знакомствах, а также не несет в себе никакого функционала по организации встреч.

В таблице 1 представлено сравнение с аналогами разработанного приложения:

Таблица 1 – Аналоги разрабатываемого приложения

Сравниваемые характеристики	Аналоги				Разработанное приложение
	InParty	LetzTalk	Random Coffee	VK Dating	Flockspot
Поиск пользователей по интересам	+ На основе просмотра анкет или людей поблизости	+ На основе просмотра анкет	+ Случайно, основываясь на интересах пользователей	+ На основе просмотра анкет	+ Автоматически на основе интересов или уже созданных групп/встреч пользователей поблизости
Фильтрация пользователей по геолокации	+	+	+	+	+
Организация пользователей в группы	+ Группы формируются автоматически и или по запросу пользователя	- Нет возможность создания групп	+ Группы формируются по запросу пользователя	- Нет возможность создания групп	+ Группы формируются автоматически или по запросу пользователя
Автоматический (фоновый) поиск пользователей со схожими интересами	+ Пользователи со схожими интересами объединяются в группы	-	-	-	+ Пользователи со схожими интересами объединяются в группы и им сразу предлагается место и время для встречи

Продолжение Таблицы 1

Сравниваемые характеристики	Аналоги				Разработанное приложение
Интеграция с заведениями в городе	-	-	-	-	+ Владельцы мест могут регистрировать свои заведения и предлагать их для посещения
Автоматическое (фоновое) создание новых встреч в созданной группе	-	-	-	-	+ В уже созданных группах пользователям предлагается встретиться повторно, спустя некоторое время после последней встречи
Возможность указать желаемое время для встреч и социальной активности	-	-	-	-	+

Исходя из сравнительного анализа, приведенного в таблице, аналоги обладают рядом недостатков, в частности связанных с процессом поиска группы единомышленников и последующей организации встречи с ними:

В частности, можно выделить следующие недостатки, сильно ухудшающие пользовательский опыт:

- Утомительный процесс оценки анкет других пользователей.

- Отсутствие функционала по организации пользователей в группы по интересам, в особенности касается групп более двух человек.
- Отсутствие интеграции с заведениями в текущей локации.
- Плохая автоматизация процессов. В том числе, поиска единомышленников в фоновом режиме и возможности системы предложить уже существующим группам увидеться снова после небольшого перерыва.
- Отсутствие возможности указать желаемое время, когда пользователь свободен и доступен для предложений о встречах.

Таким образом, существует ряд улучшения, особенно в части пользовательского опыта и функциональных возможностей, использования автоматизации для более комфортного и быстрого поиска единомышленников.

Исходя из вышесказанного, можно утверждать, что разработка такого приложения в действительности является важной и востребованной задачей, решение которой потенциально имеет высокие шансы пользоваться большим спросом со стороны конечных пользователей.

2. Проектирование системы

Стадия проектирования является одним из самых важных этапов в разработке приложения. Она определяет дальнейшую жизнеспособность проекта, правильно спроектированная система легко расширяема и дополняема новым функционалом, в тоже время, некачественное проектирование может сильно усложнить процесс дальнейшей разработки.

Для проектирования системы были выбраны методологии DDD (Domain Driven Design) [2] и Clean Architecture (Чистая архитектура) [3], призванные упростить процесс проектирования, в следствие чего был выделен список основных сущностей, необходимый для функционирования системы:

1. Пользователь (User) – пользователь системы. Может иметь роли: User (обычный пользователь), PlaceOwner (владелец места) и Administrator (Администратор).
2. Интерес (Vibe) – интерес, доступный для указания пользователями системы. Каждый интерес является уникальным в рамках системы (см. «проектирование инфологической модели») и обеспечивает возможность корректного матчинга пользователей на его основе.
3. Локация (Location) – географическая область (регион, город, район), описывающий текущее местонахождение пользователя/места
4. Доступность (Availability) – описывает временной интервал доступности пользователя или места.
5. Группа (Flock) – группа пользователей, объединенных по интересам, времени доступности, локациям. Может быть сформирована как автоматически, так и вручную пользователями системы.
6. Встреча (Meet) – созданная для группы пользователей встреча, описывающая место, время и характер времяпрепровождения. Может быть сформирована как автоматически, так и вручную пользователями системы.
7. Место (Place) – публичное место (ресторан, кафе, кинотеатр), доступное для посещения пользователями в рамках встреч.

Графическое представление описанных сущностей будет приведено в главе 3, рисунок 12, в виде UML-диаграммы классов.

2.1. Инфологическое проектирование

2.1.1. Инфологическая модель данных

На основе приведенного выше списка сущностей была разработана следующая инфологическая модель:

Предметная область: организация социальных событий и встреч на основе интересов пользователей.

Назначение системы: система, предназначенная для хранения информации о пользователях, их интересах, времени доступности, локациях, группах, встречах и публичных местах, для проведения мероприятий.

Объекты, подлежащие хранению в базе данных:

- пользователи;
- время доступности пользователя;
- интересы;
- группы;
- встречи;
- локации;
- места;
- время доступности места (расписание его работы);

На основе требований, приведенных выше, была разработана следующая инфологическая модель (рисунок 1) и описана в таблице 2.

Таблица 2 – Сущности и атрибуты

Сущность	Атрибуты	Описание
User	id, email, role, firstname, lastname, birthdate, password, nickname, avUpdated, avMimetype, avOriginal, avThumbnail, locationId, bgMimetype, bgOriginal, bgThumbnail	Таблица, хранящая всю необходимую информацию о пользователе. Атрибут email – имеет ограничение на уникальность.
Place	id, title, address, isPublic, isApproved, description, avUpdated, avMimetype, avOriginal, avThumbnail, ownerId, locationId	Таблица, хранящая информацию о местах.
Flock	id, earliest, name, bgOriginal, bgThumbnail, chatId, locationId, type	Таблица, хранящая информацию о группах. Атрибут name имеет ограничение на уникальность.
Meet	id, status, startAt, address, flockId, placeId, description	Таблица, хранящая всю необходимую информацию о встречах.
Location	id, name, bgOriginal, bgThumbnail.	Таблица, хранящая всю необходимую информацию о локациях.
(User/Place) Availability	id, day, startAt, endAt, dateStart, dateEnd, userId/placeId	Таблицы, хранящие информацию о времени доступности пользователя/места. Атрибут day представляет собой перечисление дней.

Таким образом, спроектированная структура базы данных выполняет поставленные перед ней задачи.

2.1.2 Связи

Связи между сущностями инфологической модели:

- «User» – «Flock» – m:m (через промежуточную таблицу FlockUsers).
- «User» – «Vibes» – m:m (через промежуточную таблицу _UserToVibes).
- «User» – «Meet» – m:m (через промежуточную таблицу MeetUsers).
- «Flock» – «Vibes» – m:m (через промежуточную таблицу _FlockToVibes).
- «Location» – «Vibes» – m:m (через промежуточную таблицу LocationToVibes).
- «Place» – «Vibes» – m:m (через промежуточную таблицу _PlaceToVibes).
- «User» – «UserAvailability» – 1:m (идентифицирующая).
- «User» – «Place» – 1:m (неидентифицирующая).
- «User» – «Location» – m:1 (неидентифицирующая).
- «Place» – «Location» – m:1 (идентифицирующая).
- «Place» – «PlaceAvailability» – 1:m (идентифицирующая).
- «Place» – «Meet» – 1:m (неидентифицирующая).
- «Flock» – «Location» – m:1 (идентифицирующая).

2.2. Проектирование структуры системы

Дробление на подсистемы уменьшить сложность разрабатываемой системы и выделить основные структурные части системы. В ходе разработки данного проекта были выделены следующие подсистемы и их функции:

1. Подсистема пользователей.

- 1.1. Управление списком интересов пользователя.
- 1.2. Управление временем доступности пользователя.
- 1.3. Иные возможности по редактированию профиля.

2. Подсистема групп.

- 2.1. Автоматическое создание групп и встреч.
- 2.2. Управление списком участников группы.

- 2.3. Управление списком встреч в рамках группы.
- 3. Подсистема мест (запись доступна только администратору места).
 - 3.1. Создание и обновление информации об месте.
 - 3.2. Управление временем работы места.
 - 3.3. Управление списком интересов.
- 4. Подсистема авторизации и регистрации.
 - 4.1. Авторизация пользователей.
 - 4.2. Регистрация пользователей.
 - 4.3. Предоставление и проверка прав пользователей.
- 5. Подсистема локаций (доступно только администраторам).
 - 5.1. Управление информацией о локациях.
- 6. Подсистема интересов (доступна только администраторам).
 - 6.1. Управление информацией о интересах.

Стоит отметить, что подсистемы локаций и интересов доступны только администратору, обычный пользователь в праве только предложить создание нового интереса или локации в системе. Предложение может быть рассмотрено администратором и будет принято или отклонено.

Общая архитектура разрабатываемого программного решения представлена на рисунке 2 и представляет собой клиент-серверное приложение.

Серверное приложение имеет интеграции с некоторыми внешними системами, такими как:

1. Сервисом геокодирования:

Необходим для реализации корректного ввода адресов, предоставляет API для приведения строкового представления адреса к виду географической координаты.
2. Объектным хранилищем (S3):

Необходим для хранения статических объектов, которые могут быть представлены в виде файлов. В реализуемой системе используется для хранения загруженных пользователем фотографий и прочих файлов.

Клиентское приложение также разделено на подсистемы, ниже приведен их список вместе с функциями, которые они реализуют:

1. Подсистема сетевых запросов:

Формирование и отправка сетевых запросов, а также обработка ответов.

2. Подсистема отображения:

Формирование и отображение на экране пользователя информации в виде пользовательского интерфейса.

3. Подсистема локального хранилища:

Предназначена для хранения локальных данных, необходимых для работы системы. В реализуемом приложении к таким данным, например, относятся токены доступа (JWT Tokens [12]), приложение хранит их с момента авторизации пользователя в системе до момента инициации выхода. Каждый сетевой запрос подписывается токеном для предоставления информации об аутентифицированном пользователе.

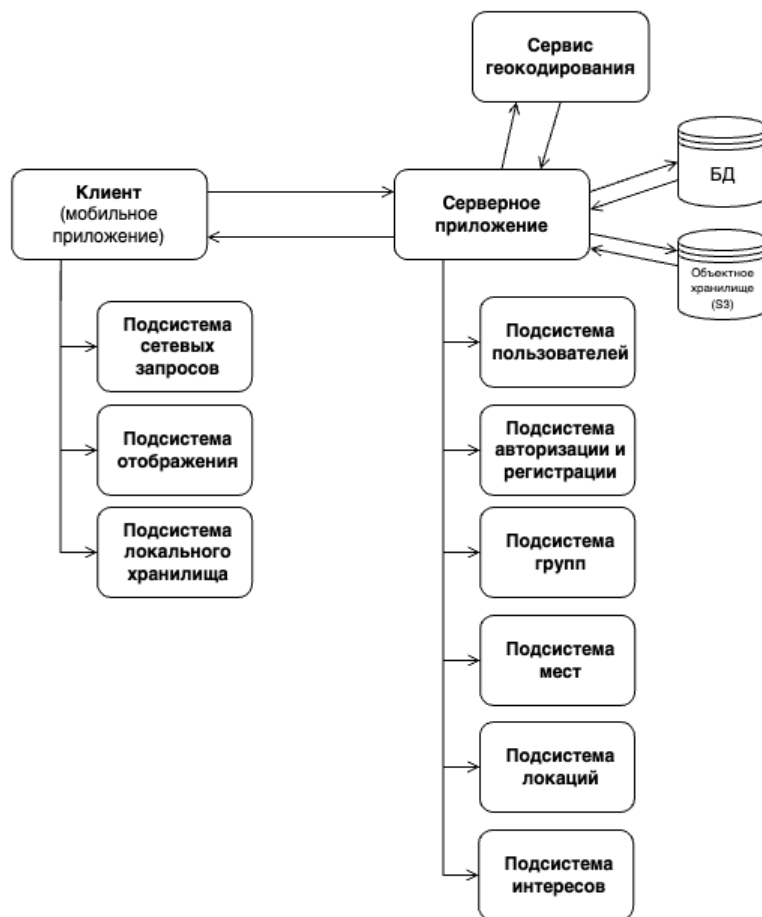


Рисунок 2 – Архитектура программного решения

На рисунке 3 приведены различные варианты использования системы:

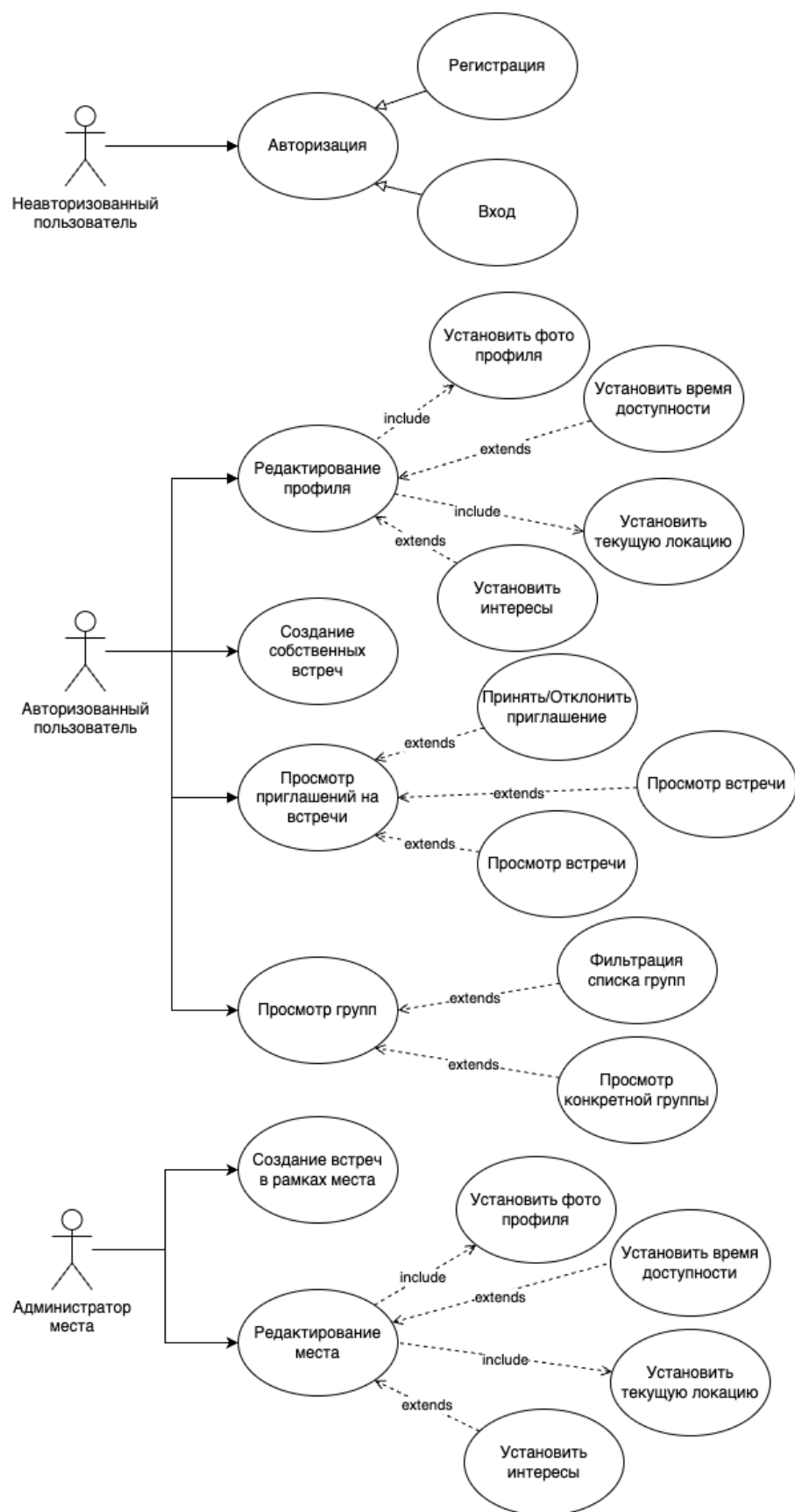


Рисунок 3 – Варианты использования системы

Диаграммы последовательностей, представленные ниже, описывают процесс выполнения некоторых операций в системе (рисунки 4-7).

Авторизация и регистрация пользователя в системе необходима для реализации профиля, который в свою очередь обеспечивает возможность указания пользователем его предпочтений (текущая локация, интересы, возможное время для встреч), на основе которых будет осуществляться подбор других пользователей. Эта важная часть приложения, описывается диаграммами последовательностей (рисунки 4-5):



Рисунок 4 – Авторизация пользователя в системе

Авторизация пользователя производится по электронной почте, пароль хранится в базе данных в зашифрованном виде для обеспечения безопасности, следовательно, для проверки верности введенного пользователем пароля необходимо выполнить операцию сверки хранящегося в базе данных хэша с результатом хэширования введенного пользователем пароля.

Сразу после успешной авторизации серверное приложение вернет соответствующий результат, а клиентское приложение отобразит для пользователя основную (главную страницу) приложения:

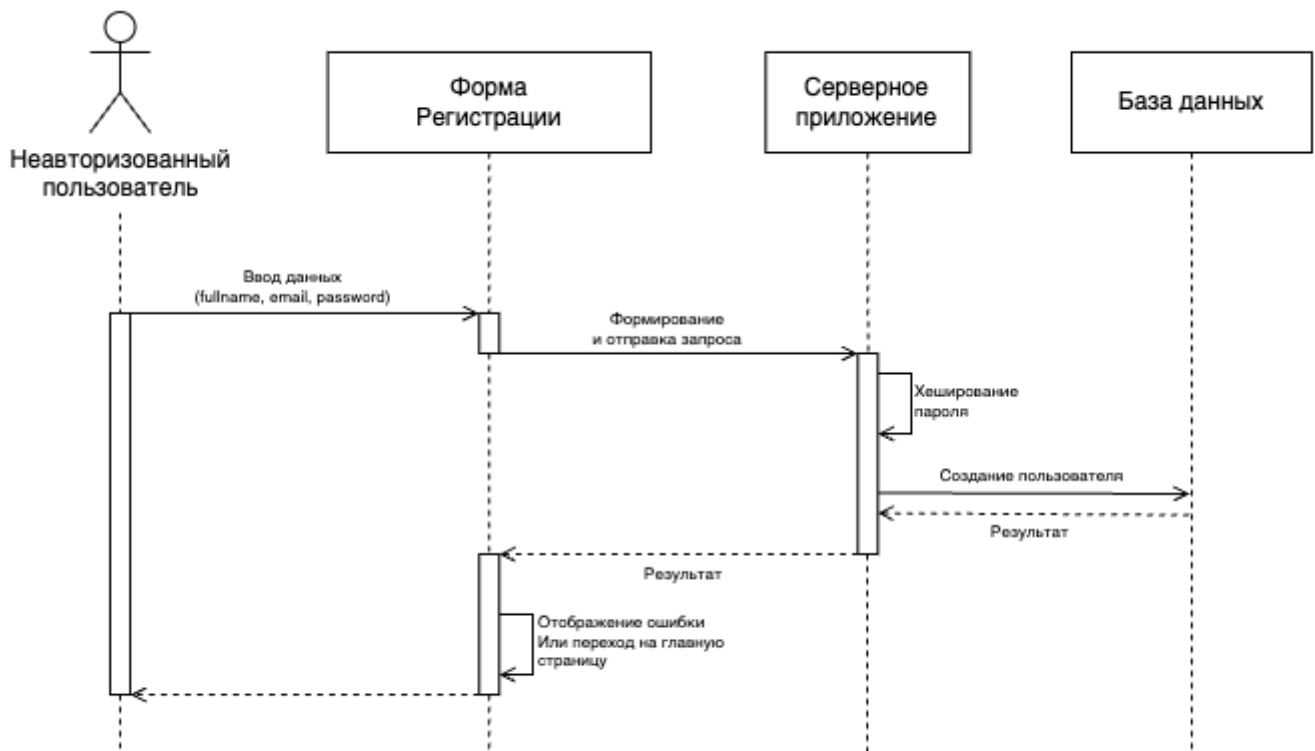


Рисунок 5 – Регистрации пользователя в системе

Регистрация пользователя также производится, основываясь на электронной почте и пароле, также пользователь должен указать полное имя. База данных и логика серверного приложения реализует ограничение на уникальность введенной пользователем электронной почты.

Кроме того, как и упоминалось выше, пароли в базе данных хранятся в захешированном виде для обеспечения безопасности. Процесс хэширования пароля происходит именно на этом этапе.

Несмотря на то, что система реализует автоматическое формирование групп пользователей и создание встреч по их интересам, нельзя отнимать возможно пользователя вручную создать встречу. В данном варианте использования пользователь должен указать некоторую обязательную информацию о желаемой встрече: желаемая дата, адрес или место проведения встречи, а также набор интересов, пользователям которых потенциально могли бы заинтересоваться и присоединиться к данному мероприятию.

Диаграмма последовательностей, описывающая этот процесс (рисунок 6):



Рисунок 6 – Мануальное создание встречи пользователем

Данная функция является одной из основных в системе, поэтому ей уделено особое внимание. Для обеспечения корректности ввода данных пользователем используется сервис геокодирования, в случае указания им адреса, а в случае выбора места – проверка наличия данного места в системе, а также совпадения его локации с локацией пользователя, потому как нельзя допускать создание встречи пользователем вне его локации.

Указанный при создании список интересов будет использован для рекомендации данного события другим пользователям, которые в свою очередь, могут принять приглашение в участие или отменить его.

Кроме того, при создании встречи, выполняется создание группы, содержащей, пока, только данного пользователя. В последствии, пользователь может поделиться ссылкой на вступление в эту группу в других социальных сетях.

Поскольку система автоматически создает группы и встречи для пользователей со схожими интересами, необходим механизм приглашений с возможностью принятия и отмены. Данный механизм продемонстрирован на рисунке 7:

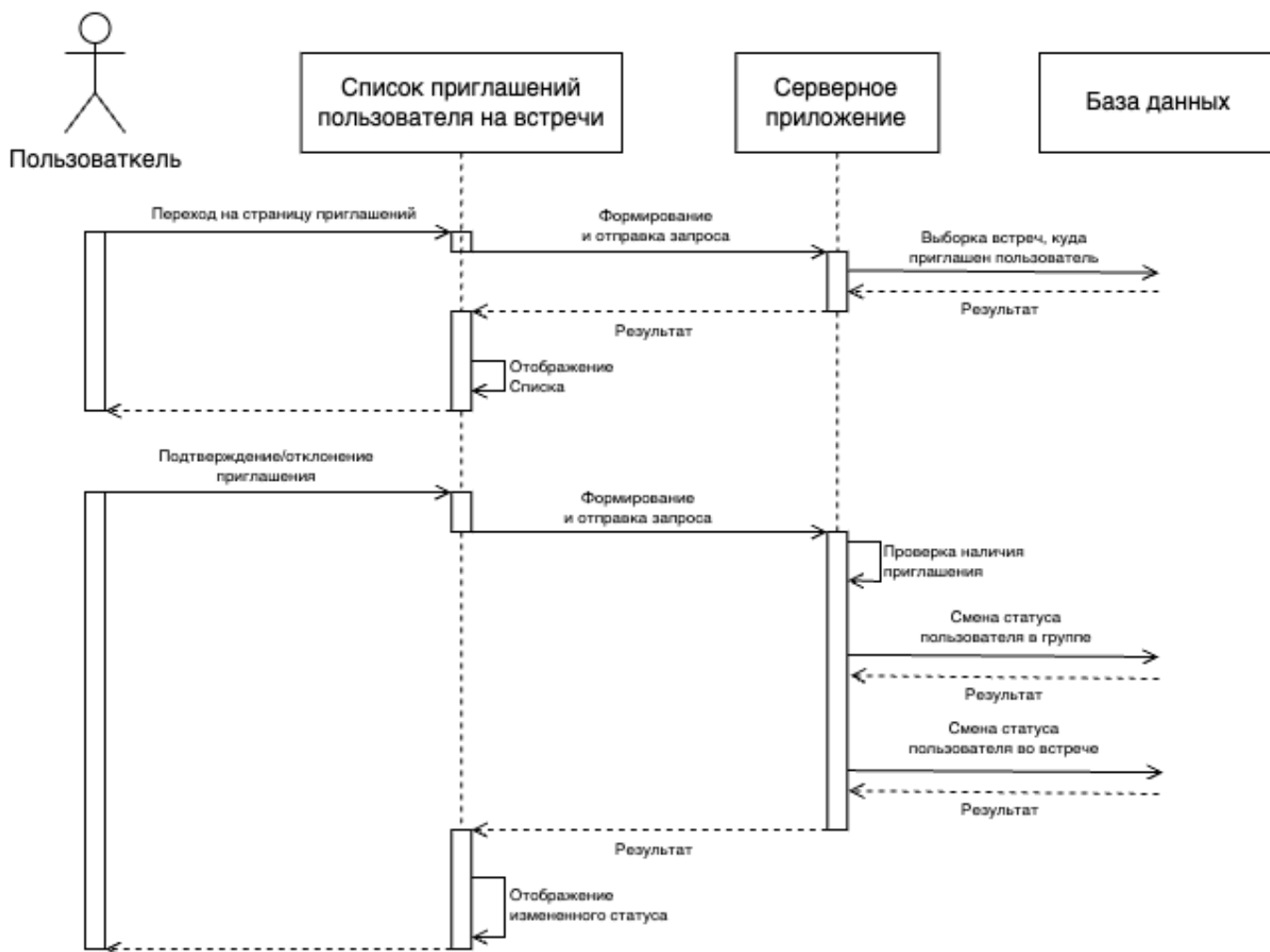


Рисунок 7 – Принятие и отмена приглашений

Важным нюансом данного процесса является то, что при принятии приглашения на встречу, пользователь добавляется и в группу, для которой эта встреча была создана. Это гарантирует возможность связи пользователям после проведения встречи, сформированная группа пользователей может быть удалена из системы только при выходе из нее всех пользователей. Это обеспечивает лучший пользовательский опыт в коммуникации с людьми уже после состоявшейся встречи.

Подводя итог, можно заключить, что система предоставляет различный функционал и обеспечивает пользователям возможность найти и вживую встретиться со своими единомышленниками, а модульная структура системы позволит, при надобности, масштабировать ее и дополнять новым функционалом.

2.3. Разработка алгоритма работы системы

В данном разделе рассматриваются основные алгоритмы работы системы, их разработка является важным этапом, поскольку обеспечивает работоспособность системы.

Вероятно, первое посещение приложения является самой важной частью в формировании пользовательского мнения о продукте, поэтому оно является важной частью, требующей большого внимания к организации работоспособности и формированию позитивного пользовательского опыта.

После авторизации или регистрации пользователя в системе он попадает на главный экран, в случае если он уже существовал в системе, в противном же случае – он попадает на экран введения. Данный экран необходим, чтобы пользователь указал основную информацию о себе: текущую локацию, фотографию профиля, список интересов и время доступности. Необходимость этой информации обуславливается потребностью иметь достаточно широкий набор информации о пользователе, так как она будет в дальнейшем использована в процессе автоматического формирования групп и встреч для пользователя. Не пройдя данный этап, пользователь не сможет двигаться дальше и получить полноценный доступ до всего функционала системы.

После прохождения всех обязательных этапов пользователь, наконец, получает полный доступ до основного функционала приложения. С данного момента пользователю доступны основные разделы приложения: профиль пользователя, список его групп, приглашений на встречи, а также раздел, позволяющий пользователю создать встречу в мануальном режиме.

Блок схема, отображающая процесс входа в приложение приведена на рисунке 8:

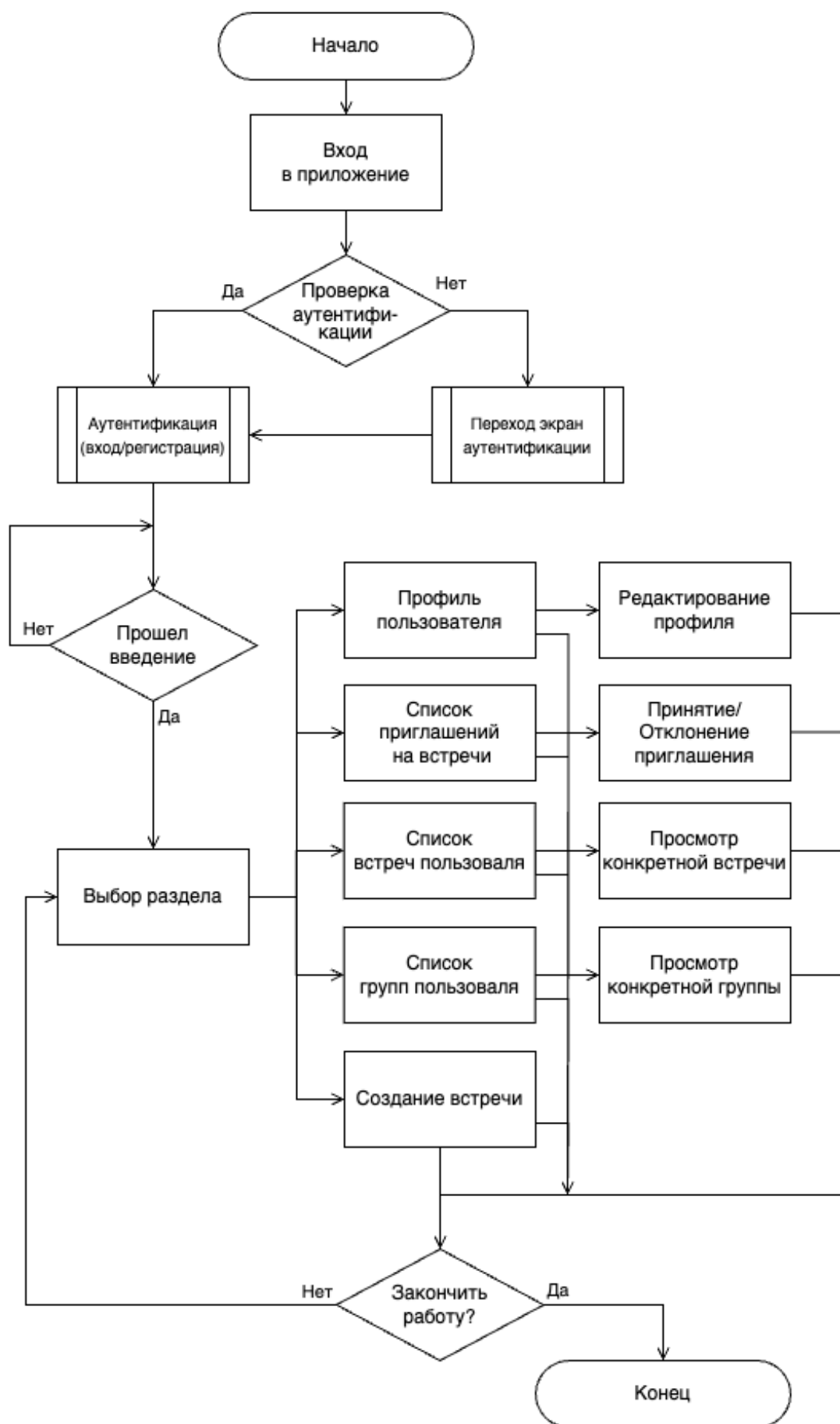


Рисунок 8 – Блок-схема алгоритма работы системы со стороны пользователя

Как упоминалось выше, в системе реализован механизм автоматического объединения пользователей в группы, на основе их интересов, а также автоматическое создание встреч внутри групп. Реализующий данный процесс алгоритм показан на рисунке 9:

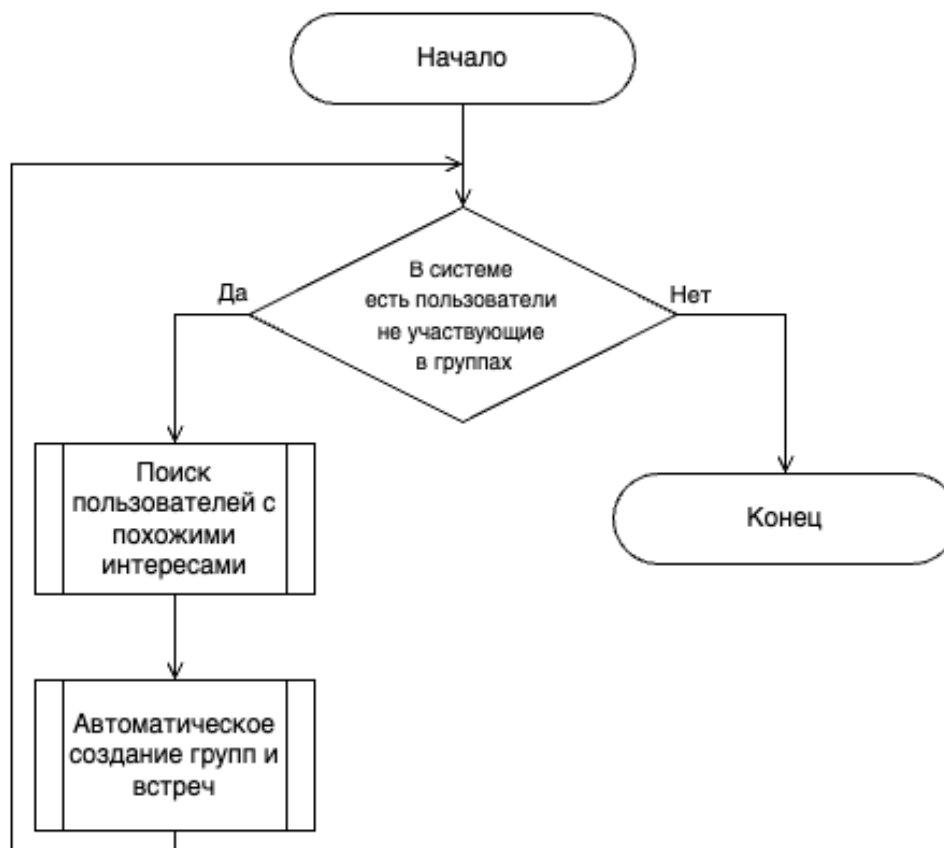


Рисунок 9 – Блок-схема алгоритма автоматического объединения пользователей в группы

Имея фоновый процесс для создания групп пользователей, было бы хорошо иметь и фоновый процесс, предназначенный для создания новых встреч для групп, не имеющих в данный момент времени активной встречи.

Данный процесс позволил бы созданным когда-то группам пользователей не терять активность и продолжать видаться друг с другом на встречах. Подразумевается, что данный фоновый процесс будет запускаться раз в 1-2 недели, чтобы напомнить людям друг о друге, и предложить встретиться вновь.

Алгоритм, демонстрирующий работу такого процесса представлен на рисунке 10:

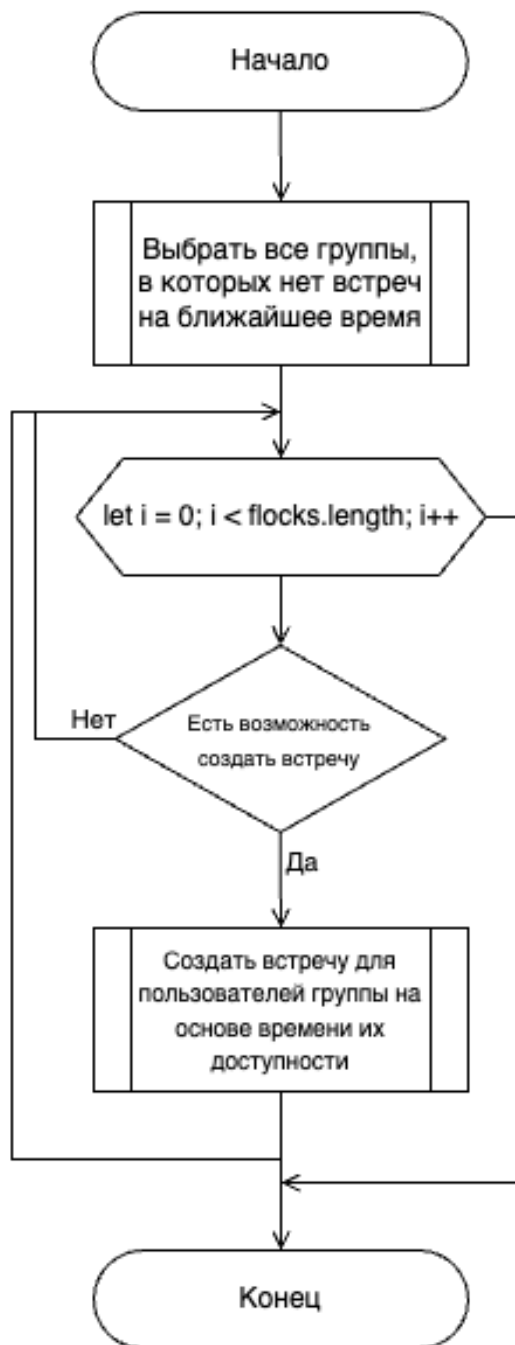


Рисунок 10 – Алгоритм фонового создания встреч

Как уже и упоминалось выше, одним из критических плюсов системы, по сравнению с аналогами, является наличие алгоритма автоматического матчинга пользователей по их интересам.

Основная суть данного алгоритма заключается в следующем:

1. Найти пользователя с максимальным количеством интересов – число его интересов будет влиять на количество запусков цикла поиска совпадающих интересов среди всех пользователей.

2. В каждом запуске внешнего цикла (количество итераций которого зависит от числа, полученного на предыдущем шаге) проходить всех пользователей и сравнивать их попарно, если число совпавших интересов равно текущему уровню. Для каждого из таких попарных сравнений записывать пользователя в промежуточную переменную.
3. Отфильтровать пользователей, уже являющихся участниками группы, которая имеет такой же набор интересов, как и полученный на предыдущем шаге набор совпавших интересов для двух попарно сравниваемых пользователей.
4. Декрементировать значение текущего уровня.

Под уровнем понимается некоторое целое число, обозначающее размер пересечения интересов двух пользователей, что необходимо для обеспечения как можно более вариативного разбиения пользователей на группы. Уровень декрементируется на каждой итерации цикла, таким образом исключается возможность, во-первых, добавлять пользователей в группы, интересы которых являлись бы подмножеством интересов какой-то другой конкретной группы, в которой пользователь уже является участником, а во-вторых, избежать ситуации, в которой пользователи раз за разом попадают в одни и те же группы, содержащие примерно одинаковые интересы.

Таким образом, будет получен список пользователей, разбитых на уникальные, не повторяющие друг друга группы. Набор интересов в каждой группе, по которым велась группировка всегда будет уникальным, а количество этих интересов, определяться значением переменной `level`, принимающей значения `[GetMaxVibesCount(users), ..., 1]`.

Настало время раскрыть эту тему подробнее. На рисунке 11 приведена блок-схема алгоритма, реализующего разбиение пользователей на группы по интересам.

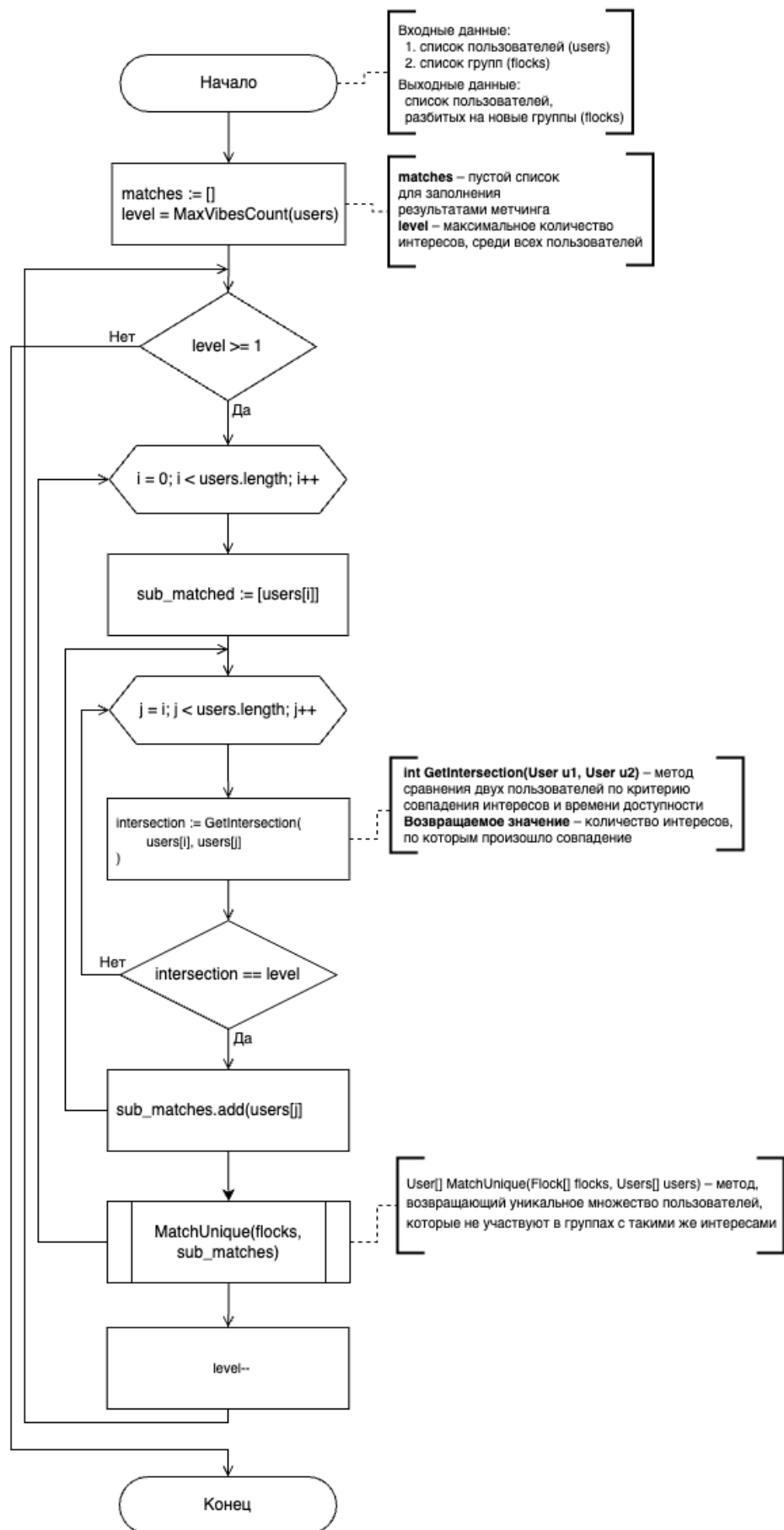


Рисунок 11 – Алгоритм матчинга пользователей по интересам

Таким образом, были спроектированы и реализованы основные алгоритмы, обеспечивающие работоспособность системы. В дальнейших главах будет рассмотрен процесс их практического применения и приведены исходные коды описанных алгоритмов.

3. Процесс разработки программного кода

Как и упоминалось ранее, для проектирования данной системы были выбраны 2 популярные методологии разработки программного обеспечения: DDD (Domain Driven Design) [2] и Clean Architecture (Чистая архитектура) [3]. Придерживаться этих методологий будем и в ходе процесса непосредственной разработки системы.

3.1. Иерархия классов

Для реализации поставленных задач и непосредственной формализации их в программном коде, необходимо четко представлять структуру и взаимосвязи компонентов системы.

Для решения этой задачи хорошо подойдет диаграмма классов. На представленной на рисунке 12 диаграмме классов показаны все необходимые сущности системы и их взаимоотношения в виде программных классов.

Продемонстрированная диаграмма классов представляет собой полный набор необходимых для работы приложения сущностей и демонстрирует взаимосвязи между ними.

3.2 Разработка запросов

В данной части пояснительной записки будет рассмотрен пошаговый процесс выполнения некоторых запросов, определенных в системе. Первым этапом запрос всегда формируется в виде http-запроса, посредством сетевого взаимодействия запрос направляется на серверное приложение, где он будет обработан, тело запроса будет провалидировано на предмет несоответствий заявленной схеме, после чего преобразован в SQL код, который в свою очередь будет выполнен системой управления базы данных.

Результат выполнения SQL запроса будет передан обратно клиентскому приложению в формате JSON, которое в свою очередь выполнит ряд действий, заложенных в логике функционирования или же будет выполнено отображение полученных данных. Приведенные ниже запросы более конкретно рассмотрены в приложение А в виде http/sql запросов и приведены http-ответы в формате JSON.

Таким образом, разработанная иерархия классов полностью удовлетворяет требованиям и позволяет обеспечить работоспособность системы.

Запрос на регистрацию пользователя

Запрос принимает информацию, введенную пользователем на экране регистрации (см. глава 4 – регистрация пользователя).

Обязательными полями для формирования запроса являются:

- адрес электронной почты: верного формата и уникальный в рамках системы;
- имя пользователя, разделенное пробелом;
- пароль имеет длину, не менее 8 символов;

Ответом на данный запрос будет объект в формате JSON, содержащий информацию о созданном пользователе, а также пару JWT токенов, используемых в дальнейшем для аутентификации запросов пользователя.

Запрос на создание встречи

Запрос принимает информацию, введенную пользователем на экране создания встречи (см. глава 4 – создание встречи).

Запрос принимает следующие поля:

- название: уникальное в рамках системы;
- описание;
- адрес: существующий (проверка осуществляется через сервис геокодирования);
- дата и время: позднее текущего момента;
- идентификатор выбранного места;
- список идентификаторов интересов;

Необходимо отметить, что часть данных, необходимых для выполнения данной операции берутся не из тела запроса, а непосредственно из базы данных (профиля пользователя), что упрощает запрос с клиентской стороны и уменьшает количество передаваемых клиентом данных

Запрос приглашений на встречи

Запрос использует постраничное чтение, поэтому принимает параметры смещения и количества вычитываемых элементов.

Ответом на данный запрос будет объект в формате JSON, содержащий список встреч, включая связанные с ними группы, в которые пользователь был приглашен. Результат данного запроса отображается на экране списка приглашений на встречи (см. глава 4 – список пользовательских приглашений).

Запрос на принятие приглашения на встречу

Запрос принимает идентификаторы встречи и группы, результатом выполнения данного запроса является смена статуса пользователя в базе данных, что в свою очередь изменит визуальное представление карточки приглашения на главном экране приложения.

В ответ на данный запрос клиентское приложение получает объект в формате JSON, содержащий только флаг, обозначающий успешное выполнение, после чего мобильное приложение должно обновить информацию о приглашении и списке групп пользователя.

4. Основные сценарии работы с приложением

Рассмотрим основные сценарии работы пользователя с системой. В данном разделе будут приведены скриншоты экрана мобильного приложения и описана последовательность действий пользователя для выполнения определенных действий.

Авторизация и регистрация пользователя

В первую очередь, пользователь должен авторизоваться или зарегистрироваться в приложение, для продолжения работы. Данный шаг будет пропущен, при последующих посещениях приложения, так как система запомнит введенные им данные, но при первом посещении эта стадия является обязательной.

Для успешной авторизации пользователь должен ввести свои данные и нажать на кнопку «Войти». В случае ввода некорректных данных (неправильный формат электронной почты, слишком короткий пароль, или же ввода почты, пользователь которой уже существует в системе) пользователю отобразится ошибка, а поля формы будут сброшены в исходное состояние.

Для успешной регистрации пользователь должен ввести свое имя и фамилию, разделенные пробелом, а также адрес электронной почты и пароль. Как и в случае с авторизацией, при обнаружении некорректного ввода или присутствия в системе пользователя с таким же адресом электронной почты, пользователю будет показано сообщение об ошибке, а поля формы сброшены в исходное состояние. Сразу после успешной регистрации пользователь попадет на экран введения.

Введение пользователя

Введение пользователя необходимо для заполнения основной информации о пользователе:

- фото пользователя;
- список интересов;
- время, когда пользователь доступен для встреч;
- также текущая локация;

Как упоминалось ранее, большая часть приведенной выше информации необходима для корректного матчанга пользователей, следовательно, данный шаг

должен являться обязательным, во избежание ситуаций, когда пользователь зарегистрирован в системе, но не получает никаких автоматических приглашений, так как не установил никакой информации о себе.

После прохождения регистрации пользователь автоматически будет направлен на данный экран, а также, в случаях, когда пользователь не указал необходимой информации сразу, после успешной авторизации.

Шаг введения с установкой фото профиля необходим для указания пользователем фото профиля, помимо этого, пользователь может установить и задний фон для его профиля. Задний фон профиля не является обязательным для заполнения полем, но он будет виден всем пользователям, посещающим профиль данного пользователя, что является хорошей возможностью подчеркнуть свою индивидуальность.

После, следует шаг выбора текущей локации. Пользователь может установить текущую локацию простым нажатием на желаемый элемент, после чего кнопка «Сохранить станет активной» и он сможет перейти на следующий шаг введения.

Следующим этапом пользователь должен выбрать интересы, из списка доступных. Как и на предыдущем шаге, пользователь может нажатием выбрать интересующие его элементы после чего нажать на кнопку «Сохранить». Единственное отличие заключается в возможности множественного выбора.

Последним, что потребуется от пользователя во введение – установка времени доступности. Этот шаг подразумевает, указание пользователем времени, когда он свободен, следовательно, может принять приглашения на встречи. После добавления одной или нескольких записей о времени и нажатия кнопки «Добавить» пользователь наконец-то попадет на главный экран приложения.

Список приглашений

На данном экране показаны все приглашения пользователя на встречи, включая приглашения, которые пользователь ещё не принял, и уже принятые.

Карточки принятых приглашений не имеют внизу кнопок для принятия/отклонения приглашения, а непринятые имеют и могут быть приняты или отклонены на этом же экране.

На рисунке 13 показано, как выглядит данная часть приложения:

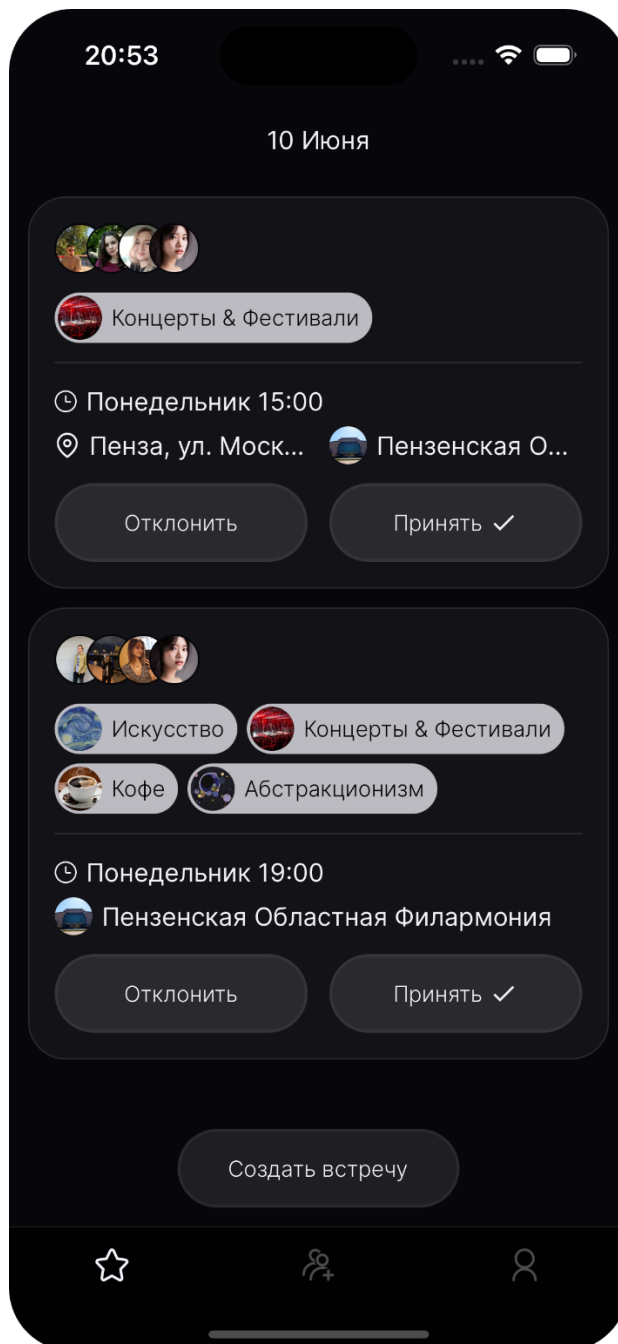


Рисунок 13 - Приглашения пользователя

После нажатия на одну из кнопок «Принять»/«Отклонить» будет отправлен соответствующий запрос на сервер и карточка пропадет из списка, в случае отклонения приглашения, а в случае принятия кнопки внизу карточки пропадут.

Список групп пользователя

Список групп пользователя отображает все группы, в которых пользователь является участником. На этом экране пользователь может увидеть основную информацию по его группам. Из него же можно попасть на экран конкретной группы, чтобы посмотреть более детальную информацию о ней.

На рисунке 14 показан вид экрана списка групп:

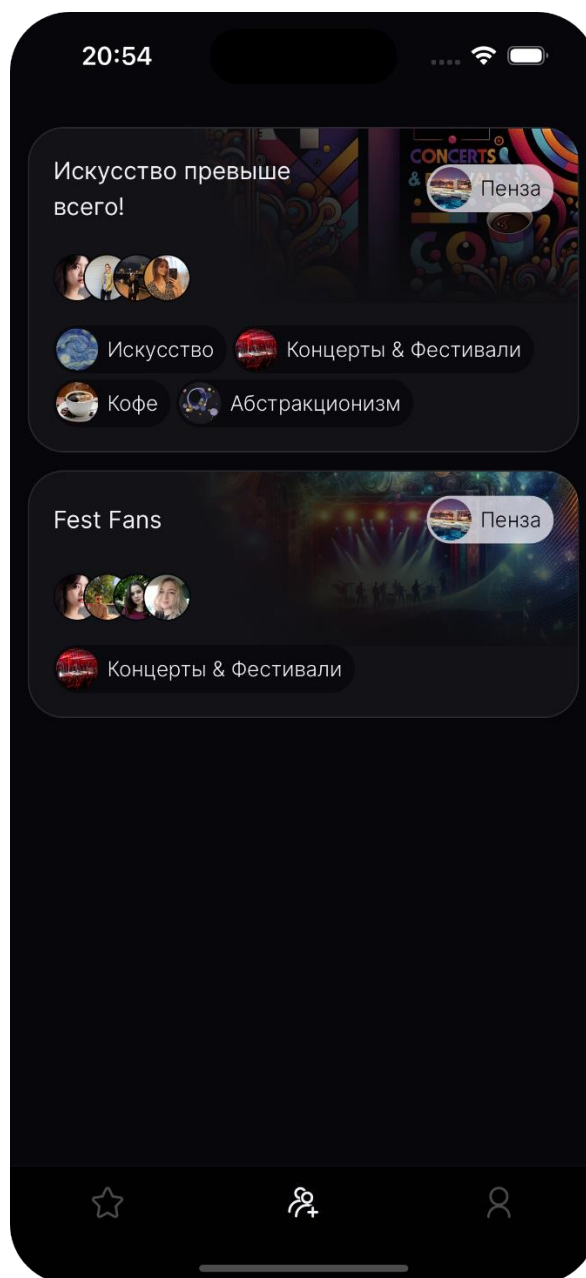


Рисунок 14 – Список групп пользователя

Основной информацией о группе является: список ее участников, список интересов группы, ее название, а также обложка, если она имеется.

Профиль пользователя

На рисунке 15 показан вид экрана профиля пользователя:

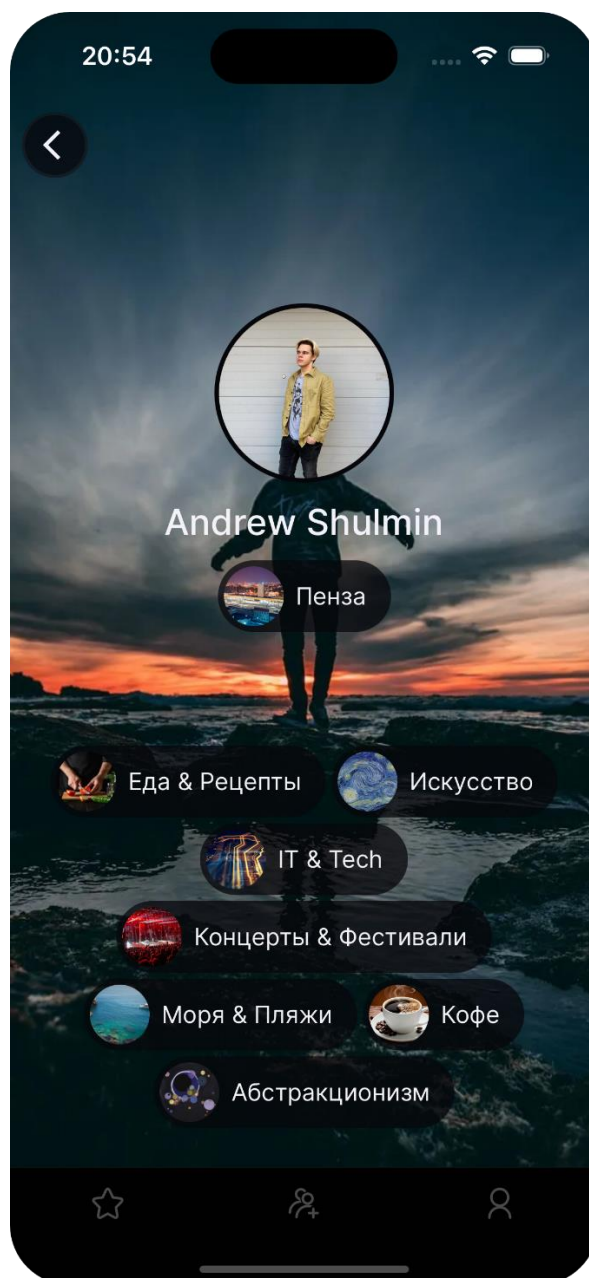


Рисунок 15 – Профиль пользователя

На данном экране пользователь видит основную информацию, которую указывал при введении. При отсутствии, указанного пользователем, заднего фона в виде изображения не будет, а фон будет залит основным цветом для фона.

Если просматривающий в данный момент профиль пользователя является его владельцем – отображается кнопка выхода из приложения. По нажатию на нее пользователь попадет обратно на экран авторизации. В этом же разделе пользователь может отредактировать основную информацию о себе.

Создания встречи

На рисунке 16 показан экран создания встречи от лица пользователя:

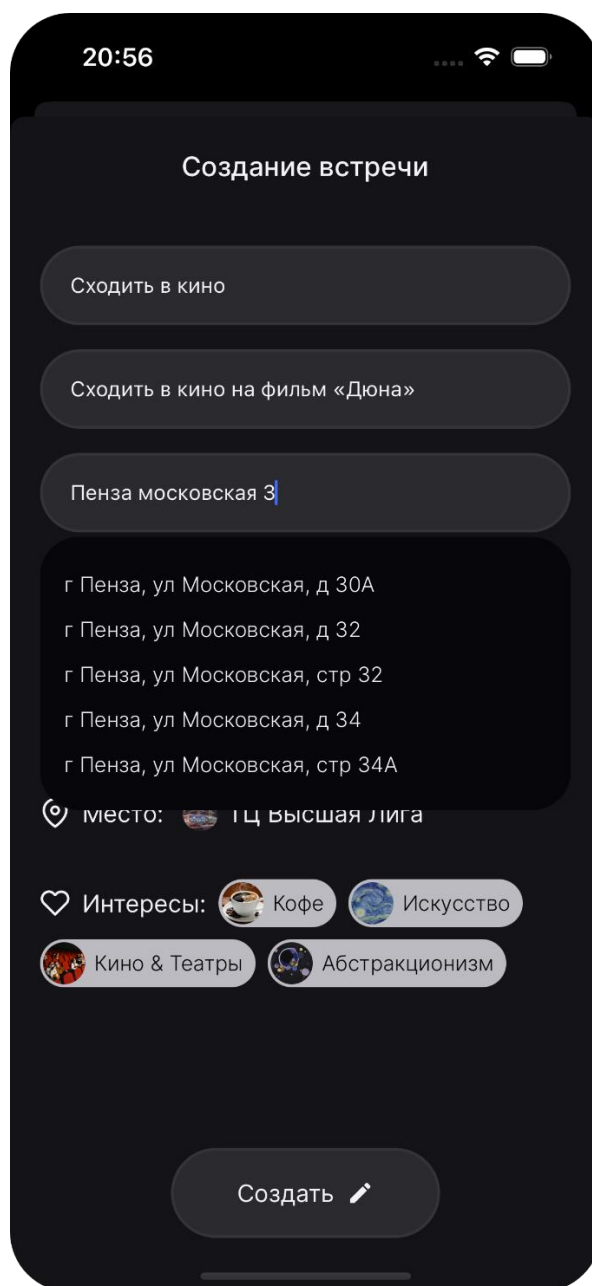


Рисунок 16 – Создание встречи

Пользователь должен указать все необходимые для создания встречи поля (список интересов, место или адрес, название, а также дату и время) в корректной форме, после чего нажать на кнопку «Создать встречу».

После чего будет произведен запрос на сервер, а пользователь возвращен на экран со списком своих групп, где будет отображена новая группа данного пользователя, привязанная к созданной встрече.

Админ-панель

Важной частью проекта является админ-панель, так как администрирование является важной и неотъемлемой частью любой социальной платформы. Доступ в админ-панель получают пользователи, имеющие роль Admin и PlaceOwner.

Администратору доступен весь функционал админ-панели, а вот PlaceOwner ограничен только рамками собственного редактирования места, владельцем которого является.

На рисунках 17-18 можно увидеть, как выглядит админ-панель, а также процесс создания нового места, от лица пользователя, имеющего такие права:

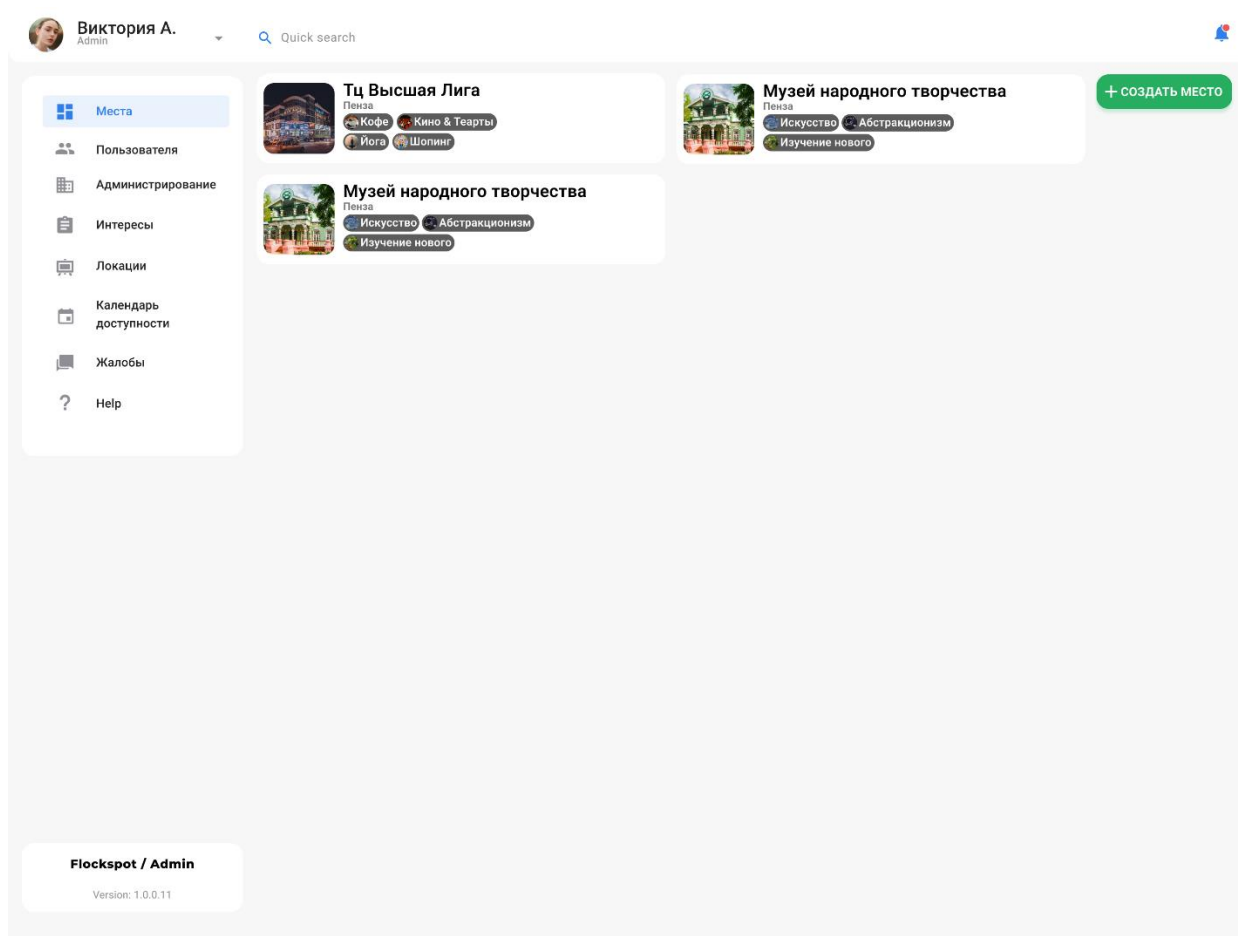


Рисунок 17 – Админ-панель (раздел мест)

Помимо создания, администратор места может редактировать все поля созданных им сущностей, а также удалять их.

Процесс создания места более подробно показан на рисунке 18:

Создание Места

Название:

Описание:

+ ДОБАВИТЬ ИНТЕРЕС

Искусство Изучение нового Музыка

Время работы: Будни Выходные Понедельник Вторник Среда Четверг Пятница Суббота Воскресенье

07 : 00 AM PM

18 : 30 AM PM

☒ **Общедоступное**

☐ Опубликовать после сохранения

+ Сохранить

Flockspot / Admin
Version: 1.0.0.11

Рисунок 18 – Создание места

Как видно на рисунке, администратор указывает название, краткое описание, список интересов, закрепленных за местом, а также время работы.

Флаг общедоступности места указывает, что место является публичным, не требует никаких специальных условий для его посещения, например, дресс-кода и тому подобное.

Кроме основных данных о месте, администратор может указать, желает ли он опубликовать место сразу после сохранения.

Таким образом были рассмотрены основные сценарии использования приложения, как со стороны пользователя, так и со стороны администратора системы, а также администратора места.

Заключение

Таким образом, разработанный программный продукт покрывает всевозможные варианты использования в области приложений для поиска единомышленников и организации социальных событий и встреч, а также имеет критические преимущества на фоне аналогов, а благодаря использованию лучших практик разработки программного обеспечения были достигнуты высокие показатели качества кода, его читаемости и возможности дальнейшего его развития и масштабирования.

Важнейшими преимуществами разработанной системы на фоне аналогов является:

- высокая степень автоматизации процессов: автоматическое создание групп, на основе интересов пользователей; автоматическое создание встреч для уже существующих групп;
- возможность интеграции с публичными заведениями в локации пользователя;
- высокая степень индивидуализации профиля пользователя;
- приятный и минималистичный пользовательский интерфейс;

Особенно стоит отметить возможные перспективы развития разработанного приложения:

- интеграция с календарями пользователей: возможность импортировать напоминание о созданной встрече в календарь пользователя;
- уведомления: интеграция с сервисами уведомлений на мобильных устройствах позволит вовремя напоминать пользователям о предстоящих событиях и встречах;
- сбор статистики для владельцев места: позволит владельцу места лучше понимать портрет пользователя, посетившего его заведение благодаря приложению;

Также, можно сделать вывод о высокой актуальности данной темы и отсутствию предпосылок для изменения этого положения. Большинство

пользователей сильно устали от несовершенства и нацеленности на романтические отношения аналогов данного продукта. Разработанное приложение отвечает высоким и постоянно растущим требованиям пользователей.

В целом, разработанное программное обеспечение отвечает всем поставленным целям и потребностям пользователей, решая не только уже существующие проблемы в данной сфере, но и принося множество новых возможностей.

Таким образом, выпускная квалификационная работа выполнена в полном соответствии с требованиями технического задания и с соблюдением сроков, установленных календарным планом.

Список используемых источников

1. Шульмин, А. Н. Разработка интернет-платформы для организации некоммерческих событий и встреч на основе интересов пользователей / А. Н. Шульмин, А. С. Бождай // Новые информационные технологии и системы (НИТиС-2023): Сборник научных статей по материалам XX Международной научно-технической конференции, посвященной 80-летию юбилею Пензенского государственного университета, Пенза, 16–17 ноября 2023 года. – Пенза: Пензенский государственный университет, 2023. – С. 76-80. – EDN QVZAGH.
2. Вернон В. Реализация методов предметно-ориентированного проектирования. – М.: Вильямс, 2018. – 683 с.: ил.
3. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2023. – 351 с.: ил. – (Серия «Библиотека программиста»).
4. Социальная сеть InParty [Электронный ресурс]. URL: <https://inparty.app>. (Дата обращения 20.04.2024).
5. Сервис поиска знакомств LetzTalk [Электронный ресурс]. URL: <https://beta.letztalk.app>. (Дата обращения: 20.04.2024).
6. Сервис дружеских и профессиональных знакомств RandomCoffee [Электронный ресурс]. URL: <https://random-coffee.ru>. (Дата обращения: 21.04.2024).
7. VK Знакомства – крупнейший сервис знакомств [Электронный ресурс]. URL: <https://dating.vk.com>. (Дата обращения: 21.04.2024).
8. Документация Node js [Электронный ресурс]. URL: <https://nodejs-org-git-meta-performance-bundle-improvements-openjs.vercel.app/it/docs> (дата обращения 13.05.24).
9. Документация | NestJS: [Электронный ресурс]. URL: <https://nestjs.ru>. (Дата обращения: 04.03.2024).

10. Документация по PostgreSQL 16.2: [Электронный ресурс] // СУБД PostgreSQL. URL: <https://postgrespro.ru/docs/postgresql/16/index>. (Дата обращения: 25.03.2024).
11. Документация Flutter [Электронный ресурс]. URL: <https://docs.flutter.dev>. (Дата обращения: 10.04.2024).
12. Статья «Пять простых шагов к пониманию JWT». [Электронный ресурс]. URL: <https://habr.com/ru/articles/340146/> (Дата обращения 07.05.2024).
13. Документация | Prisma ORM [Электронный ресурс]. URL: <https://www.prisma.io/> (Дата обращения: 27.03.2024).
14. Документация Nginx [Электронный ресурс]. URL: <https://nginx.org/ru/> (Дата обращения: 01.04.2024).
15. Dart programming language | Dart [Электронный ресурс]. URL: <https://dart.dev/> (Дата обращения: 10.04.2024).


```
"lastname": "Shulmin",
"birthdate": null,
"nickname": null,
"avatar": null,
"background": null
```

Запрос на создание встречи

HTTP-запрос

```
curl 'http://localhost:3000/api/flocks/manual-meet' \
-H 'sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A/Brand";v="24"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjQyZTAwMzdmLTlINDEtNDIwZS04OGFkLW
M4NjhhYjg5MTVmYyIsImVtYWlsIjoiYW5kcmV5LnNodWxtaW5Ac2h1bG1pbj5jb20iLCJyb2xlIjoi
VXNlciIsImhhdCI6MTcxNjk2ODEyMSwiZXhwIjoxNzE3MDU0NTIxfQ.opcjG4oUbImIbcuKLLIEiD0
zSgOrLlcJTXOnYUb_gcM' \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/125.0.0.0 Safari/537.36' \
-H 'Content-Type: application/json' \
-H 'accept: application/json' \
-H 'Referer: http://localhost:3000/docs/' \
-H 'sec-ch-ua-platform: "macOS"' \
--data-raw '${\n  "vibes": [\n    "181a12df-f4fd-42cd-8339-4f86b9858f74", "199ccde7-c8c9-4d75-
81e4-fbb1441affd7", "be029928-dcc5-4a1f-8885-e1e57ee5424b"\n  ],\n  "startAt": "2024-05-
31T07:44:27.232Z",\n  "description": "Новая встреча с очень глубокими людьми",\n  "name":
"Любители высокого",\n  "address": "г. Пенза, ул. Московская 37"\n}'
```

SQL

```
BEGIN
SELECT * FROM "public"."User" WHERE ("public"."User"."id" = $1)
SELECT "public"."Flock"."id" FROM "public"."Flock" WHERE ("public"."Flock"."id" = $1)
SELECT "public"."Location"."id" FROM "public"."Location" WHERE ("public"."Location"."id" = $1)
INSERT INTO "public"."Flock"
  ("id","size","earliest","type","name","chatId","locationId")
  VALUES ($1,$2,$3,CAST($4::text AS "public"."FlockType"),$5,$6,$7)
RETURNING *;
INSERT INTO "public"."_FlockToVibe" ("A","B") VALUES ($1,$2), ($3,$4), ($5,$6) ON CONFLICT
DO NOTHING
INSERT INTO "public"."FlockUsers" ("createdAt","status","userId","flockId")
  VALUES ($1,CAST($2::text AS "public"."FlockUserStatus"),$3,$4)
RETURNING "public"."FlockUsers"."userId", "public"."FlockUsers"."flockId"
INSERT INTO "public"."Meet" ("id","status","startAt","description","address","flockId")
  VALUES ($1,CAST($2::text AS "public"."MeetStatus"),$3,$4,$5,$6)
RETURNING *
```



```

left JOIN "public"."_FlockToVibes" AS "fv" ON "fv"."A" = "f"."id"
left JOIN "public"."Vibe" AS "v" ON "fv"."B" = "v"."id"
WHERE ("public"."Flock"."id") IN (SELECT "t1"."flockId" FROM "public"."Meet" AS "t1"
WHERE (("t1"."id") IN (SELECT "t2"."meetId" FROM "public"."MeetUser" AS "t2"
WHERE ("t2"."userId" = $1 AND "t2"."status" IN
(CAST($2::text AS "public"."MeetUserStatus"),CAST($3::text AS
"public"."MeetUserStatus")) AND "t2"."meetId" IS NOT NULL))
AND "t1"."flockId" IS NOT NULL))
HTTP-OTBET

```

```

[
{
  "id": "caecbd88-d51f-46c2-b533-c575ca0afd78",
  "status": "Active",
  "startAt": "2024-05-27T18:00:00.000Z",
  "description": null,
  "address": null,
  "flock": {
    "id": "b098b35f-7a30-4484-a9a7-b263cb2324b3",
    "size": 10,
    "name": "Culinary Gatherers, Code Crafters",
    "earliest": "2024-05-27T18:00:00.000Z",
    "background": {
      "original": "http://127.0.0.1:9000/flockspot/dalle_d0ebc5cd-d884-4fa0-85cc-ee1cb5e1c76a.webp",
      "thumbnail": "http://127.0.0.1:9000/flockspot/dalle_d0ebc5cd-d884-4fa0-85cc-ee1cb5e1c76a_thumbnail.webp"
    },
  },
  "chat": null,
  "upcomingMeets": [],
  "vibes": [
    {
      "id": "45d7caa1-94e0-4aab-987a-e6d1b8784830",
      "name": "Food&Cooking",
      "status": "Approved",
      "description": "For culinary experts and food lovers",
      "flockSize": 8,
      "icon": "🍳",
      "background": null
    },
    {
      "id": "ae78f597-78f9-46a5-9d3b-d3f0e4baa3a4",
      "name": "Computer Science",
      "status": "Approved",
      "description": null,
      "flockSize": 6,
    }
  ]
}

```

```

    "icon": "📺",
    "background": null
  },
],
"location": {
  "id": "696d2b34-44bb-41d0-8100-6a046ae3ce5d",
  "name": "Los Angeles",
  "background": null
},
},
"self": {
  "inFlock": "Pending",
  "inMeet": "Pending"
},
"users": [
  {
    "created": "2024-05-24T21:40:41.348Z",
    "status": "Pending",
    "user": {
      "id": "39609478-00b1-4aae-9605-43c4927962ae",
      "role": "User",
      "email": "user.24d3bf7bb6d76@gmail.com",
      "firstname": "random_user_19",
      "lastname": "user_19",
      "birthdate": null,
      "nickname": null,
      "avatar": null,
      "background": null
    }
  },
  {
    "created": "2024-05-24T21:40:41.348Z",
    "status": "Pending",
    "user": {
      "id": "bb150564-db60-4d9d-885a-913d8a6ad7d0",
      "role": "User",
      "email": "greg@gmail.com",
      "firstname": "Greg",
      "lastname": "Oriso",
      "birthdate": "1995-08-25T00:00:00.000Z",
      "nickname": "origreg",
      "avatar": null,
      "background": null
    }
  },
],

```

```

{
  "created": "2024-05-24T21:40:41.348Z",
  "status": "Pending",
  "user": {
    "id": "4352fb6c-643a-49ef-83e2-14ca3cdea83a",
    "role": "User",
    "email": "user.685fe8fd5085a@gmail.com",
    "firstname": "random_user_38",
    "lastname": "user_38",
    "birthdate": null,
    "nickname": null,
    "avatar": null,
    "background": null
  }
},
],
"place": {
  "id": "4bd4d2d4-38a8-4a4c-9000-c87c39aad383",
  "title": "Georgian National Museum",
  "isPublic": true,
  "isApproved": true,
  "description": null,
  "avatar": null
}
},
{
  "id": "f63d8cc9-73bf-421c-99b8-cc2d8ee5caaf",
  "status": "Active",
  "startAt": "2024-05-29T08:00:00.000Z",
  "description": null,
  "address": null,
  "flock": {
    "id": "f623a04f-9c46-4ddf-a2a7-2fd126e4781a",
    "size": 10,
    "name": "Yoga Zenheads\nCode Crusaders\nDrama Dynamos",
    "earliest": "2024-05-29T08:00:00.000Z",
    "background": {
      "original": "http://127.0.0.1:9000/flockspot/dalle_d9368286-c7a8-4ccd-811f-600e73367d86.webp",
      "thumbnail": "http://127.0.0.1:9000/flockspot/dalle_d9368286-c7a8-4ccd-811f-600e73367d86_thumbnail.webp"
    }
  },
  "chat": null,
  "upcomingMeets": [
    {
      "id": "f63d8cc9-73bf-421c-99b8-cc2d8ee5caaf",

```

```

    "status": "Active",
    "startAt": "2024-05-29T08:00:00.000Z",
    "description": null,
    "address": null
  }
],
"vibes": [
  {
    "id": "199ccde7-c8c9-4d75-81e4-fbb1441affd7",
    "name": "Yoga",
    "status": "Approved",
    "description": "For try yoga lovers",
    "flockSize": 10,
    "icon": "☯",
    "background": null
  },
  {
    "id": "ae78f597-78f9-46a5-9d3b-d3f0e4baa3a4",
    "name": "Computer Science",
    "status": "Approved",
    "description": null,
    "flockSize": 6,
    "icon": "💻",
    "background": null
  },
  {
    "id": "181a12df-f4fd-42cd-8339-4f86b9858f74",
    "name": "Theatrs",
    "status": "Approved",
    "description": null,
    "flockSize": null,
    "icon": "🎭",
    "background": null
  }
],
"location": {
  "id": "696d2b34-44bb-41d0-8100-6a046ae3ce5d",
  "name": "Los Angeles",
  "background": null
},
"self": {
  "inFlock": "Pending",
  "inMeet": "Pending"
},

```

```

"users": [
  {
    "created": "2024-05-24T21:40:41.272Z",
    "status": "Pending",
    "user": {
      "id": "7ef5a5c6-6c15-406f-a613-e7c32b7d92c0",
      "role": "User",
      "email": "user.29ca39e164079@gmail.com",
      "firstname": "random_user_11",
      "lastname": "user_11",
      "birthdate": null,
      "nickname": null,
      "avatar": null,
      "background": null
    }
  },
  {
    "created": "2024-05-24T21:40:41.272Z",
    "status": "Pending",
    "user": {
      "id": "bb150564-db60-4d9d-885a-913d8a6ad7d0",
      "role": "User",
      "email": "greg@gmail.com",
      "firstname": "Greg",
      "lastname": "Oriso",
      "birthdate": "1995-08-25T00:00:00.000Z",
      "nickname": "origreg",
      "avatar": null,
      "background": null
    }
  },
  {
    "created": "2024-05-24T21:40:41.272Z",
    "status": "Pending",
    "user": {
      "id": "fe20c56a-5f04-41ee-9e4a-d92a176f6f31",
      "role": "User",
      "email": "user.c8338681e6f2b@gmail.com",
      "firstname": "random_user_4",
      "lastname": "user_4",
      "birthdate": null,
      "nickname": null,
      "avatar": null,
      "background": null
    }
  }
]

```

```

    }
  ],
  "place": {
    "id": "4bd4d2d4-38a8-4a4c-9000-c87c39aad383",
    "title": "Georgian National Museum",
    "isPublic": true,
    "isApproved": true,
    "description": null,
    "avatar": null
  }
}

```

Запрос на принятие приглашения на встречу

HTTP-запрос

```

-X 'POST' \
-H 'sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A/Brand";v="24"' \
-H 'accept: application/json' \
-H 'Referer: http://localhost:3000/docs/' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImJiMTUwNTY0LWRiNjAtNGQ5ZC04ODVhL
TkxM2Q4YTZhZDdkMCIsImVtYWlsIjoiZ3JlZ0BnbWFpbC5jb20iLCJyb2xlIjoiVXNlciIsImhhdCI6MT
cxNjk3MDIyMSwiZXhwIjoxNzE3MDU2NjIxfQ.PkUBuyoIUyGgRRYA3LWm8s5SZpo3XEYedaTn1_
RH6YQ' \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/125.0.0.0 Safari/537.36' \
-H 'sec-ch-ua-platform: "macOS"'

```

SQL

```

SELECT "public"."MeetUser"."userId", "public"."MeetUser"."meetId"
FROM "public"."MeetUser"
WHERE (((("public"."MeetUser"."userId" = $1
AND "public"."MeetUser"."meetId" = $2))
AND "public"."MeetUser"."meetId" IN ($3))
UPDATE "public"."MeetUser"
SET "status" = CAST($1::text AS "public"."MeetUserStatus"), "userId" = $2, "flockId" = $3
WHERE (("public"."MeetUser"."userId" = $4 AND "public"."MeetUser"."meetId" = $5)
AND (("public"."MeetUser"."userId" = $6
AND "public"."MeetUser"."meetId" = $7)))
RETURNING "public"."MeetUser"."userId", "public"."MeetUser"."meetId"

```

Приложение Б. Листинг кода

*/*Мобильное приложение организации социальных событий и встреч на основе интересов пользователей, серверная часть реализована на платформе Node.js и языке программирования JavaScript, клиентская часть реализована с помощью фреймворка Flutter для языка Dart с использованием среды разработки Visual Studio Code*/*

// main.ts (точка входа в серверное приложение)

```
import cookie from '@fastify/cookie';
import multipart from '@fastify/multipart';

import { ConfigService } from '@nestjs/config';
import { NestFactory } from '@nestjs/core';
import {
  FastifyAdapter,
  NestFastifyApplication,
} from '@nestjs/platform-fastify';

import { bootstrapSwagger } from '@swagger/bootstrap';

import { IConfig } from '@core/config/interfaces';
import { IAppLogger } from '@core/modules/shared/abstract';
import { AppCluster } from '@core/modules/shared/services';

import { AppModule } from './app.module';

// функция вызывается при запуске серверного приложения
async function bootstrap() {
  const app = await NestFactory.create<NestFastifyApplication>(
    AppModule,
    new FastifyAdapter(),
    { bufferLogs: true },
  ); // конфигурация логирования и модуля http

  const { port, prefix } = app
    .get(ConfigService<IConfig, true>)
    .get('app', { infer: true });

  await app.register(cookie as any); // регистрация модуля обработки cookie
  await app.register(multipart as any); // регистрация модуля обработки файлов

  app.useLogger(app.get(IAppLogger));
  app.setGlobalPrefix(prefix);
  bootstrapSwagger(app);
  await app.listen(port, '0.0.0.0');
}

const isProd = process.env.NODE_ENV === 'prod';
if (isProd) AppCluster.clusterize(bootstrap); // кластеризация приложения при использовании в производственной среде
else bootstrap();
```

// user-registration.handler.ts (регистрация пользователя)

```
import { CommandHandler, ICommandHandler } from '@nestjs/cqrs';

import { ICryptoService } from '@core/modules/shared/abstract';

import { IJwtService } from 'src/modules/user/domain/adapters/jwt.adapter';
import { UserFactory } from 'src/modules/user/domain/factories/user.factory';
import { IUserRepository } from 'src/modules/user/domain/user.repository';

import { UserAuthorizedResult } from '../user.authorized.result';
import { RegisterCommand } from './register.command';

@CommandHandler(RegisterCommand)
export class RegisterHandler
  implements ICommandHandler<RegisterCommand, UserAuthorizedResult>
{
  constructor(
    private readonly userRepo: IUserRepository,
    private readonly crypto: ICryptoService,
    private readonly jwt: IJwtService,
    private readonly factory: UserFactory,
  ) {}

  public async execute(
    command: RegisterCommand,
  ): Promise<UserAuthorizedResult> {
    const user = this.factory.create({
      ...command,
      password: await this.crypto.hash(command.password), // хэширование пароля
    });

    await this.userRepo.save(user); // сохранение пользователя в базе данных

    const [access, refresh] = await this.jwt.sign({
      id: user.Id,
      email: user.Email,
      role: user.Role,
    }); // генерация пары JWT токенов

    return { access, refresh, user };
  }
}
```

// user-login.handler.ts (авторизация пользователя)

```
import { CommandHandler, ICommandHandler } from '@nestjs/cqrs';

import { ICryptoService } from '@core/modules/shared/abstract';

import { DomainError } from 'src/modules/abstract/throwable';
import { IJwtService } from 'src/modules/user/domain/adapters/jwt.adapter';
import { IUserRepository } from 'src/modules/user/domain/user.repository';

import { UserAuthorizedResult } from '../user.authorized.result';
import { LoginCommand } from './login.command';
```

```

@CommandHandler(LoginCommand)
export class LoginHandler
  implements ICommandHandler<LoginCommand, UserAuthorizedResult>
{
  constructor(
    private readonly userRepo: IUserRepository,
    private readonly crypto: ICryptoService,
    private readonly jwt: IJwtService,
  ) {}

  public async execute({
    email,
    password,
  }: LoginCommand): Promise<UserAuthorizedResult> {
    const user = await this.userRepo.findByEmail({ email }); // чтение пользователя из базы данных

    const isPasswordEquals =
      user.Password?.length &&
      (await this.crypto.compare(user.Password, password)); // хэширование переданного пароля для сравнение
    // сравнение паролей
    if (!isPasswordEquals) throw new DomainError('Invalid credentials');

    const [access, refresh] = await this.jwt.sign({
      id: user.Id,
      email: user.Email,
      role: user.Role,
    }); // генерация пары JWT токенов

    return { access, refresh, user };
  }
}

```

// matching.service.ts (матчинг пользователей)

```

import { EntityId } from '@core/domain';

import { Nullable } from '@app.types/common';

import { Availability } from 'src/modules/abstract/entities';
import { Flock } from 'src/modules/flock/domain/entities';
import { Place } from 'src/modules/place/domain/entities';
import { User } from 'src/modules/user/domain/entities';

import {
  IMatch,
  IMatchingService,
  IPickedPlace,
  PickedPlaceType,
} from './base.matching.service';

export class MatchingService implements IMatchingService {
  public GetMatches(_users: User[], flocks: Flock[]): IMatch[] {
    const users = _users.sort((a, b) => b.Vibes.length - a.Vibes.length);
  }
}

```

```

let level = users[0].Vibes.length;
const matches: IMatch[] = [];

while (level >= 1) {
  for (let i = 0; i < users.length; i++) {
    let vibes: EntityId[] = users[i].Vibes;
    let availabilities: Availability[] = users[i].Availabilities;
    const matched: User[] = [users[i]];

    for (let j = 0; j < users.length; j++) {
      if (i == j) continue;

      const intersection = users[j].GetVibesIntersection(vibes); // получить пересечение интересов пользователей
      if (intersection.length != level) continue;

      const availabilitiesIntersection = availabilities.filter((av) =>
        users[j].Availabilities.some((x) => x.IsOverlaps(av)),
      ); // получить список пересекающихся объектов времени доступности пользователей
      if (!availabilitiesIntersection.length) continue;

      vibes = intersection;
      availabilities = availabilitiesIntersection;

      matched.push(users[j]);
    }

    const matchedWithoutDuplicates = matched.filter(
      (u) =>
        !flocks
          .filter(({ Users }) => Users.some((x) => x.User == u.Id))
          .some(({ Users }) =>
            Users.some(
              (x) =>
                x.User != u.Id && matched.some((user) => user.Id == x.User),
            ),
          ) &&
        !matches.some(
          (m) =>
            m.users.some((x) => x == u.Id) &&
            u.GetVibesIntersection(m.vibes, vibes).length == vibes.length,
        ),
    ); // получить список уникальных матчей с учетом уже существующих групп пользователей и интересов в них

    if (matchedWithoutDuplicates.length > 2) {
      const nearest = this.PickNearestDate(matchedWithoutDuplicates);
      const users = matchedWithoutDuplicates.map((u) => u.Id);
      if (nearest) matches.push({ level, vibes, users, nearest });
    }
  }

  level--;
}

return matches;

```

```

}
// выбор подходящего места для проведения встречи
public PickPlace(
  places: Place[],
  vibes: EntityId[],
  date: Date,
): IPickedPlace {
  const candidates = places
    .sort(
      (a, b) =>
        a.GetVibesIntersection(vibes).length -
        b.GetVibesIntersection(vibes).length,
    )
    .filter(
      (x) =>
        x.GetVibesIntersection(vibes).length > 0 &&
        x.Availabilities.some((av) => av.IsMatchWithDate(date)),
    );

  if (candidates.length <= 0)
    return {
      type: PickedPlaceType.Random,
      place: places[Math.floor(Math.random() * places.length)],
    };

  return {
    type: PickedPlaceType.Matched,
    place: candidates.at(-1)!,
  };
}
// метод для поиска ближайшей даты для встречи
public PickNearestDate(users: User[]): Nullable<Date> {
  const availabilities = users.map((x) => x.Availabilities);
  return Availability.GetNearestDate(
    Availability.GetGeneralAvailabilities(...availabilities),
  );
}
}

```

// transactional.decorator.ts (декоратор реализующий транзакции базы данных)

```

import { Inject } from '@nestjs/common';

import { DatabaseClient } from '../database.service';

export function Transactional(key?: string | symbol): MethodDecorator {
  const injectDb = Inject(DatabaseClient);

  return (target: any, _: string | symbol, descriptor: PropertyDescriptor) => {
    const original = descriptor.value;
    const _key = key ?? '_db';

    if (!key) injectDb(target, _key);

    descriptor.value = async function (...args: unknown[]) {

```

```

    if (!(_key in this) || !('$transaction' in (this as any)[_key]))
      throw new Error('Unknown Inject key '${_key.toString()}'');

    const db: DatabaseClient = (this as any)[_key];
    return db.$transaction(async () => original.apply(this, args));
  };

```

```

    return descriptor;
  };
}

```

// main.dart (точка входа в мобильное приложение)

```

import 'package:flockspot_app/app_widget.dart';
import 'package:flockspot_app/injection/injection.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:injectable/injectable.dart';

```

```

const env = String.fromEnvironment('ENV', defaultValue: Environment.dev);

```

```

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await configureInjection(environment: env);

```

```

  runApp(const ProviderScope(
    child: AppWidget(),
  ));
}

```

// api_clients.dart (абстрактный класс для работы с сетевыми запросами)

```

import 'dart:io';

```

```

import 'package:dio/dio.dart' hide Headers;
import 'package:flockspot_app/flockspot_models/flockspot_models.dart';
import 'package:flockspot_app/flockspot_models/models/created_location.dart';
import 'package:flockspot_app/flockspot_models/models/generated_image.dart';
import 'package:flockspot_app/flockspot_models/models/generated_text.dart';
import 'package:injectable/injectable.dart';
import 'package:retrofit/retrofit.dart';

```

```

part 'api_client.g.dart';

```

```

@injectable
@RestApi()
abstract class ApiClient {
  @factoryMethod
  factory ApiClient(Dio dio, {@Named('BaseUrl') String baseUrl}) = _ApiClient;

```

```

  @POST('api/auth/login')
  @FormUrlEncoded()
  Future<UserToken> login({@Body() required TokenRequest body});

```

```

  @POST('api/auth/register')
  Future<UserToken> register({@Body() required UserRegisterRequest body});

```

```

@POST('api/auth/refresh')
Future<UserToken> refresh();

@GET('api/flocks/my')
Future<Paginator<Flock>> getMyFlocksList({
    @Query('take') int take = 10,
    @Query('skip') int skip = 0,
});

@GET('api/flocks/{id}')
Future<Flock> getFlock({@Path('id') required String id});

@PATCH('api/flocks/{id}/leave')
Future<bool> leave({@Path('id') required String id});

@GET('api/locations/onboarding')
Future<Paginator<Location>> getLocationsOnboarding({
    @Query('take') required int take,
    @Query('skip') required int skip,
    @Query('name') String name = "",
});

@POST('api/locations')
Future<CreatedLocation> addLocation(
    {@Body() required AddLocationRequest addLocation});

@GET('api/flocks/meets/invitations')
Future<List<Meet>> getInvitations({
    @Query('statuses') String? statuses,
    @Query('start') DateTime? start,
    @Query('end') DateTime? end,
});

@POST('api/flocks/{id}/meets/{meetId}/accept-invitation')
Future<bool> acceptInvitation({
    @Path('id') required String flockId,
    @Path('meetId') required String meetId,
});

@POST('api/flocks/{id}/meets/{meetId}/cancel-invitation')
Future<bool> cancelInvitation({
    @Path('id') required String flockId,
    @Path('meetId') required String meetId,
});

@GET('/api/profile')
Future<User> getProfile();

@PATCH('api/profile')
Future<User> updateProfile(
    {@Body() required UpdateProfileRequest updateProfile});

@POST('api/profile/avatar')
Future<Avatar> updateAvatar({@Body() required File file});

```

```

@PATCH('api/profile/vibes')
Future<bool> updateVibes({required List<String> list});

@POST('api/profile/availability')
Future<Availability> updateAvailability(
  {@Body() required UpdateAvailabilityRequest updateAvailability});

@DELETE('api/profile/availability/{id}')
Future<bool> deleteAvailability({@Path('id') required String id});

@GET('api/users/{id}')
Future<User> getUser({@Path('id') required String id});

@GET('api/vibes/onboarding')
Future<Paginator<Vibe>> getVibesOnboarding(
  {int take = 10, int skip = 0, String name = ""});

@GET('api/vibes')
Future<Paginator<Vibe>> getVibes(
  {int take = 10, int skip = 0, String name = ""});

@POST('api/vibes')
Future<Vibe> postVibe({@Body() required UpdateVibeRequest updateVibe});

@POST('api/openai/generate/image')
Future<GeneratedImage> generateImage({@Field() required String prompt});

@POST('api/openai/generate/text')
Future<GeneratedText> generateText({@Field() required String prompt});
}

```