МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА СИНТЕЗА ОПТИМАЛЬНЫХ МНОГОЦЕЛЕВЫХ МАРШРУТОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ТУРИСТИЧЕСКИХ ПРЕДПОЧТЕНИЙ

Студент _____ 21.06.2024 Прокофьев Никита Андреевич
(подпись, дата) (ФИО полностью)

Руководитель _____ 21.06.2024 Финогеев А. Г.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 24.06.2024 Бычков А. С.
(подпись, дата) (фамилия, инициалы)

Рецензент доцент ПензГТУ _____ 24.06.24 Пегорский С. В.
(подпись, дата) (фамилия, инициалы) (должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой _____ Митрохин М. А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 25.06.24 № 41)

Секретарь ГЭК _____ Гурин Е. И.
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой

_____М. А. Митрохин
«13» сентября 2022г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ МАГИСТРА

1.Студент Прокофьев Никита Андреевич гр. 22ВВИм2 факультета вычислительной техники, направления 09.04.01.

2. Тема работы «Рекомендательная система синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений».

Тема утверждена приказом ПГУ № 765/о от "13"сентября 2022 г.

3. Руководитель магистерской диссертации д.т.н., профессор кафедры САПР ПГУ Финогеев А. Г.

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

Назначение разработки: создание рекомендательной системы синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений.

Разработать автоматизированное программное обеспечение с использованием искусственного интеллекта для предоставления информации по туристическим маршрутам на основе предпочтений пользователей.

5. Перечень подлежащих разработке вопросов:

1) Анализ уже действующих решений из сферы рекомендательных систем.
2) Создание базы данных.
3) Разработка рекомендательной системы на основе ИИ.

4) Тестирование системы согласно ранее описанному техническому заданию на разработку.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучить теоретические основы	10%	27.09.2022–15.11.2023	
2.	Анализ технических заданий, анализ существующих ИИ– ассистентов	10%	16.11.2023–22.03.2023	
3.	Выбор инструментов для разработки и утверждение функционала системы	5%	23.03.2023–14.06.2023	
4.	Спроектировать базу данных	10%	15.06.2023–10.09.2023	
5.	Написать логику работы приложения	40%	11.09.2023–05.01.2024	
6.	Интегрировать ИИ– ассистента и реализовать подключение собственного датасета	15%	06.01.2024–17.04.2024	
7.	Апробация разработанного клиент–серверного приложения в форме тестирования	5%	18.04.2024–9.05.2024	
8.	Оформление выпускной работы	5%	10.05.2024–28.05.2024	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации _____ 13.09.2022 Финогеев А.Г.
(подпись, дата)

Задание к исполнению принял студент _____ 13.09.2022 Прокофьев Н. А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 24.06.2024 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 88 листов, 22 рисунка, 17 таблиц, 11 источников, 2 приложения

AIOGRAM, SQLALCHEMY, PYTHON, PYCHARM IDE, ALEMBIC, CHATGPT, OPENAI, OPENWEATHERMAP, ЧАТ– БОТЫ, ОБРАБОТКА СОСТОЯНИЙ, РЕКОМЕНДАТЕЛЬНЫЙ СИСТЕМЫ, ПОСТРОЕНИЕ МАРШРУТОВ, БАЗА ДАННЫХ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, НЕЙРОННЫЕ СЕТИ

Цель работы – необходимо разработать рекомендательную систему синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений.

В результате проведенной работы было разработано программное обеспечение, позволяющее автоматизировать поиск оптимальных туристических маршрутов и решить проблему информационного пузыря, за счет использования метода корреляционной фильтрации и интеграции с искусственным интеллектом, позволяющие использовать конкретный датасет для использования искусственным интеллектом.

В ходе разработки рекомендательной системы, использовались следующие технологии: python, aiogram, sqlalchemy, alembic, sqlite, telegram API, ChatGPT, OpenWeatherMap, currency converter.

					ПГУ 1.090401.55.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Прокофьев Н.А.		11.04.24	Рекомендательная система синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений Пояснительная записка	Лит.	Лист	Листов
Провер.		Финогеев А. Г.		11.04.24			4	88
Н. Контр		Бычков А.С.		11.04.24		ФВТ, каф. ВТ гр. 22ВВИм2		
.								
Утверд.		Митрохин М.А.		11.04.24				

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 7
ВВЕДЕНИЕ 8
1 Постановка задачи 9
2 Анализ существующих рекомендательных систем 11
2.1 Аналоги 11
2.1.1 Tripomatic 11
2.1.2 Inspirock 12
2.1.3 Routeperfect 13
2.2 Основной функционал 14
3 Обзор методов построения рекомендательной системы 18
3.1 Матричная факторизация 19
3.2 Нейронные сети 22
3.3 Байесовские классификаторы 28
4 Обзор существующих методов анализа туристических предпочтений на основе искусственного интеллекта 31
4.1 Обучение с учителем 31
4.2 Обучение без учителя 32
4.3 Глубокое обучение и его применение в анализе поведения туристов 32
4.4 Использование API для доступа к актуальной информации об услугах 33
5 Проектирование рекомендательной системы 36
5.1 Выбор инструментов разработки 38
5.1.1 Выбор мессенджера 38
5.1.2 Выбор ИИ – ассистента 38
5.1.3 Выбор языка программирования и среды разработки 39
5.2 Исходные данные для построения системы рекомендательных сервисов 41
5.3 Создание чат-бота 49
5.4 Интерфейс приложения и доступные функции 55

5.5 Описание программы 60
ЗАКЛЮЧЕНИЕ 70
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 71
ПРИЛОЖЕНИЕ А 73
ПРИЛОЖЕНИЕ Б 84

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – база данных

ИИ – искусственный интеллект

ПК – персональный компьютер

СУБД – система управления базами данных

ЯП – язык программирования

ACID (Atomicity, Consistency, Isolation, Durability) – требования к транзакционной системе, обеспечивающие наиболее надёжную и предсказуемую её работу

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

GPT (Generative Pre-trained Transformer) – генеративный, предобученный трансформер

IDE (Integrated development environment) – интегрированная среда разработки

ID (Identifier) – идентификатор объекта

JSON (JavaScript Object Notation) – текстовый формат обмена данными

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

ВВЕДЕНИЕ

В современном мире туризм занимает одно из ведущих мест в структуре мировой экономики и культуры. Каждый год миллионы людей отправляются в путешествия, в поисках новых впечатлений, отдыха, знаний или вдохновения. В эру цифровизации и персонализации услуг возникает потребность в индивидуальном подходе к планированию туристических поездок, что ставит перед нами задачу разработки сложных и многозадачных систем, способных удовлетворить самые разнообразные запросы и предпочтения потребителей. В ответ на эти вызовы возникают алгоритмы и подходы для анализа туристических предпочтений и создания оптимальных маршрутов, основанных на больших данных и методах машинного обучения.

Важность таких систем нельзя недооценивать: они предоставляют возможность не только экономить время при планировании поездок, но и обеспечивают более глубокое взаимодействие между туристами и местами, которые они посещают, делая путешествие более полезным и гармоничным. Использование машинного обучения для анализа данных о предыдущих поездках пользователей открывает новые горизонты для создания персонализированных рекомендаций, сочетая в себе индивидуальные предпочтения и объективные данные о мировых туристических тенденциях.

Целью данной практической работы является разработка рекомендательной системы синтеза оптимальных многоцелевых маршрутов на базе искусственного интеллекта, которая будет отвечать современным требование, функциональности и удобству пользования. В ходе практики буду рассмотрены различные аспекты, такие как анализ существующих систем и алгоритмов, разработка программного обеспечения, создание дополняемой базы данных, а также оценка эффективности внедрения рекомендательных систем в туристическую среду.

1 Постановка задачи

В рамках преддипломной практики ставится задача разработать рекомендательную систему для подбора оптимального маршрута для туристических путешествий, которая будет интегрироваться в образовательную среду и обеспечивать эффективное управление данными студентов. Система должна предоставлять возможности для выбора интересующего города или страны, получения полной информации о местонахождении объектов и их описании, а также получении сведения о погоде в интересующем месте. В качестве дополнения, в программное обеспечения будет добавлен конвертер валют, для увеличения функционала и удобства пользователя.

Объект исследования**:** объект пользовательского интереса (туристические маршруты, события, локации).

Предмет исследования: рекомендательный сервис для решения задачи прогнозирования туристических маршрутов на основе предпочтений пользователя.

Цель исследования: разработка рекомендательной системы туристических маршрутов с целью сокращения временных затрат пользователя на поиск туристического маршрута, удовлетворяющего его требованиям, а также повышения востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее.

Задачи исследования:

− анализ критериев классификации туристических маршрутов,

− выявление методов, необходимых для классификации туристических маршрутов и используемых для формирования списка релевантных объектов интереса на основе предпочтений пользователя,

− разработка рекомендательной системы туристических маршрутов.

Результатом проделанной работы должна стать полнофункциональная рекомендательная система поиска оптимальных многоцелевых маршрутов на базе искусственного интеллекта, способная адаптироваться к изменчивым тенденциям в области туризма и удовлетворять потребности пользователя в построении идеального путешествия.

2 Анализ существующих рекомендательных систем

2.1 Аналоги

Из всего многообразия приложений были выделены 3 наиболее популярных. К ним относятся: Tripomatic, Inspirock и Routeperfect. Именно эти системы в дальнейшем будут рассмотрены и проанализированы на сильные и слабые стороны.

2.1.1 Tripomatic

Данный аналог содержит подробные карты городов, на которых отмечены основные достопримечательности, а также рестораны, вокзалы, гостиницы (с описанием и фотографиями). Вы можете отметить для себя все, что захотите посетить.

Положительные стороны:

− наличие 25+ языков (в том числе русского),

− удобный интерфейс,

− возможность бронирования автомобиль для путешествия,

− наличие навигатора для мобильного приложения,

− возможность вручную добавлять достопримечательности в маршрут или выбрать из представленных.

Отрицательные стороны:

− отсутствие возможности выбора отеля или аренда машины на территории Российской Федерации,

− отсутствие мобильного приложения,

− отсутствие автоматического подбора маршрутов на основе интересов пользователя.

Интерфейс сервиса предоставлен на рисунке 1.

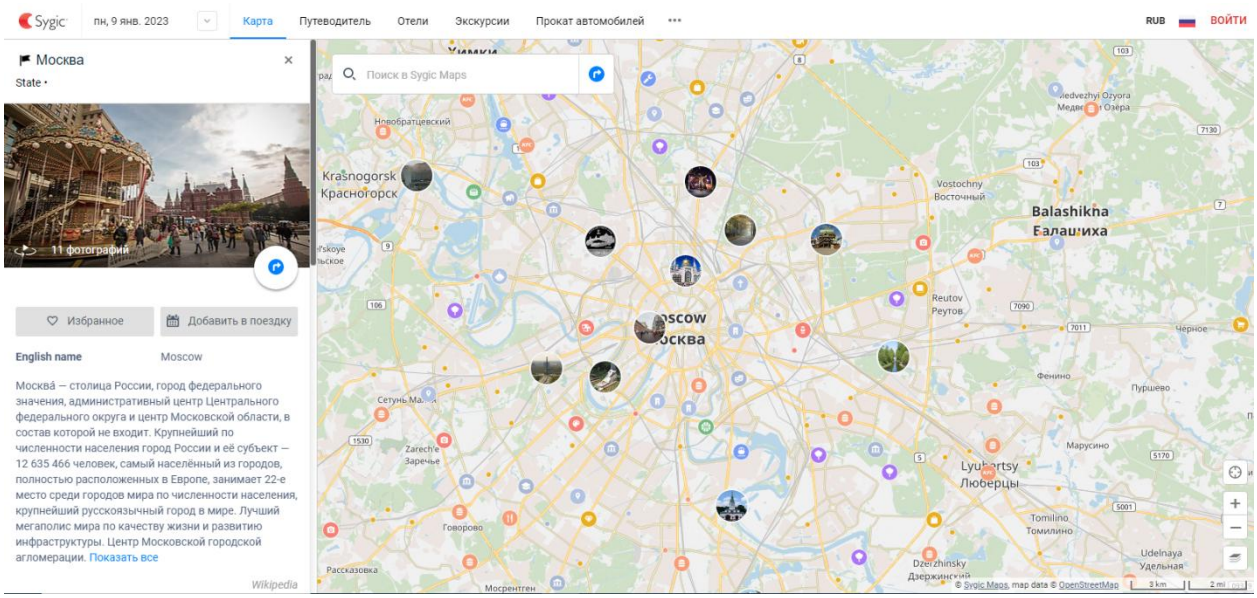


Рисунок 1 – Интерфейс “Tripomatic”

Вывод**:** Представленное приложение имеет сильные и слабые стороны. Оно подходит для построения маршрута по Европе, но на территории России отсутствуют многие важные функции.

2.1.2 Inspirock

Данный аналог – это приложение для составления маршрутов, которое помогает в планировании поездки и сборе важной информации в одном месте. Включает в себя информацию о достопримечательностях, отелях.

Сильные стороны:

– возможность ручного добавления достопримечательности в маршрут или выбор из представленных,

– наличие автоматического подбор маршрута, на основе предпочтений и длительности путешествия,

– наличие возможности забронировать отель, внести информацию о билетах на самолет.

– возможность арендовать машину.

Слабые стороны:

– отсутствие возможности выбрать отель или арендовать машину в России,

– отсутствие мобильного приложения,

– поддержка малого количества языков, в том числе русского.

Интерфейс сервиса предоставлен на рисунке 2.

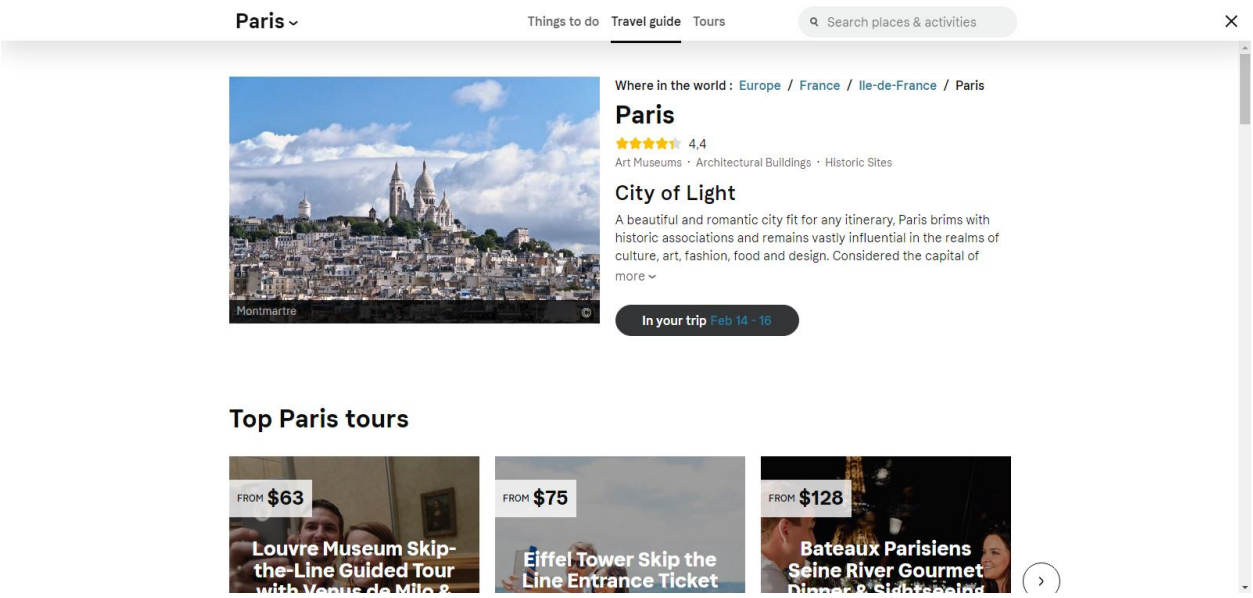


Рисунок 2 – Интерфейс “ Inspirock ”

Вывод: Приложение подходит для путешествия по Европе, но имеет ограниченный функционал на территории России. Помимо этого, наличие интуитивно непонятного интерфейса затрудняет взаимодействие с сервисом.

2.1.3 Routeperfect

Данный аналог в отличие от предыдущих приложений является наиболее подходящим для путешествия по России. Включается в себя большое количество достопримечательностей и отелей.

Сильные стороны:

– адаптирован для путешествий по России,

– интуитивно понятный интерфейс,

– возможность забронировать автомобиль для путешествия,

– возможность забронировать отель,

– автоматическое построение маршрута на основе длительности путешествия и интересов,

– возможность ручного добавления достопримечательности в маршрут,

– наличие мобильного приложения.

Слабые стороны:

– отсутствие русского языка,

– построение маршрута осуществляется исключительно для путешествия на автомобиле.

Интерфейс сервиса предоставлен на рисунке 3.

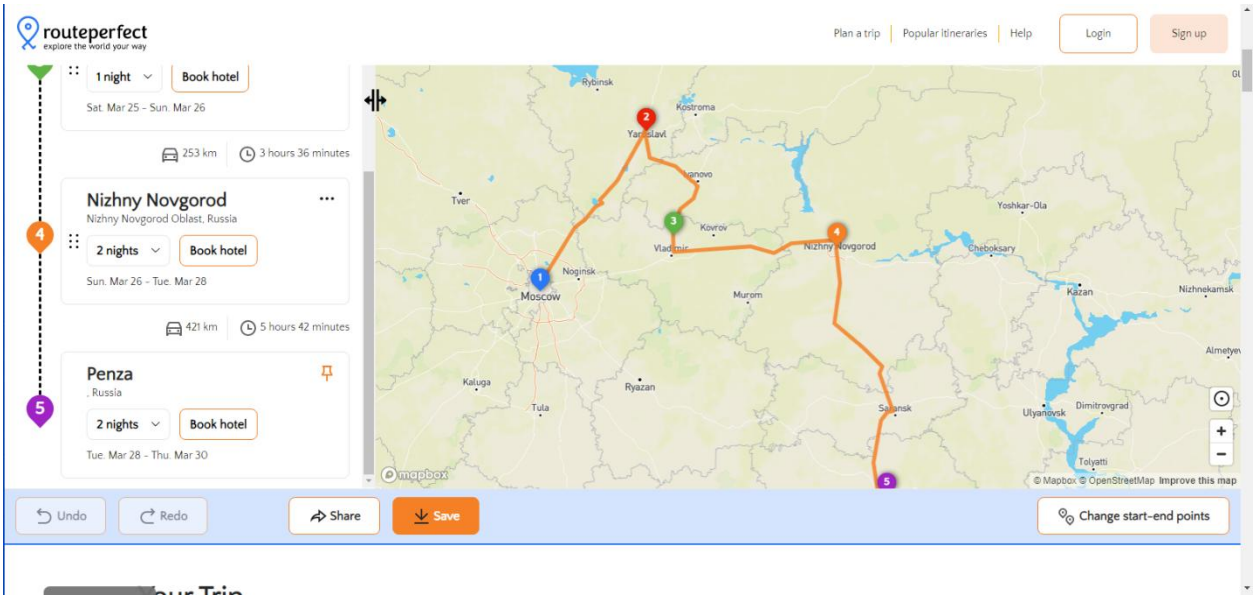


Рисунок 3 – Интерфейс “ Routeperfect ”

Вывод: Приложение отлично адаптировано для путешествий по России и Европе. Широкий функционал и интуитивно понятный интерфейс исключает сложности в использовании.

2.2 Основной функционал

В качестве основных качеств можно выделить: разнообразие языков, наличие мобильного приложения, автоматический подбор маршрута на основе интересов пользователя и функционал(возможность забронировать отель, приобрести билеты на поезд/самолет, разнообразие выбора досуга).

Для выделения лучшего приложения из представленных, используются методы многокритериальной оптимизации, а именно метод минимальной точки.

Данный метод имеет уникальную и очень важную особенность: он позволяет получить информацию как об идеальном фреймворке, так и о взглядах потребителя на существующие фреймворки. Формула, на которой основан метод идеальной точки (1), выглядит следующим образом:

$$\sum_{i=1}^{n} \lambda_i * |Q_i - X_i|, \quad (1)$$

где λ_i – значимость показателя

Q_i – идеальное значение характеристики

X_i – мнение о фактической величине показателя

n – число значимых показателей

Итоговое значение отношения к инструменту разработки определяется следующим образом. Сначала находится разность между идеальным и фактическим значением показателя. Берется абсолютная величина разности, на что указывает символ модуля в формуле. Полученная величина затем умножается на значение важности. Далее аналогичным образом определяются слагаемые по остальным характеристикам и, наконец, их сумма. Предпочтение отдается наименьшим результатам. Наилучшее значение отношения равно нулю, что говорит о том, что инструмент разработки в точности соответствует представлениям потребителей об идеальном наборе свойств.

Полученные данные указаны в таблице 1.

Таблица 1 – Оценка характеристик

	Tripomatic	Inspirock	Routeperfect	Л	Q
Разнообразие языков	1	0,3	0,1	0,2	1
Функционал	1	1		0,4	1
Наличие приложения	0,5	0,5	1	0,2	1
Подбор маршрутов по интересам	0,5	0,5	1	0,2	1

На основе полученных данных производится расчёт коэффициентов по формуле 1.

Результаты расчетов указаны в таблицах 2, 3.

Таблица 2 – Разность между значением показателя

Q − x		
Tripomatic	Inspirock	Routeperfect
0	0,7	0,9
0	0	0.3
0,5	1	0
0,5	0,5	0

Таблица 3 – Отношение к сервису

Λ*|(Q – x)|		
Tripomatic	Inspirock	Routeperfect
0	0,14	0,18
0	0	0.12
0,1	0.2	0
0,1	0,1	0

На основе полученных данных, составляется таблица 4, на основе которой можно сделать выводы о наиболее подходящем сервисе из представленных выше.

Таблица 4 – Результат вычислений

	Tripomatic	Inspirock	Routeperfect	Л	Q
Разнообразие языков	1	0,3	0,1	0,2	1
Функционал	1	1		0,4	1
Наличие приложения	0,5	0,5	1	0,2	1
Подбор маршрутов по интересам	0,5	0,5	1	0,2	1
Метод идеальной точки	0,1	0,44	0,3		

Вывод: Анализируя полученные в ходе исследования результаты, можно отметить, что сервис Tripomatic показал наименьший результат по

всем изучаемым критериям. Это говорит о том, что этот сервис наиболее соответствует нашим исходным требованиям и ожиданиям по части инструмента разработки мобильных приложений.

Оценка проводилась по множеству параметров, каждый из которых имел свое значение при принятии решения. Исследовались не только технические аспекты работы сервиса, вроде быстродействия, но и удобство интерфейса, работа отдельных функций, соответствие предпосылкам и т.д.

По результатам нашего исследования, можно сказать, что Tripomatic будет оптимальным выбором для дальнейшей работы. Этот сервис сочетает в себе наиболее важные для нас функциональные возможности, поэтому он будет служить примером при разработке функционала для нашей рекомендательной системы.

Такое решение позволит нам сконцентрироваться на внедрении наиболее важных и необходимых функций, повысив тем самым качество и эффективность нашего продукта. Опираясь на пример успешно работающего сервиса, мы сможем усовершенствовать наше приложение и сделать его еще более удобным и полезным для пользователей.

3 Обзор методов построения рекомендательной системы

Современные подходы формирования прогнозов, представленные на рисунке 4, можно разделить на две базовые группы:

1) алгоритмы, основанные на данных, в которых выявляется уровень подобия объектов и субъектов системы, базируясь на имеющейся исходной информации,

2) алгоритмы, основанные на моделях, в которых создается, а затем обучается модель пользователя, объектов интереса и взаимосвязей между ними [1]. Алгоритмы первой группы в сравнении со второй являются наименее точными при формировании релевантных подборок, требуют накопления большого объема целевой информации, отсутствие которой приводит к невозможности составления списка рекомендаций [2, 3]. На основе этого, для выполнения данной работы были рассмотрены алгоритмы второй группы.

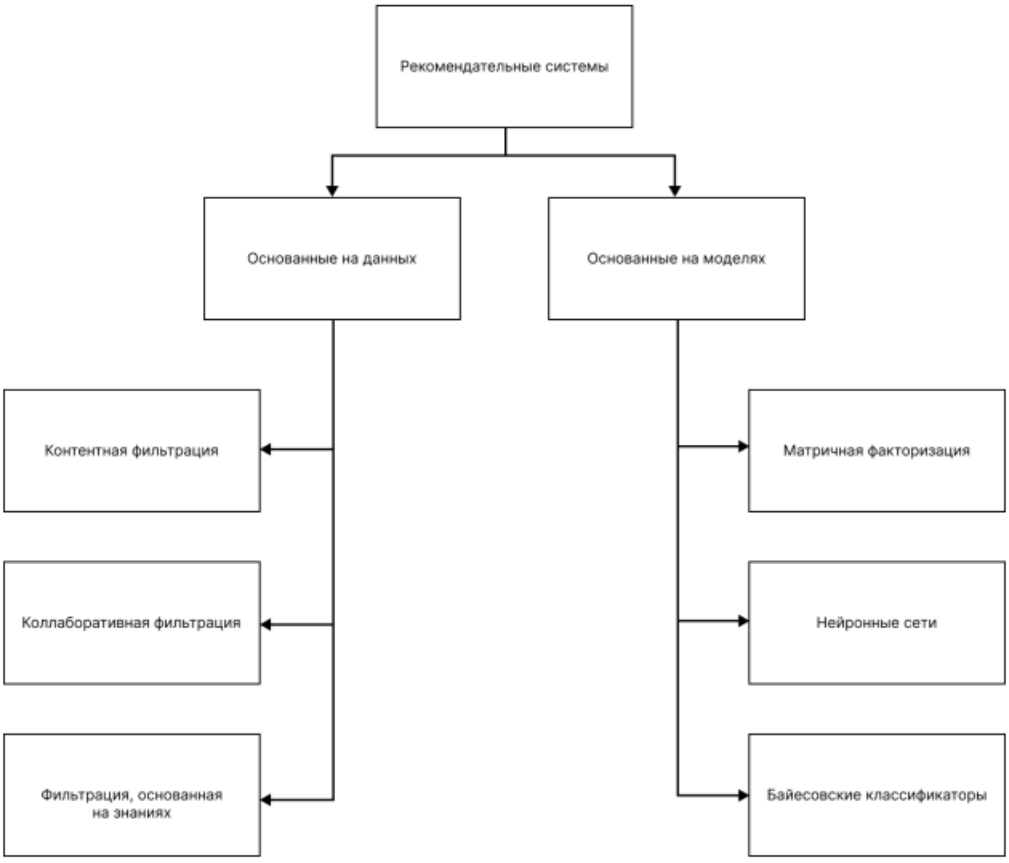


Рисунок 4 – Современные подходы формирования прогнозов

Подходы, основанные на моделях, при составлении рейтинговых прогнозов используют накопленную информацию для формирования модели и ее дальнейшего обучения. Модели в данном подходе базируются на коллективном поведении пользователей информационной системы, что приводит к созданию поведенческих паттернов, представленных в компактной форме. Результатом обучения модели является функция, позволяющая получить рекомендательный прогноз, исходя из уже имеющихся оценок. При поступлении новых данных модель модернизируется для повышения точности рекомендаций. При этом весь процесс использования модели разбивается на ресурсоемкую оптимизацию параметров и структуры модели (обучение) и не требовательный к ресурсам прогноз рейтингов в реальном времени [4].

Среди наиболее широко используемых методов формирования рекомендаций на основе моделей выделяют: алгоритмы матричной факторизации, нейронные сети и байесовские классификаторы.

3.1 Матричная факторизация

В последнее время модели матричной факторизации, являющиеся классом алгоритмов коллаборативной фильтрации, приобрели популярность в рекомендательных системах ввиду своей высокой точности и масштабируемости.

Методы редукции размерности, нацеленные на решение проблем с разреженностью в базах данных рекомендательных систем, так же основаны на матричной факторизации, поскольку данный подход наиболее полезен при обработке больших объемов данных. Данный механизм позволяет ускорить вычисления путем понижения размерности.

Модели факторизации матриц формируют содержательное описание объектов интереса пользователей в терминах латентных признаков, что частично решает проблему ограниченного покрытия.

Алгоритмы матричной факторизации работают путем разложения матрицы взаимодействия пользователя и объекта интереса A на

произведение двух матриц меньшей размерности, а именно на отдельные двумерные массивы пользователей U и матрицы объектов интереса V. Разложение исходной матрицы происходит с применением алгоритма ранговой факторизации, схематично представленной на рисунке 5.

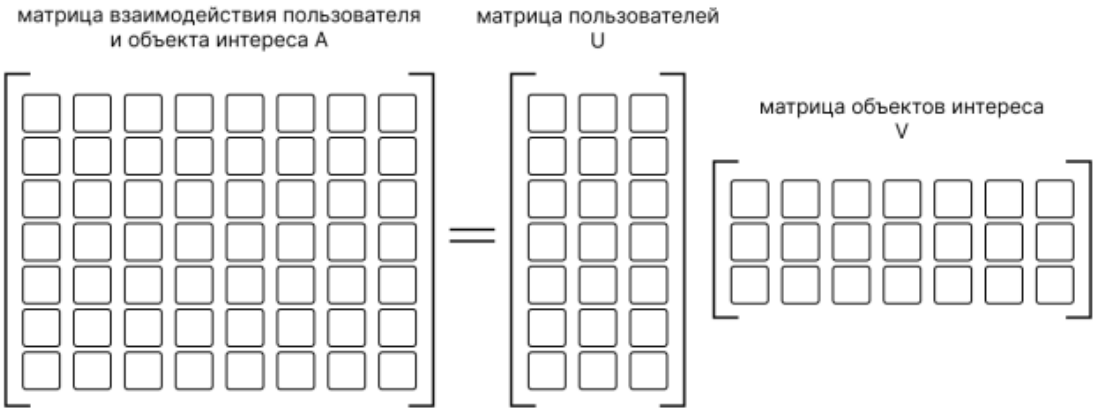


Рисунок 5 – Разложение матрицы рекомендаций

Релевантность определенного объекта интереса для конкретного пользователя рассчитывается как скалярное произведение полученных матриц.

Для повышения эффективности, а также уменьшения трудоемкости вычислений при большом наборе данных используется метод сингулярной декомпозиции значений (SVD) [5].

Методы SVD обеспечивают хорошие результаты прогнозирования за счет выявлении латентных семантических факторов. Применение данного механизма в рекомендательной системе позволяет предсказать оценки, которые могут выставить пользователи конкретному товару или услуге.

Теорема сингулярного разложения значений подразумевает, что любую матрицу A, имеющую размеры $n \times m$ и ранг $r = ran(A) \leq \min\{n, m\}$, можно разложить на произведение трех матриц U, Σ и $V\ T$:

$$A_{n\ x\ m} = U_{n\ x\ m} * \Sigma_{n\ x\ m} * V^{T}_{n\ x\ m}, \quad (2)$$

где $U \in Mat(n\ ,\ r)$ – матрица пользователей, столбцы которой представляют собой ортогональный базис столбцов матрицы A;

$V \in Mat(m\ ,\ r)$ – матрица объектов интереса столбцы которой представляют собой ортогональный базис строк матрицы A;

$\Sigma \in Mat(r\ ,\ r)$ – диагональная матрица, элементы которой отсортированы по убыванию и не являются нулевыми.

Метод факторизации с урезанным набором данных базируется на таком преобразовании ранга исходной матрицы, что $d < r$. Тогда диагональная матрица будет состоять только из d (ранг матрицы) наибольших диагональных элементов: $\Sigma d \in Mat(d,\ d)$. А матрицы объектов интереса и пользователей изменятся на $U \in Mat(n\ ,\ d)\ V \in Mat(m\ ,\ d)$.

Соответственно, при вычислении релевантности товаров или услуг как скалярного произведения матриц объектов интереса и пользователей будет иметь вид:

$$\tilde{A}_{n\ x\ m} = U_{n\ x\ d} * \Sigma_{d\ x\ d} * \ V^{T}_{d\ x\ m}\ . \qquad (3)$$

Выбор ранга матрицы d оказывает огромное влияние на точность прогнозов рекомендательной системы. Данное число должно быть, и в меру маленьким ввиду необходимости ускорения вычисления, и достаточно большим, чтобы в матрицу были включены важные взаимосвязи между объектами интереса и потребителями.

Исходя из этого, можно сделать вывод, что использование ограниченного набора наиболее схожих объектов и субъектов системы, а также замена значений элементов матрицы объектов интереса и пользователей меньшего ранга не только упрощает вычисления, но и повышает эффективность прогнозирования ввиду уменьшения влияния факторов переобучения модели.

К достоинствам использования моделей матричной факторизации в рекомендательных системах можно отнести: − уменьшение сложности исходной задачи, − обеспечение компактного представления данных взаимодействия между пользователями и элементами системы на основе

поведенческих паттернов. Несовершенство данного механизма заключается в погрешностях прогнозов, обусловленных наличием неявного взаимодействия пользователя с объектами интереса.

3.2 Нейронные сети

Помимо традиционных эвристик, которые основаны на механизме поиска информации, используются иные методы, базирующиеся на 16 контентной модели рекомендаций. Среди них можно выделить искусственные нейронные сети, которые являются прототипом нервной системы человека, состоящей из нейронов. Биологические нейроны – это чувствительные клетки, которые обрабатывают, хранят и передают информацию с помощью электрических и химических сигналов. На разветвленных отростках нейрона, называемыми дендритами, располагаются синапсы, являющиеся точками контакта. Изменение силы синаптических связей между нейронами под действием внешних раздражителей приводит к приобретению человеком новых навыков.

Искусственные нейронные сети имитируют биологический процесс обучения. Нейроны в данном случае – это отдельные узлы, являющиеся единицами вычислений, которые получают входные данные от других нейронов, производят обработку полученной информацией и передают ее остальным.

Аналогично изменению силы синаптических связей между биологическими нейронами, изменение веса в искусственных нейронных сетях приводит к тому, что вычислительная функция может быть изучена, а следовательно, приобретен новый навык. Внешним раздражителем в данном случае выступает массив данных, который необходимо обучить. Исходя из этого, искусственный нейрон можно представить как некоторую нелинейную функцию с единственным аргументом – линейной комбинации входных сигналов. Обучение модели состоит в постепенном изменении веса при неверном предсказании с текущим набор весов [6].

Существует множество архитектур, используемых для организации связи между узлами, начиная от простого однослойного персептрона и заканчивая сложными многослойными нейронными сетями. Выбранная архитектура является залогом эффективности работы нейронной сети. Искусственная нейронная сеть состоит из нескольких слоев, которые можно разделить на три типа: входные, скрытые и выходные. Входные слои реагируют на исходные данные, поступившие в сеть, скрытые блоки получают взвешенный выходной сигнал от входных блоков, выходные узлы, в свою очередь, реагируют на взвешенный выход из скрытых блоков и генерируют конечные выходные сигналы.

Наиболее распространенным подходом является использование базовой модели персептрона. Модель персептрона – это линейный классификатор, имеющий простой и эффективный алгоритм обучения. В данном случае используется два слоя узлов, где сигналы распространяются строго одним способом: от входа к выходу. Входные узлы, число которых равно количеству исходных данных, получают и передают один числовой атрибут выходному узлу, который в базовой модели персептрона является единственным узлом, выполняющим математическую функцию на своих входах.

Формальное описание искусственного нейрона представлено на рисунке 6, где x – входные данные, w – вес синаптической связи, S – текущее состояние нейрона, Y – выходные данные, вычисляемые функцией.

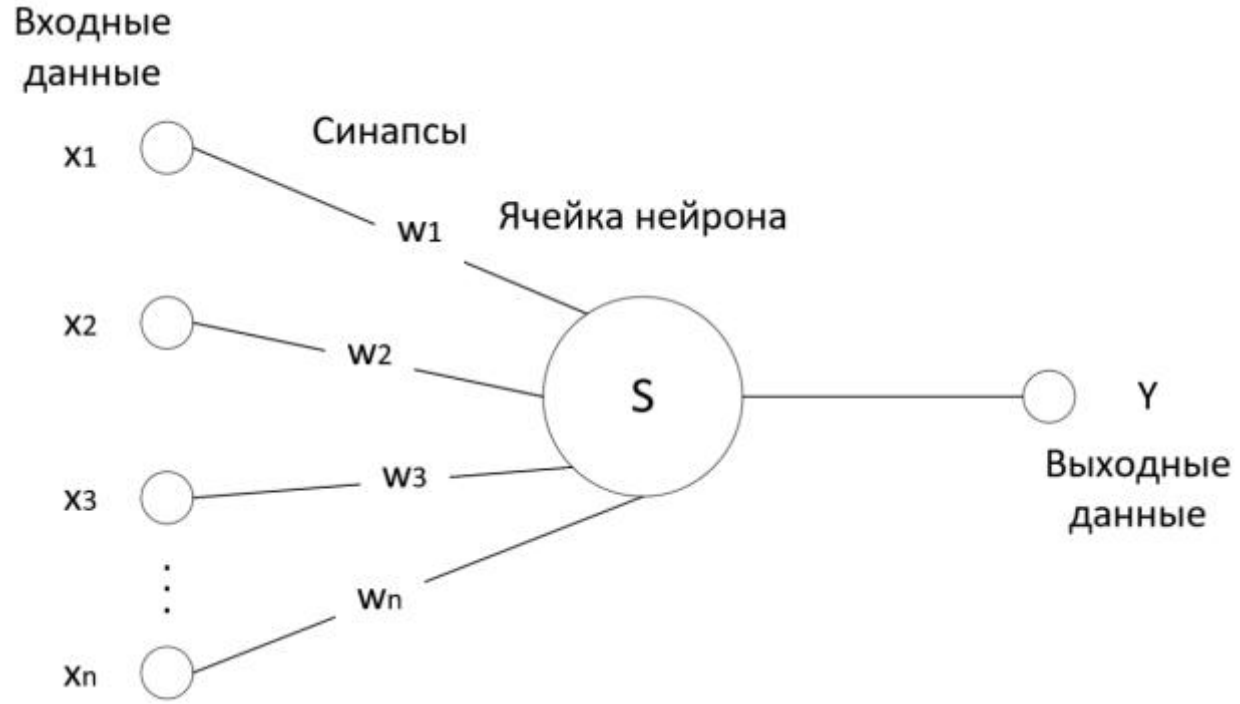


Рисунок 6 – Формальное описание искусственного нейрона в модели персептрона

Математический аппарат нейрона включает в себя:

− синапсы, характеризующиеся своим весом, осуществляют связь между нейронами, умножают входной сигнал на весовой коэффициент синапса,

− сумматор, аналог тела клетки нейрона. Выполняет сложение внешних входных сигналов, поступающих по синаптическим связям от других нейронов. Определяет уровень возбуждения нейрона,

− функция активации, определяет окончательные выходные сигналы. Обучение в персептроне осуществляется путем изменения весов связей, соединяющих входные узлы с выходными узлами, когда предсказанная метка не соответствует истинной метке [7].

Пусть, $\bar{W}$ – веса для соединений n различных входов с выходным нейроном для записи данных размерности n:

$$\bar{w} = (w_1 \dots w_n). \quad (4)$$

Кроме того, смещение n связано с функцией активации. Результат сложения $z_i \in \{-1, +1\}$ внешних входных сигналов ($xi\ 1 \ldots xi\ n$) из i− записи данных $\bar{X}\, i$ выглядит следующим образом:

$$z_i = sign\left\{\sum_{j-1}^{n} w_j * x_i^i + n\right\} = sign(\bar{W} * \bar{X}_i), \quad (5)$$

Значение z_i представляет собой прогноз персептрона для входного параметра $\bar{X}_i$. Однако, для обучения модели необходимо произвести подбор весов таким образом, чтобы значение z_i было равно значению y_i активационной функции.

При предсказании $(z_i - y_i)$ с текущем набором весов результат может принимать любое из значений –2, 0, +2. Цель алгоритмов нейронной сети состоит в том, чтобы узнать вектор весов $\bar{W}$ и смещение n, так, что z_i приближено к истинному результату y_i.

Алгоритм персептрона начинается с рандомного вектора весов, затем механизм последовательно заменяет элементы входных данных для создания следующего прогноза. Затем веса обновляются, исходя из значения предсказания. Обновление вектора весов $\bar{W}$ на t− итерации выглядит следующим образом:

$$\overline{W^{t+1}} = \overline{W^t} + \eta * (y_i - z_i) * \bar{X}_\iota, \quad (6)$$

где η – скорость обучения.

Значение η прямо пропорционально быстрым темпам обучения и обратно пропорционально оптимальности результирующих прогнозов. В действительности значения η в начале обучения выбирается наибольшим, постепенно уменьшая, поскольку результат сложения входных данных на выбранных весах приближается к своим наиболее благоприятным значениям. Идея данного подхода состоит в том, что высокие темпы обучения полезны

на ранних стадиях, однако могут привести к колебаниям между решениями на более поздних этапах.

Модель персептрона является базовой, однако она не всегда применима в сложных системах. В таком случае используются многослойные искусственные нейронные сети. Такие сети в своей архитектуре имеют скрытие слои помимо входных и выходных блоков. Сам скрытый слой, в свою очередь, может состоять из множества составных частей, в которых узлы связаны между собой. Многослойные нейронные сети не ограничиваются использованием линейных функций входных сигналов. Они могут использоваться в различных узлах скрытых слоев. В таком случае, вычисление сложения внешних сигналов выглядит следующим образом:

$$z_i = \sum_{j=1}^{n} w_j * \frac{1}{1+e^{x_j^i}} + n. \quad (7)$$

В данном случае z_i не является прогнозируемым, если оно вычисляется в узлах скрытого слоя, так как эти данные используются в дальнейшем.

При использовании многослойных нейронных сетей возникает проблема неизвестности выходных данных в скрытых слоях, поскольку обучающие метки этих узлов не определены. Из чего появляется необходимость составления корректного алгоритма обновления веса узлов в случае несовпадения результата сложения внешних входных сигналов и значения активизирующей функции. Проблема решается с помощью алгоритма обратного распространения.

В прямой фазе данного алгоритма входные данные подаются в нейронную сеть, происходит вычисления по слоям, используя текущей набор весов. Прогнозируемый результат сравнивается с меткой обучающего экземпляра для проверки их схожести. Далее, в обратной фазе, определяется погрешность вычислений в скрытом слое как функция оценок ошибок и весов узлов в предыдущем слое. Затем она используется при вычислении

погрешности относительно весов в узле и их обновления. Фактическое уравнение обновления весов в данном случае не очень отличается от базового персептрона.

Нейронные сети применяются в рекомендательных системах в двух вариациях:

1) механизм формирования рекомендаций,

2) промежуточный этап при построении моделей матричной факторизации.

В первом случае в качестве исходных данных предоставляются список объектов интереса и пользовательские оценки. Во время обучения модели вычисляемый результат сопоставляется с реальными предпочтениями, и в конечном итоге формирует принцип, на основе которого осуществляется составление рейтинговых прогнозов.

Основу второго подхода составляет матричная факторизация, роль нейронной сети в данном случае второстепенна. На начальном этапе происходит разложение исходной матрицы на двумерные массивы меньше размерности. Затем, полученные матрицы объединяются и подаются на вход в нейронную сеть. После чего, происходит стандартное обучение нейронной сети, целью которого является получение сложной нелинейной модели для факторизации матрицы.

На основе описанного механизма формирования рекомендаций при помощи нейронных сетей можно сделать вывод, что преимуществом их использования является высокий уровень эффективности, обусловленной работой даже в случае отказа некоторых фрагментов.

К недостаткам данного подхода можно отнести:

− сложность проектирования архитектуры сети,

− обучение сети в ряде случаев приводит к переобучению,

− продолжительные временные затраты на выполнение процесса обучения.

3.3 Байесовские классификаторы

Еще одним методом, базирующимся на контентной модели рекомендаций, являются байесовские классификаторы, представляющие собой вероятностную структуру, основанную на теореме Байеса и определения условной вероятности для решения задач классификации. К ним относят анализ данных и распознавание образов, требующих построения функции, которая присваивает метку класса экземплярам с собственным набором атрибутов. Модели байесовских классификаторов достаточно просты и эффективны при работе с большим массивом входных данных [8].

Как и нейронные сети, байесовский классификатор строится на основе обучения с учителем, когда вычисляемое значение сравнивается с меткой обучающего экземпляра. Частным случаем данного метода является наивный байесовский классификатор. Ключевая особенность данного механизма заключается в том, что каждый параметр классифицируемых данных рассматривается независимо от других, т.е. наличие или отсутствие одного атрибута не влияет на существование остальных. Благодаря данной специфики разреженность входных данных существенно не повлияет на рекомендательные прогнозы. Однако на практике анализируемые параметры не являются полностью независимыми. В таком случае используются байесовские сети, которые включают в себя ориентированный ациклический граф для кодирования зависимостей между атрибутами и таблицей вероятности, связывающей узлы между собой. Вершины данного орграфа соответствуют атрибутам, а ребра связывают данные переменные. Совместное условное распределение использует операторы независимости и определяется произведением локальных условных распределений, заданных в каждом узле. Полученная графическая модель дает полное представление о взаимосвязях всех компонентов системы.

Байесовский классификатор представляет собой вероятностную модель, где атрибуты объектов и метка класса рассматриваются как дискретные величины. Он запоминает все признаки, а также вероятности

того, что конкретный атрибут входит в конкретную классификацию (класс). Набор данных поступает в классификатор последовательно, после выполнения проверки наличия атрибутов в конкретном классе происходит обновление массива объектов интереса. Таким образом происходит обучение контентной модели.

Вероятностный классификатор предсказывает класс C_i с самой большой условной вероятностью P_i для заданного вектора признаков $x = (x_1 \dots x_n)$:

$$P\left(C_i \middle| x_i \& \dots x_n\right) = \frac{P(x_1 | C_i) * \dots * P(x_n | C_i) * P(C_i)}{P(x_1) * \dots * P(x_n)}. \quad (8)$$

Вероятность класса C_i находится на основе параметров $= (x_1 \dots x_n)$, подразумевая, что признаки являются независимыми, и следовательно, приведенная выше вероятность пропорциональна:

$$P(C_i) = \prod_{np} P\left(x_i \middle| C_i\right). \quad (9)$$

Исходя из этого, простой байесовский классификатор можно рассматривать как функцию, которая каждому выходному значению модели присваивает метку класса, т.е. $y = C_i$ следующим образом:

$$y = argmax \prod_{np} P\left(x_i \middle| C_i\right). \quad (10)$$

Из этого следует, что условная вероятность P_i , вычисленная для каждого класса C_i и признака x_i , присваивается к имеющему большую вероятность классу C_i . Результатом обучения контентной модели является список атрибутов вместе с условными вероятностями.

Несмотря на предположение, что признаки являются независимыми, байесовский классификатор обеспечивает высокую точность классификации.

Использование данного механизма в рекомендательных системах обусловлено преимуществом использования данного подхода, базирующегося на контентной модели рекомендаций, в том, что разреженность базы данных, а также наличие нерелевантных атрибутов игнорируются при вычислении вероятностной оценки.

К отрицательным сторонам можно отнести следующие аспекты:

− отсутствие в обучающем наборе данных некоторого признака приведет к его потери при формировании рекомендаций, поскольку, исходя из политики классификатора, анализируются только известные данные (явление известно под названием «нулевая частота»),

− значения спрогнозированных вероятностей не всегда являются достаточно точными,

− на практике допущение о полностью независимых признаках недопустимо.

4 Обзор существующих методов анализа туристических предпочтений на основе искусственного интеллекта

Алгоритмы машинного обучения значительно трансформировали сферу туризма, предоставляя инструменты для извлечения полезной информации из огромного объема данных. Эти данные включают в себя информацию о бронированиях, отзывах клиентов, паттернах путешествий и многое другое.

Два основных типа обучения машинного обучение с учителем и обучение без учителя. Они применяются для решения различных задач, таких как оптимизация предложений, улучшение качества обслуживания и персонализация пользовательского опыта.

Глубокое обучение, являющееся подмножеством машинного обучения, и подходы к обработке естественного языка дополнительно расширяют возможности анализа данных в туристической отрасли.

4.1 Обучение с учителем

Обучение с учителем - это подход к машинному обучению, при котором алгоритм обучается на заранее размеченных данных. То есть для каждого объекта из тренировочного набора данных известен целевой ответ. На основе этих данных алгоритм обучается находить закономерности, которые позволяют ему делать предсказания или принимать решения для новых данных.

Применения:

− прогнозирование спроса. Алгоритмы могут предсказать, какие туристические направления или услуги будут востребованы в определенное время года на основе исторических данных,

− рекомендация туров. Системы рекомендации могут предлагать пользователю туры исходя из его предпочтений, персональных данных и поведения в прошлом (например, предпочтения в отелях или активностях),

− определение ценовых стратегий. Анализирование влияния разных факторов на цены и оптимизация цен на билеты, номера в отелях и пакеты услуг в реальном времени,

− анализ отзывов. Анализ текста отзывов клиентов на отели, рестораны и туры с целью выявления положительных и отрицательных откликов для повышения качества услуг.

4.2 Обучение без учителя

Обучение без учителя используется для обработки данных, в которых нет явно размеченных примеров. Алгоритмы обучения без учителя поиск структур, шаблонов или кластеров в данных и используют эти знания для анализа новых данных.

Применения:

− сегментация клиентов. Кластеризация позволяет группировать клиентов по паттернам покупок, предпочтений, демографическим характеристикам для более эффективного целевого маркетинга и персонализации услуг,

− анализ маршрутов. Определение популярных маршрутов и комбинаций направлений среди туристов для оптимизации маршрутов и улучшения транспортной логистики,

− выявление аномалий. Нахождение нетипичных паттернов поведения, которые могут указывать на мошенничество или другие проблемы в сфере туризма,

− управление инвентарем. Оптимизация запасов и распределение ресурсов на основе обнаруженных закономерностей спроса и предпочтений клиентов.

4.3 Глубокое обучение и его применение в анализе поведения туристов

Глубокое обучение имеет ряд преимуществ перед традиционными методами машинного обучения, включая способность автоматически определять сложные характеристики данных и строить абстрактные

представления. Это особенно ценно в сфере туризма, где данные часто являются многообразными и неструктурированными. Применяются различные архитектуры глубоких нейронных сетей, включая сверточные нейронные сети (CNN) для обработки изображений, рекуррентные нейронные сети (RNN) для анализа последовательностей и сети с долгой кратковременной памятью (LSTM) для прогнозирования временных рядов.

4.4 Использование API для доступа к актуальной информации об услугах

Интеграция данных и системы поддержки принятия решений являются неотъемлемой частью многих современных бизнес− процессов. Их роль особенно актуальна в условиях постоянно растущего объема данных и необходимости оперативного их использования для повышения эффективности работы предприятий и организаций. Одним из ключевых компонентов, обеспечивающих эффективную работу таких систем, является использование API (Application Programming Interface) для доступа к актуальной информации об услугах.

API представляет собой набор правил и спецификаций, которые позволяют разным программным продуктам общаться между собой. В контексте интеграции данных и систем поддержки принятия решений, API позволяет автоматизировать процесс получения данных из различных внешних источников, таких как веб-сервисы, базы данных, CRM и ERP системы, а также социальные сети и облачные сервисы.

− эффективность и экономия времени. Автоматизация процесса получения данных с помощью API позволяет сократить время, затрачиваемое на ручной сбор информации. Это позволяет переключить ресурсы на более важные задачи, такие как анализ данных и принятие решений,

− уменьшение ошибок. Ручной сбор данных часто подвержен ошибкам из-за человеческого фактора. Использование API уменьшает вероятность таких ошибок, так как данные передаются непосредственно между системами,

− гибкость. API предоставляет гибкость в выборе источников данных и позволяет быстро интегрировать новые сервисы по мере их появления,

− актуальность данных. Данные, получаемые через API, обычно предоставляются в реальном времени, что обеспечивает их актуальность и повышает эффективность принятия решений,

− масштабируемость. По мере роста компании и увеличения объема работ, API позволяют масштабировать системы, добавляя новые источники данных без необходимости полной переработки системы.

Примеры использования API:

− финансовые сервисы. API позволяют получать актуальную информацию о курсах валют, акциях, бондах и других финансовых продуктах для проведения анализа рынка и принятия инвестиционных решений,

− логистика. Оптимизация маршрутов, отслеживание товаров и управление запасами могут быть облегчены использованием API для интеграции информации от транспортных компаний, поставщиков и складов.

− маркетинг. Сбор и анализ данных о поведении пользователей на веб−сайтах и в приложениях, а также о результатах рекламных кампаний, часто осуществляется с помощью API, интегрированных с различными маркетинговыми инструментами и платформами,

− управление клиентами. API позволяют интегрировать данные из социальных сетей и других каналов общения с клиентами для улучшения уровня обслуживания и персонализации предложений,

− здравоохранение. Интеграция данных от множества устройств и систем, используемых в медицинских учреждениях, повышает качество и оперативность оказания медицинских услуг.

API играет центральную роль в интеграции данных и систем поддержки принятия решений. Благодаря гибкости, экономии времени, уменьшению ошибок, актуальности предоставляемых данных и масштабируемости, API предоставляют широкие возможности для

улучшения бизнес-процедур и увеличения конкурентоспособности компаний.

5 Проектирование рекомендательной системы

Туристическая система решает задачу выработки рекомендаций туристических активностей на основе предпочтений пользователей. Прогнозирование базируется на модели предметной области, которая основана на данных об объектах интереса и информации о пользовательском поведении в системе, полученных от туристической платформы в определённом формате. Схема обработки действий пользователей для составления прогнозов представлена на диаграмме кооперации (рисунок 7).

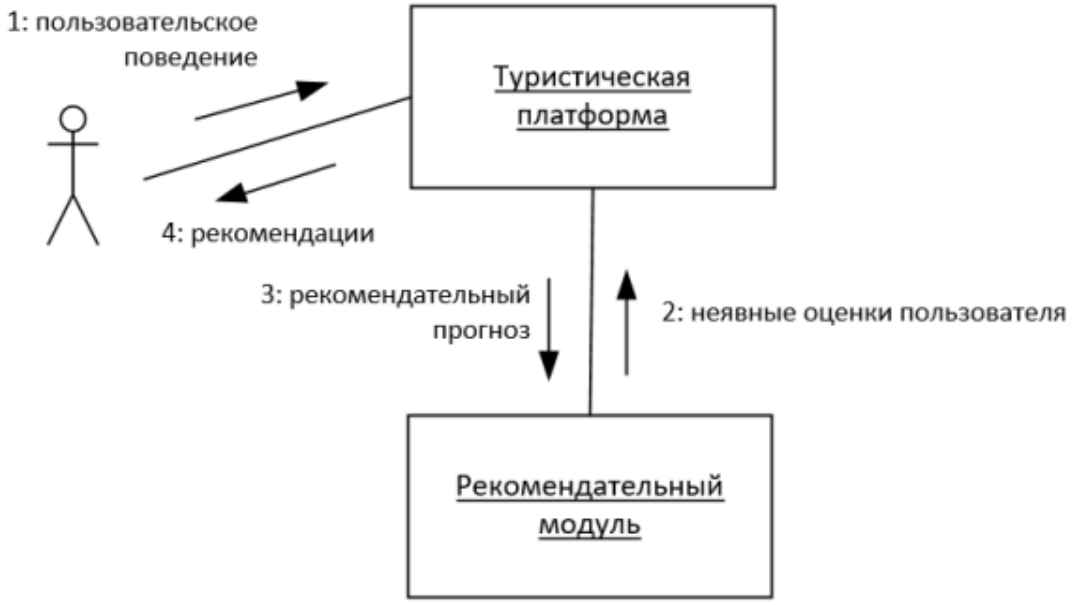


Рисунок 7 – UML-диаграмма кооперации

Целевая туристическая платформа направлена на развитие внутреннего туризма, предоставляя пользователям обширный перечень маршрутов, актуальных событий, туристических пакетов и разнообразных локаций. Функционал системы позволяет осуществлять поиск, добавление в избранное, а также бронирование и покупку объектов интереса по следующим категориям:

− туры,

− места,

− рестораны,

− события,

− маршруты.

Объекты интереса, в зависимости от типа, имеют свои характеристики, позволяющие получить полную информацию о конкретной активности. Некоторые из них являются универсальными, т.е. не зависят от категории, в которую входит объект. К ним относят:

− наименование,

− описание,

− список изображений, включая картинку предварительного просмотра,

− местоположение.

Другие параметры характерны только для определенного типа. Объекты интереса, относящиеся к категории «туры», помимо общих характеристик, содержит информацию о параметрах тура, включающих в себя тариф, сезонность, возрастные ограничения, квоту на количество человек, длительность и рейтинг места проживания. Кроме того, данная активность предоставляет данные о программе тура, дополнительных условиях и перечне услуг, входящих в стоимость.

Активности категории «места» предоставляют данные о режиме работы, примерном времени проведения, прейскурант на доступные билеты, точную геолокацию, а также места, находящиеся рядом с данным объектом интереса.

Информация о ресторанах содержит тип и кулинарные направления, средний чек, режимы работы и среднее время пребывания в данном месте, а также заведения, располагающиеся неподалеку.

К данным категории «события» относят приблизительную длительность события, стоимость билетов и расписание сеансов.

Объекты интереса категории «маршруты» содержат параметры, характеризующие продолжительность маршрута, его тип и состав, стоимость посещения, точную геолокацию, а также количество заинтересованных путешественников.

Обобщая имеющуюся информацию о данных, хранящихся в системе и вариантах ее использования, наиболее подходящим для составления рекомендаций была выбрана реализация системы в виде ИИ-ассистента на базе чат-бота в мессенджере telegram, что позволит использовать сервис как с десктопной версии, там и мобильного приложения.

5.1 Выбор инструментов разработки

5.1.1 Выбор мессенджера

Основной причиной выбора Telegram в качестве платформы для создания чат-бота является его современная, исчерпывающая и хорошо документированная API для разработки ботов. Также, по сравнению с другими мессенджерами, Telegram предлагает больше возможностей как для пользователей, так и для разработчиков, что делает его привлекательным для создания и использования чат-ботов.

Наличие понятной и хорошо документированной API делает процесс разработки чат-бота в Telegram относительно простым и прямолинейным. Благодаря доступу к широкому спектру функций и возможностей, разработчики могут создавать более сложные и функциональные боты, соответствующие потребностям пользователя. Более того, Telegram предоставляет разработчикам инструменты и руководства, которые помогают в быстрой и эффективной разработке чат-ботов.

Telegram является одним из самых популярных мессенджеров в мире с более чем 400 миллионами активных пользователей. Это позволяет чат− ботам Telegram достичь широкой аудитории пользователей по всему миру. Более того, поскольку пользователи Telegram привыкли к использованию чат− ботов, они скорее всего будут открыты для взаимодействия с новыми и инновационными ботами.

5.1.2 Выбор ИИ – ассистента

ChatGPT4 подходит для создания чат-бота по нескольким причинам. Во− первых, этот AI обучен на обширном корпусе данных, включающем

многочисленные разговоры, что позволяет ему улавливать и воспроизводить оттенки человеческого общения.

Во-вторых, ChatGPT4 способен поддерживать длинные и согласованные разговоры, соединяя несколько предшествующих запросов как единое целое, что делает его поведение более связным и предсказуемым для пользователей.

Третья причина - модель имеет мощные средства для генерации ответов на основе контекста, что позволяет ей быть более адаптивной и персонализированной. Она может анализировать предыдущие запросы пользователя и отвечать на них соответствующим образом, невзирая на сложность или длину вводимых данных. Это чрезвычайно полезно для чат− ботов, которые обязаны иметь возможность адаптироваться к любой данной ситуации.

Четвертая причина - ChatGPT4 может быть подстроен под конкретные потребности и предпочтения, позволяя разработчикам более тонко настроить его поведение в соответствии с деталями проекта, что делает его универсальным решением для различных приложений на основе чат− ботов.

5.1.3 Выбор языка программирования и среды разработки

В качестве языка разработки был выбран Python − это один из самых популярных языков программирования, который идеально подходит для разработки чат-ботов для Телеграма по причине простоты и чистоты кода. Python очень прост и читаем, что облегчает разработку и отладку. Код на Python обычно занимает меньше строк, чем на многих других языках, что значительно сокращает время разработки,

– библиотеки и фреймворки. Python имеет огромное сообщество разработчиков, которые создали множество библиотек и фреймворков, что заметно упрощает работу. В частности, существуют специализированные библиотеки для работы с API Телеграма, такие как python− telegram− bot.

– мультиплатформенность. Python-кроссплатформенный язык. Это означает, что чат-боты, написанные на Python, могут быть запущены на любой операционной системе, включая Linux, Windows и macOS,

– интеграция с внешними API. Python обладает мощными возможностями для интеграции с внешними API и веб-сервисами,

– асинхронность. Python поддерживает асинхронное программирование, что позволяет улучшить производительность чат-ботов Телеграма, обрабатывающих большое количество запросов.

В качестве среды разработки был выбран PyCharm − это оптимальная среда разработки (IDE) для Python по следующим причинам:

– интеграция со всеми нужными инструментами. PyCharm прекрасно интегрируется со всеми популярными фреймворками и библиотеками Python, а также с системами контроля версий, сервисами непрерывной интеграции и т.д.,

– мощный отладчик. PyCharm содержит настраиваемый отладчик для Python, что является ключевым компонентом при создании чат-ботов,

– Поддержка юнит-тестирования. Quality assurance важна при разработке чат− ботов, поэтому поддержка PyCharm различных фреймворков модульного тестирования Python, таких как pytest, очень полезна,

– удобство работы. PyCharm имеет множество функций, которые упрощают работу разработчика, например, автодополнение кода, подсветка синтаксиса, автоматическую корректировку и очень гибкую систему настроек.

– удобство отслеживания ошибок. PyCharm предоставляет возможность "на лету" проверки кода на ошибки, а также имеет интегрированный PEP8-линтер, который помогает делать код более понятным и красивым,

В целом, сочетание Python и PyCharm − это мощный и удобный инструмент для разработки чат-ботов в Телеграме, предлагающий широкий функционал, удобство работы и быстроту разработки.

5.2 Исходные данные для построения системы рекомендательных сервисов

Для построения рекомендательной системы необходимы данные, на основе которых будет осуществляться прогнозирование. Для формирования рекомендаций объектов интереса конкретного типа в туристической платформе запрашивается следующая информация:

− список туров (tours),

− список ресторанов (restaurants),

− список городов (cities),

− список маршрутов или пакеты (route),

− список мест (places),

− список событий (events),

− список подборок (collections),

− список тегов (tags).

Исходные данные можно представить в виде реляционной модели, состоящей из следующего набора связанных между собой отношений (таблиц):

− мест (Places),

− событий (Events),

− типов событий (Event_types),

− ресторанов (Restaurants),

− кулинарных направлений (Cuisines),

− партнеров (Parner_types),

− городов (Cities),

− регионов (Regions),

− тегов (Tags),

− туров (Tours),

− маршрутов (Routes),

− пользователей (Users).

Подробное описание атрибутов таблицы «Places» представлено в таблице 5.

Таблица 5 – Структура таблицы «Places»

Атрибут	Описание атрибута
id	идентификатор места
title	наименование места
address	адрес места
coordinates	координаты места
district_ids	идентификатор районов
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
tags_ids	идентфикаторы тегов
metro_ids	идентификаторы метро

Подробное описание атрибутов таблицы «Events» представлено в таблице 6.

Таблица 6 – Структура таблицы «Events»

Атрибут	Описание атрибута
city	идентификатор города
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
cg_announcement	рекламный слоган
link_source	список картинок данного ссылка но событие
ticket_price	стоимость билета
duration	длительность события
events_types_ids	идентификаторы типов событий
sort	номер сортировка

Подробное описание атрибутов таблицы «Events_types» представлено в таблице 7.

Таблица 7 – Структура таблицы «Events_types»

Атрибут	Описание атрибута
id	идентификатор типа события
title	наименование типа события
image	список картинок данного типа события

Подробное описание атрибутов таблицы «Restaurants» представлено в таблице 8.

Таблица 8 – Структура таблицы «Restaurants»

Атрибут	Описание атрибута
id	идентификатор ресторана
title	наименование ресторана
address	адрес ресторана
avg_time_visit	режим работы
city	идентификатор районов
coordinates	список картинок данного места
can_reserve	картинка предварительного просмотра места
cuisines_ids	идентификаторы кулинарных направлений
images	список картинок данного ресторана
image_preview	картинка предваритедьного просмотра ресторана

Подробное описание атрибутов таблицы «Cuisines» представлено в таблице 9.

Таблица 9 – Структура таблицы «Cuisines»

Атрибут	Описание атрибута
id	идентификатор кулинарного направления
title	наименование кулинарного направления
address	адрес места
coordinates	координаты места

Подробное описание атрибутов таблицы «Parner_types» представлено в таблице 10.

Таблица 10 – Структура таблицы «Parner_types»

Атрибут	Описание атрибута
id	идентификатор партера
title	наименование партнера
images	список картинок данного партнера

Подробное описание атрибутов таблицы «Cities» представлено в таблице 1

Таблица 11 – Структура таблицы «Cities»

Атрибут	Описание атрибута
id	идентификатор города
list_image	список картинок города
rating	рейтинг города
region	регион
short_description	описание

Подробное описание атрибутов таблицы «Regions» представлено в таблице 12.

Таблица 12 – Структура таблицы «Regions»

Атрибут	Описание атрибута
id	идентификатор региона
titile	наименование региона

Подробное описание атрибутов таблицы «Tags» представлено в таблице 13.

Таблица 13 – Структура таблицы «Tags»

Атрибут	Описание атрибута
id	идентификатор тега
list_image	список картинок тега
count	количество объектов
title	наименование тега

Подробное описание атрибутов таблицы «Tours» представлено в таблице 14.

Таблица 14 – Структура таблицы «Tours»

Атрибут	Описание атрибута
id	идентификатор тура
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
cg_announcement	рекламный слоган
link_source	список картинок
ticket_price	стоимость тура
duration	длительность тура
events_types_ids	идентификаторы типов туров
sort	номер сортировка

Подробное описание атрибутов таблицы «Routes» представлено в таблице 15.

Таблица 15 – Структура таблицы «Routes»

Атрибут	Описание атрибута
id	идентификатор места
title	наименование места
address	адрес места
coordinates	координаты места
district_ids	идентификатор районов
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
tags_ids	идентфикаторы тегов
metro_ids	идентификаторы метро

Таблицы «Places», «Events» и «Routes» связаны с отношением «Tags» по атрибутам «tags_ids», «tags_ids» и «route_tags_ids» соответственно как один ко многим.

Отношения «Events» и «Event_types», «Events» и «Places» связаны по атрибутам «event_types_ids» и «place_ids» соответственно как один ко многим.

Таблицы «Tours», «Routes», «Restaurants» и «Events» связаны с отношением «Cities» по атрибуту «city» как один ко многим.

Отношение «Cities», в сою очередь, связано с таблицей «Regions» по атрибуту «region» как один к одному.

Отношение «Restaurants» связано с таблицами «Cuisines» и «Partner_types» по атрибутам «cuisines_ids» и «partner_type_id» соответственно как один ко многим.

Взаимодействие между описанными выше отношениями, а также их атрибутами, представлено на рисунке 8.

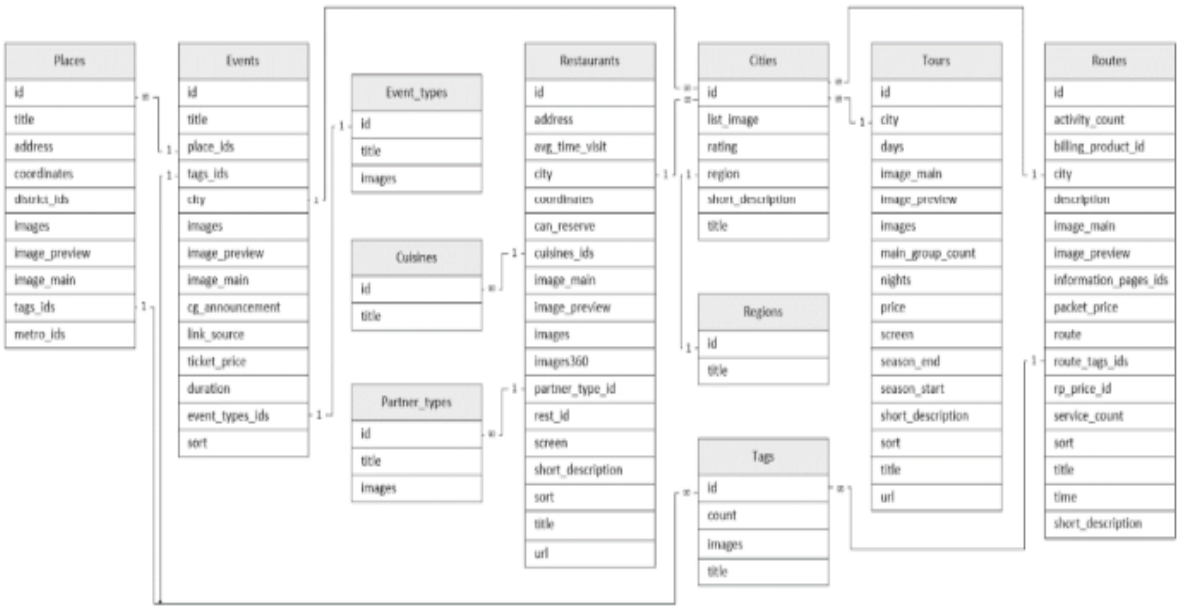


Рисунок 8 – Реляционная модель исходных данных

Отношение «Relationships» связано с таблицей «Users» и таблицами «Places», «Events», «Routes», «Tours» и «Restaurants» по атрибутам «uid» и «iid» соответственно как один ко одному.

Подробное описание атрибутов таблицы «Users» представлено в таблице 16.

Таблица 16 – Структура таблицы «Users»

Атрибут	Описание атрибута
id	идентификатор пользователя
titile	идентификатор пользователя в битриксе

Подробное описание атрибутов таблицы «Relationships» представлено в таблице 17.

Таблица 17 – Структура таблицы «Relationships»

Атрибут	Описание атрибута
iid	идентификатор объекта интереса
uid	идентификатор пользователя
rating	оценка

Взаимосвязи между объектами и субъектом туристической платформы можно представить в виде реляционной модели, представленной на рисунке 9.

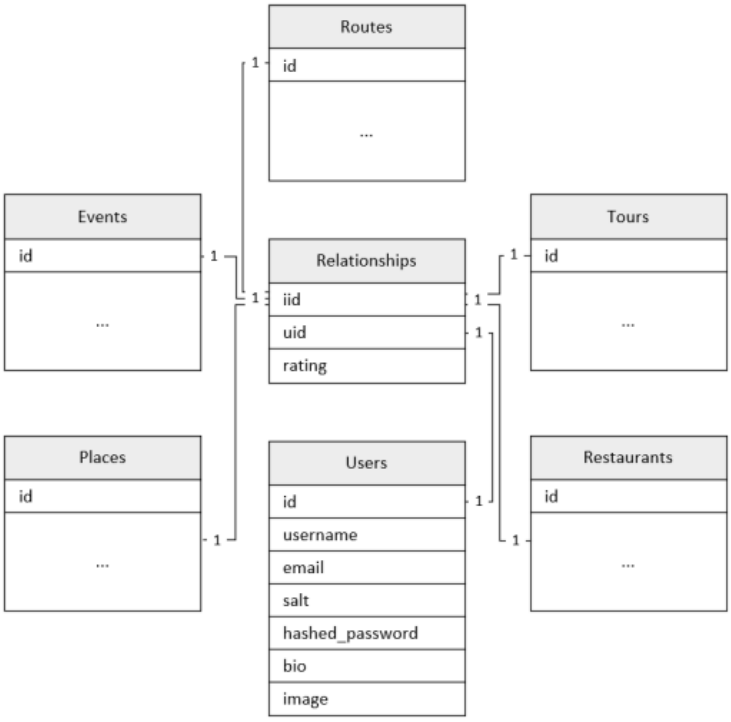


Рисунок 9 – Реляционная модель взаимосвязей между пользователями и объектами интереса

5.3 Создание чат-бота

Перед началом работы с чат-ботом, его нужно создать. Для этого необходимо зарегистрироваться с мессенджере Telegram и найти через окно поиска бота @BotFather, представленного на рисунке 10. Это официальный бот представленный командой мессенжера для создания чат-ботов/ веб-приложений, а так же игр на платформе. Для более удобного использования, рекомендуется установить декстопную версию приложения на свой компьютер с официального сайта Telegram.

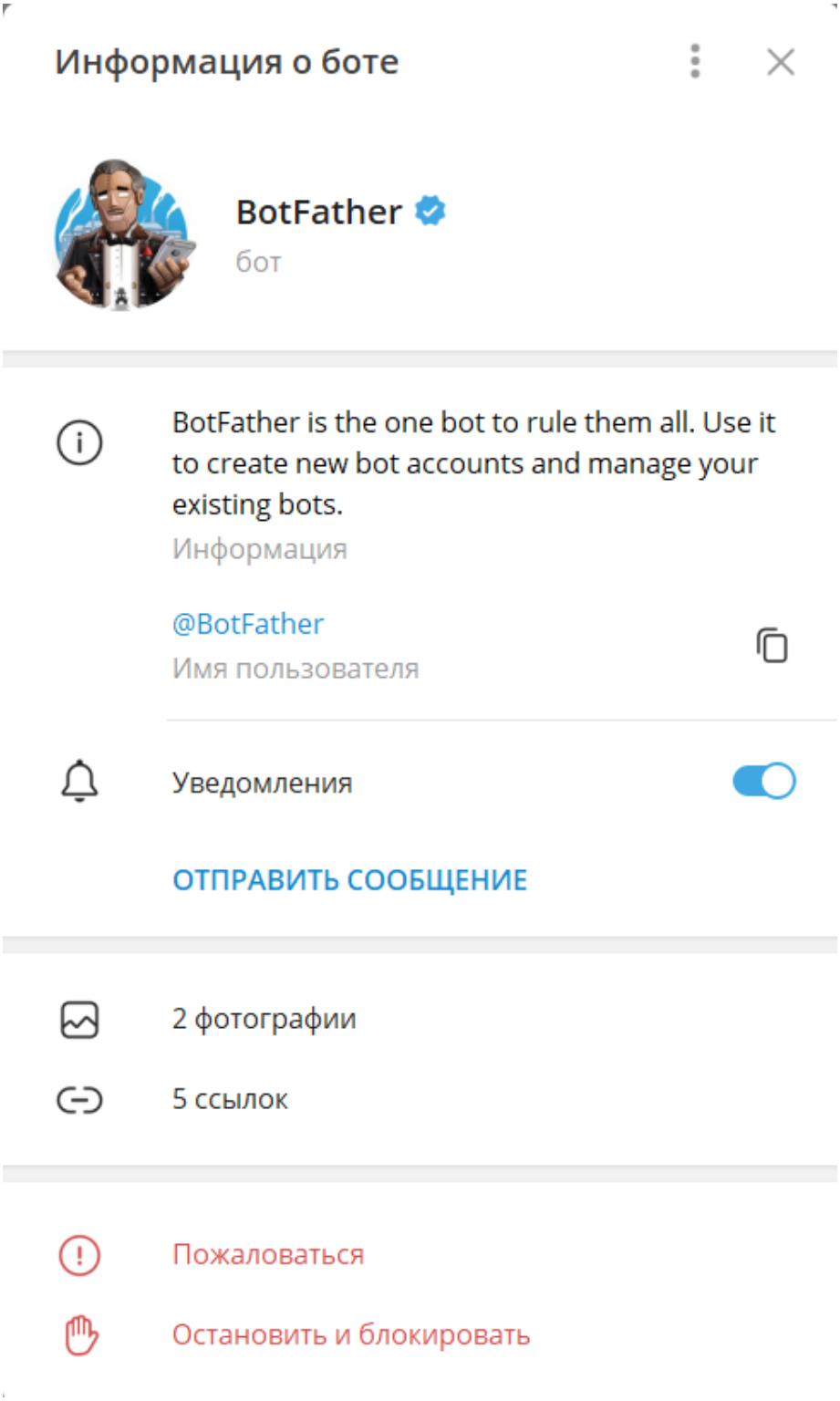


Рисунок 10 – BotFather

Для началы работы с ботом, необходимо отправить команду /start и получить в ответном письме информацию о доступных функциях и настройках. Возможности сериса достаточно обширны, поэтому рекомендуется внимательно ознакоситься со всеми представленными командами. Результат команды показан на рисунке 11.

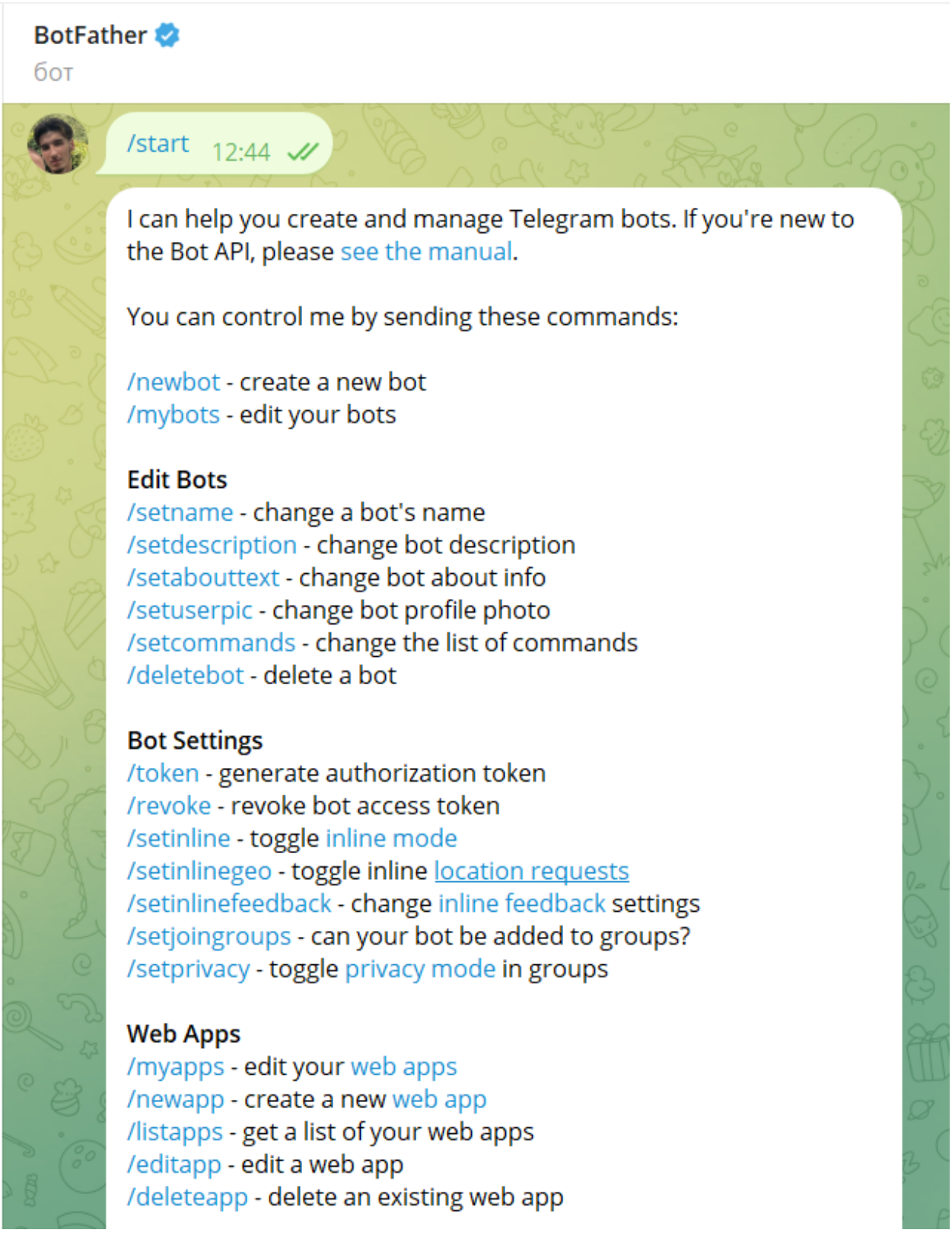


Рисунок 11 – Команда /start

Чтобы создать бота, нужно отправить команду /newbot и задать имя и уникальний идентификатор (ID). В случае, если ID уже занято, бот об этом сообщит и отправить письмо с предложением выбрать другой идентификатор. Процесс создания бота показан на рисунке 12.

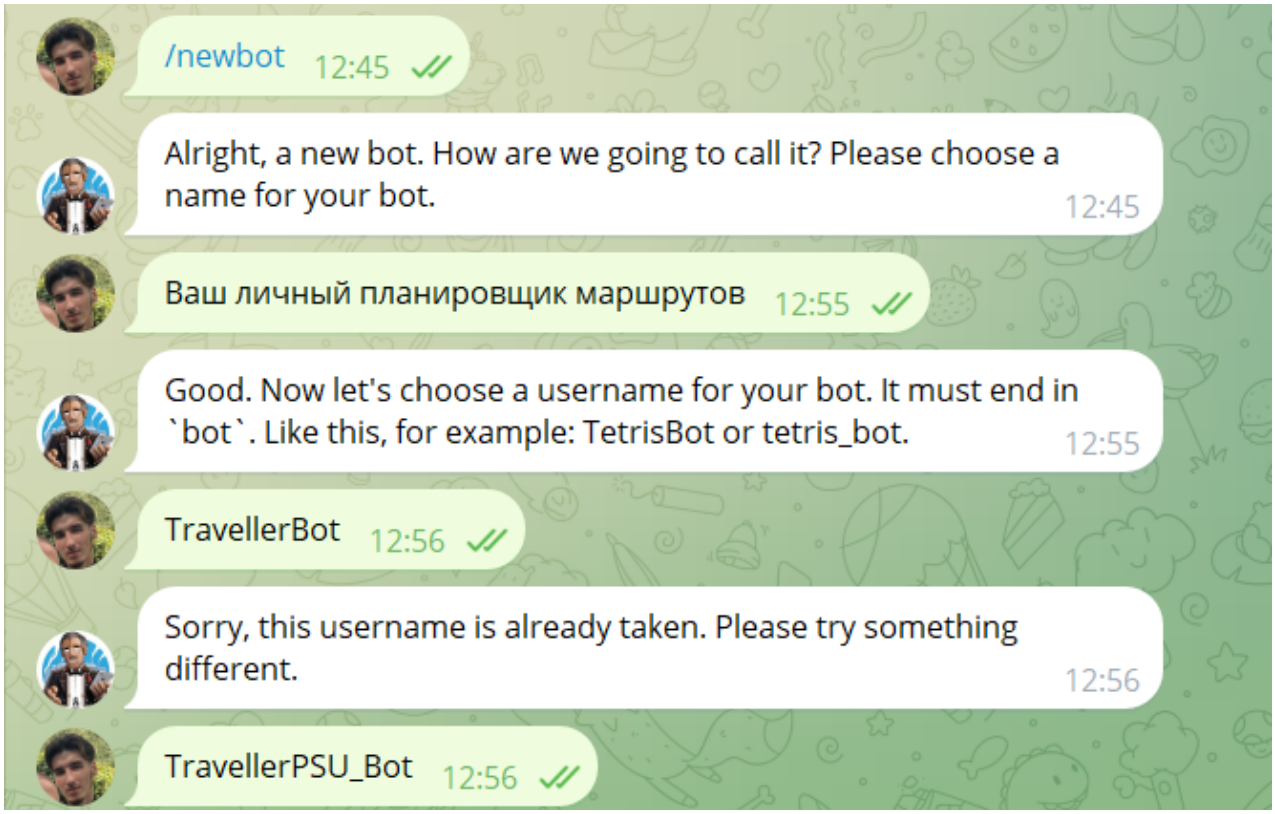


Рисунок 12 – Создание бота

Далее нам приходит сообщение с токеном бота, необходимым для дальнешего внедрений собственных функций, а так же подключения стороннего кода к боту. Полученное сообщение представленно на рисунке 13.

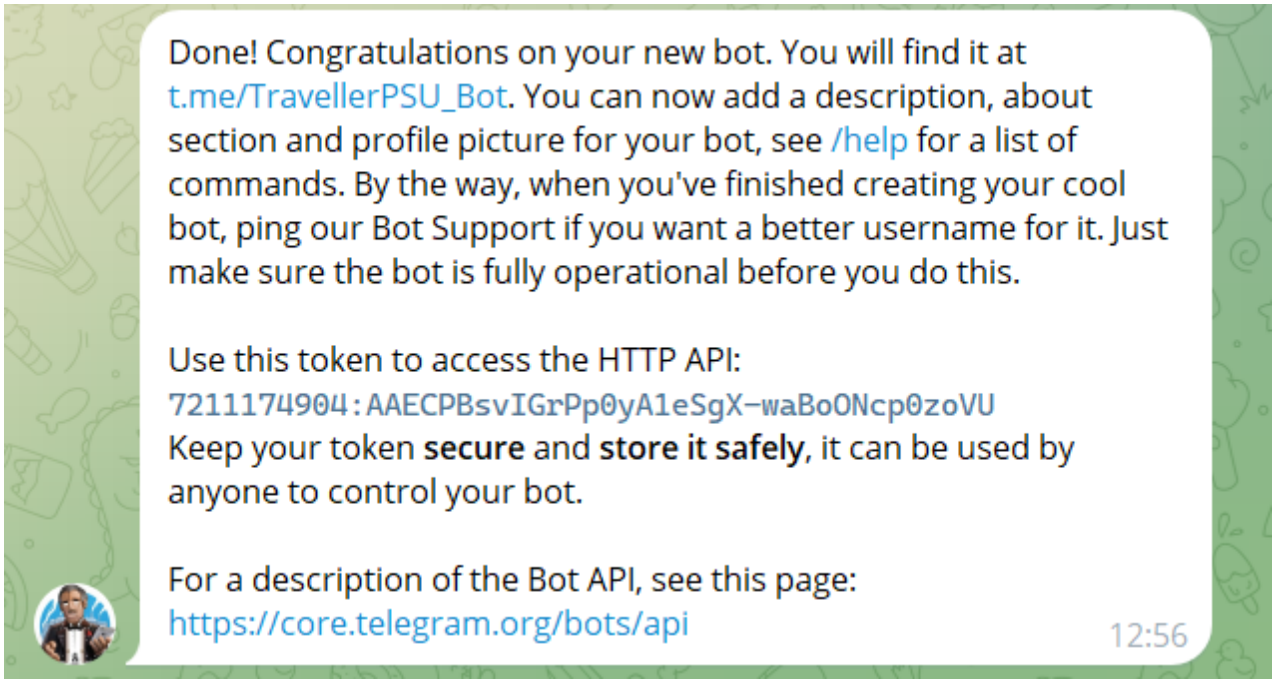


Рисунок 13 – Токен бота

Помиимо создания бота, BotFather позволяет загрузить собственное изображение которое пользователь увидит рядом с название чат− бота.

Процесс добавления изображения отображен на рисунке 14.

Рисунок 14 – добавление изображения

5.4 Интерфейс приложения и доступные функции

Интерфейс рекомендательной системы выглядит следующим образом. На рисунке 15 показан главный экран чат-бота, который видит пользователь при переходе. На экране отображено окно, в котором описываются возможность бота, поле для ввода текста, а так же кнопка вызова меню, через которую воспользоваться быстрыми командами, например /start, для начала работы с чат-ботом.

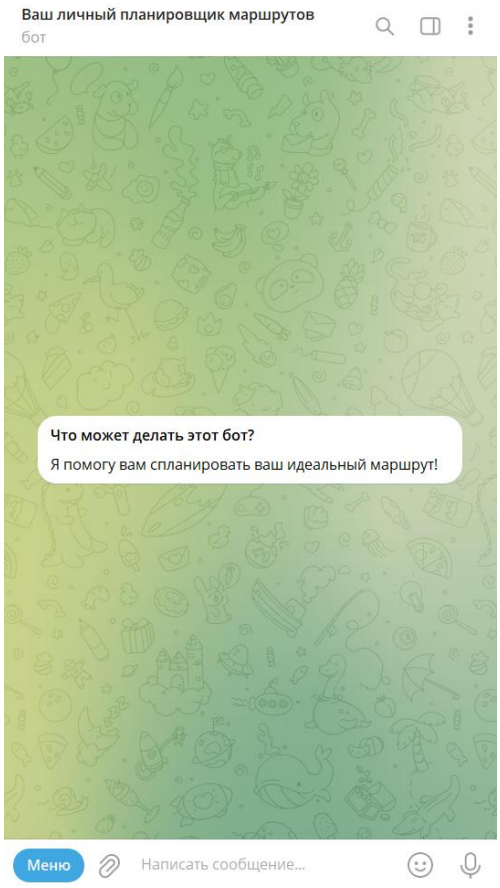


Рисунок 15 – Приветственное окно

При клике в верхнюю часть бота, появляется всплывающее окно с подробной информацией, изображенное на рисунке 16. В нем есть возможность поделиться ссылкой на бота, отключить уведомления и прочитать описание, которое можно задать в BotFather.

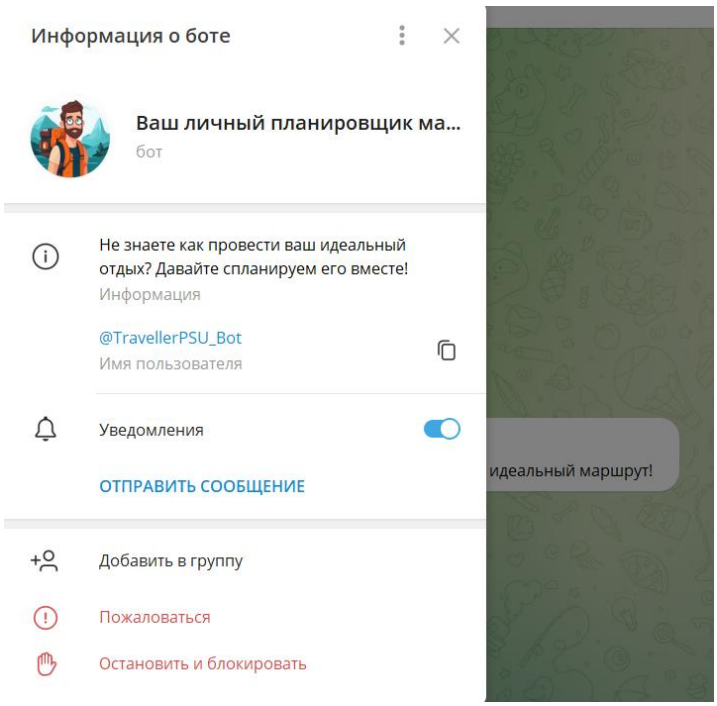


Рисунок 16 – Информация о чат-боте

Для начала взаимодействия с ассинстентом, необходимо написать команду /start или выбрать ее по клику на кнопку “меню”, после чего появится клавиши навигации и пользователь получит приветственное письмо, изображенное на рисунке 17.

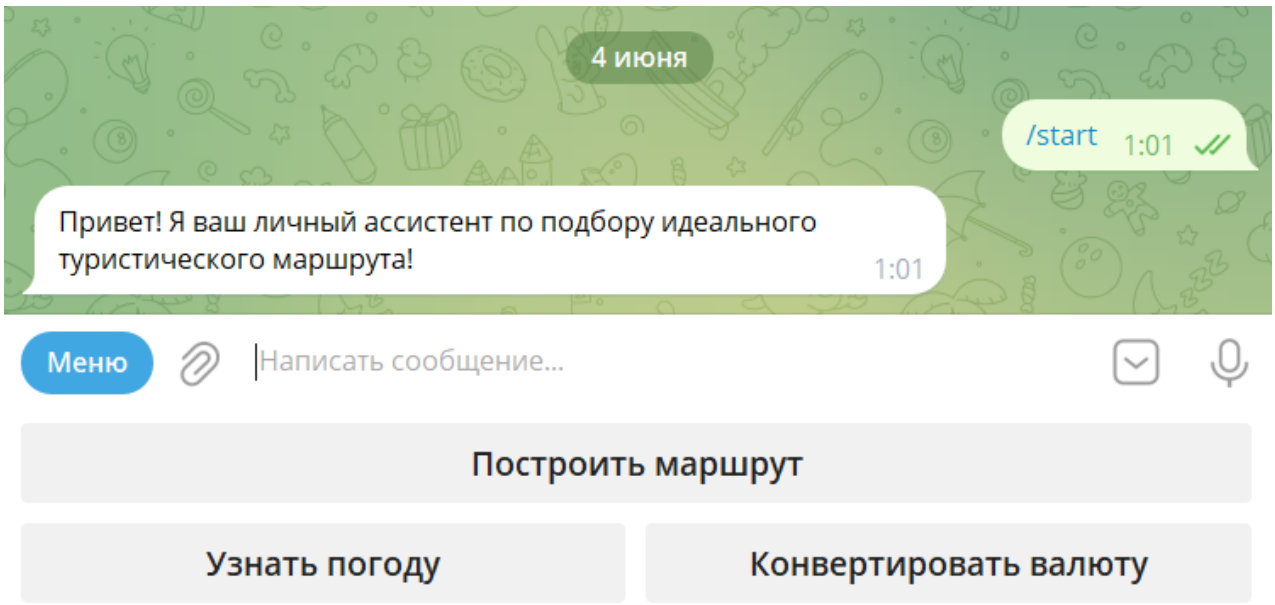


Рисунок 17 – Навигация

Присутствует 3 кнопки навигации. Первая – “Построить маршрут”, начинает диалог с ассистентом, который задает уточняющие вопросы, о

городе, интересующих мероприятиях, виде отдыха и бюджете. Начало построения маршрута показано на рисунке 18.

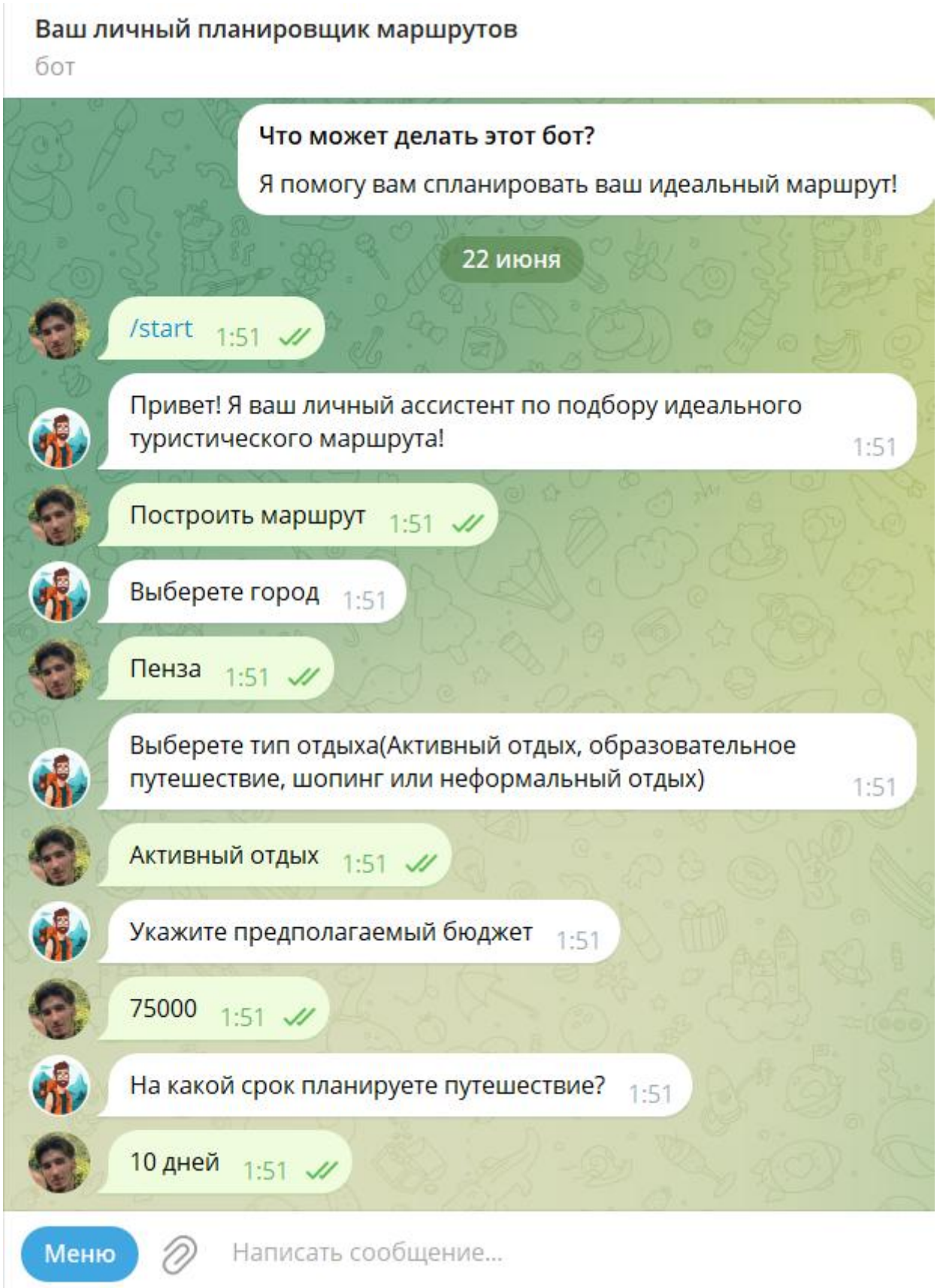


Рисунок 18 – Построение маршрута

После выполненных действия составляет маршрут по дням и часам с адресами заведений, музеев и прочих мест досуга, основанный на оценках пользователей благодаря методу коллабаративной фильтрации. Результат работы показан на рисунке 19.

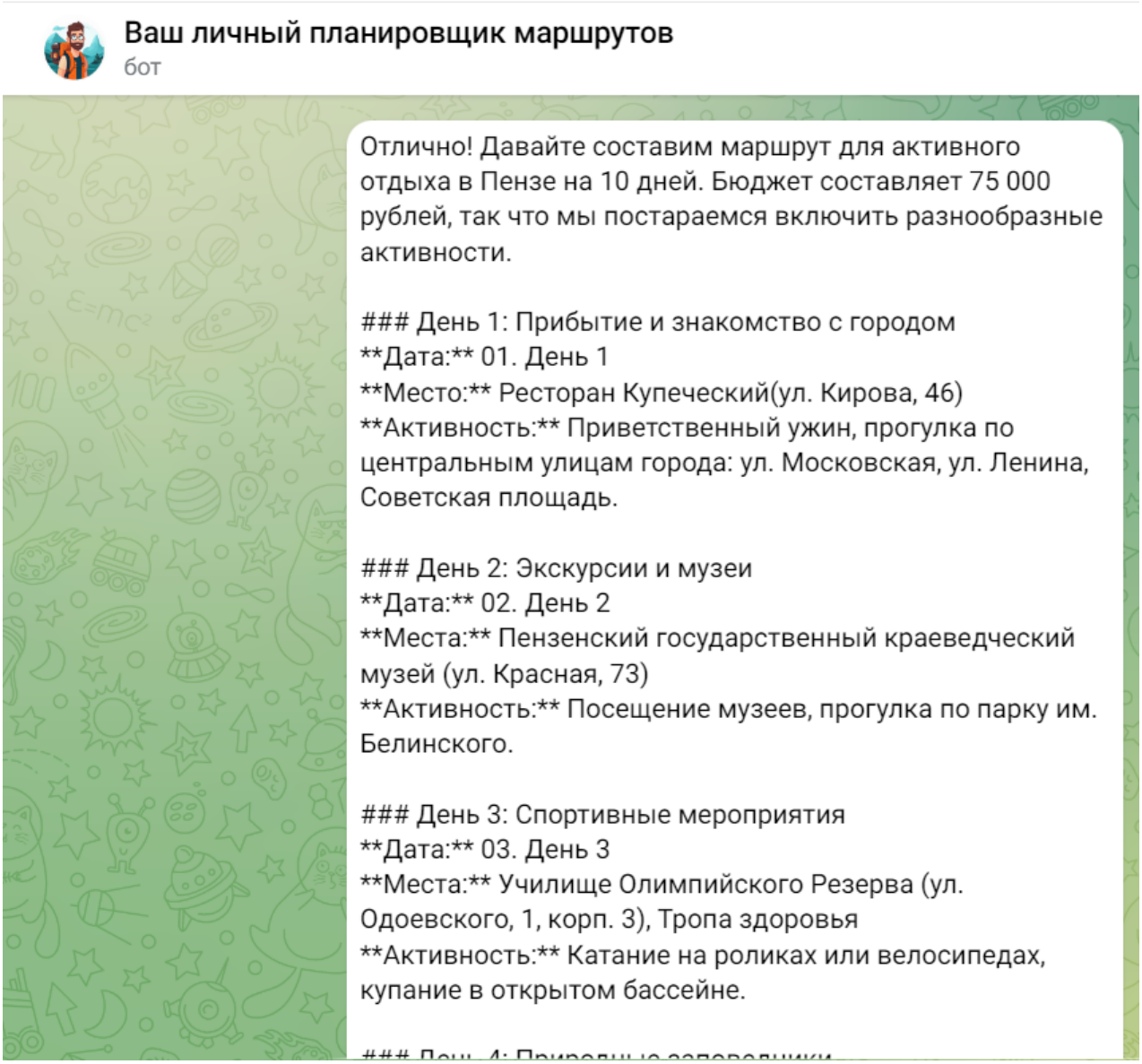


Рисунок 19 – Получение маршрута

После получение рекомендованного маршрута, мы можем задать чат–боту уточняющие вопросы по маршруту, по поводу маршрута, внести в него свои корректировки или получить дополнительную информацию по объектам, их истории и интересных фактах. Нейросеть в данном случае служит как ИИ– ассистент. Работа ИИ – гида показана на рисунке 20.

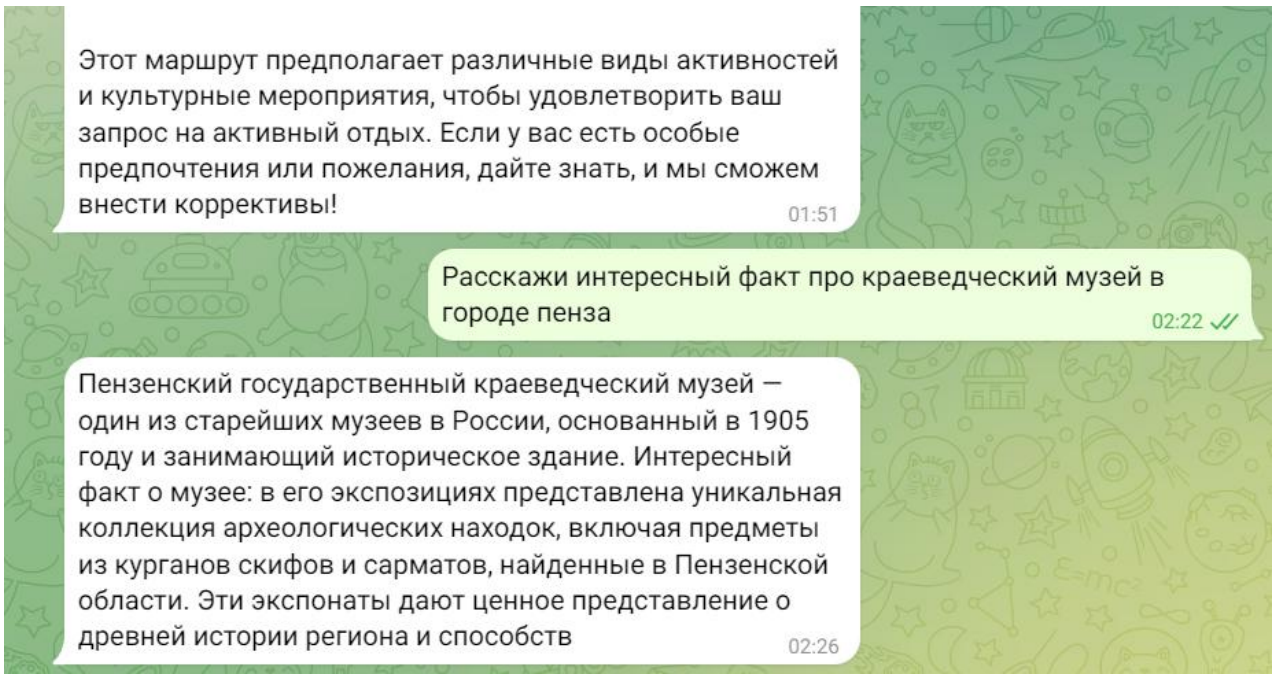


Рисунок 20 – Дополнительная информация

Помимо кнопки “Построить маршрут” в чат боте присутствует команда “Узнать погоду”, которая позволяет получить данные о погоде в интересующем городе в режиме реального времени. Для этого пользователю необходимо указать место, например “Пенза”, после чего бот выведет температуру и скорость ветра в режиме реального времени. Алгорит реализован на осново API сервиса OpenWeatherMap. Работа алгоритма изображена на рисунке 21.

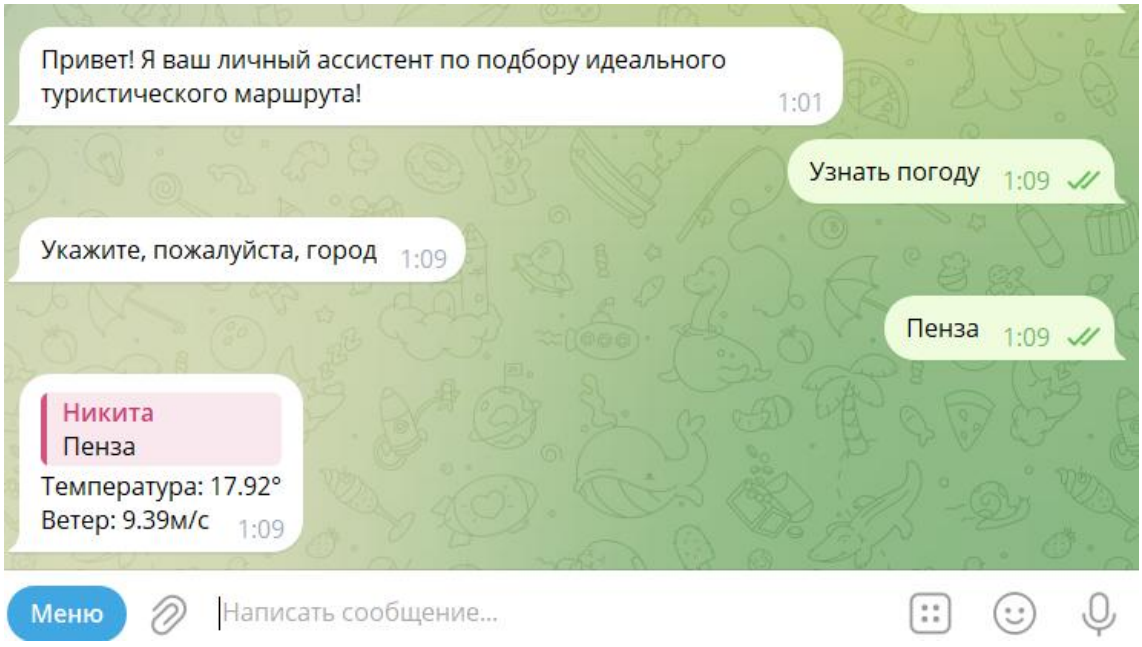


Рисунок 21 – Работа алгоритма по выводу погоды

Так же в боте есть третья кнопка, которая будет полезна при путешествии в другие страны. В ней реализова алгоритм конвертации валюты при помощи пакета Currency_Converter. При клике на кнопку, требуется указать сумму, которою необходимо конвертировать. На следующем шаге предоставляется выбор из наиболее популярных вариаций, а так же возможность создать собственную пару значений. Результат работы алгоритма указан на рисунке 22.

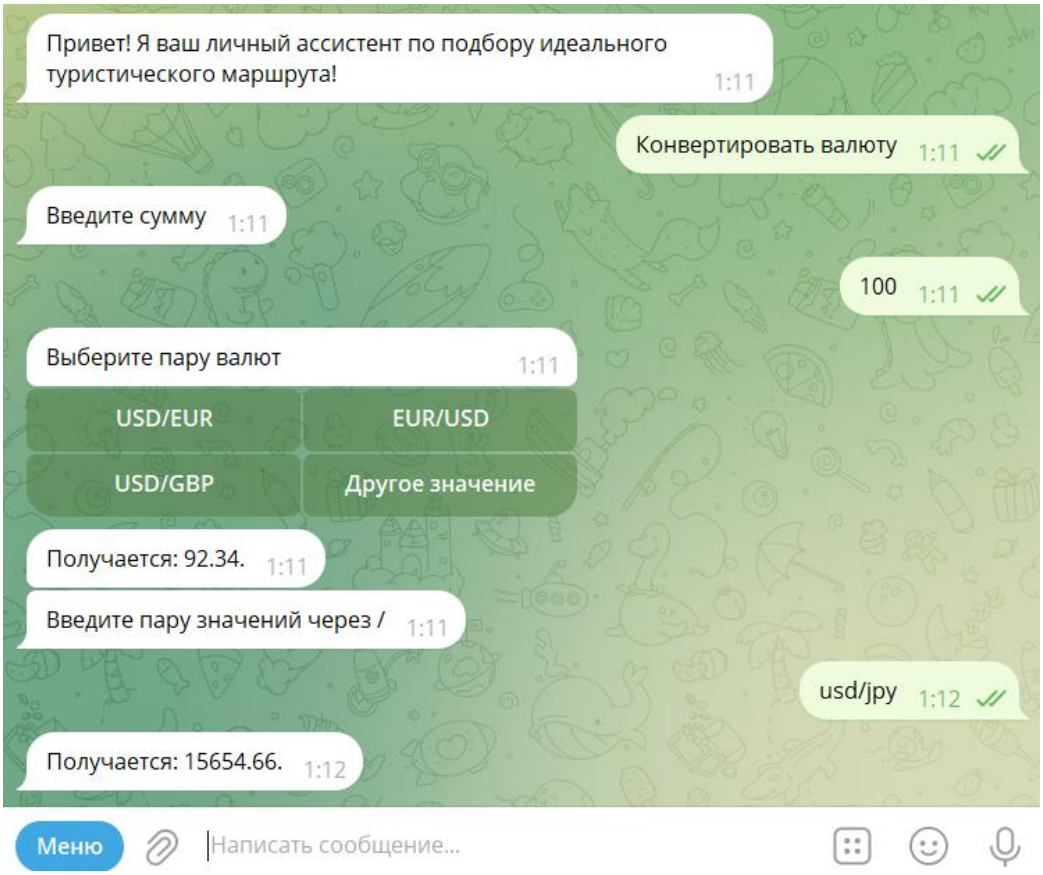


Рисунок 22 – Конвертер валюты

5.5 Описание программы

В первую очередь подключаем все необходимые библиотеки для работы с перечислениями, с данными в формате CSV, для работы с чат− ботами на Python, а также библиотека для работы с API OpenAI и API OpenWeatherMap. В данной работе используются библиотека enum, необходимая для определения перечислений, представляющих собой набор символических имен, библиотека time, используемая для создания задержек выполнения при ожидании выполнения операций. Помимо этого, из библиотеки Currency

Converter необъодимо импортировать класс CurrencyConverter. Он необходим для конвертации валюты в режиме реального времени по актуальному курсу. Библиотека aiogram необходима для асинхронной работы с ботами на базе telegram. Из библиотеки необходимо импортировать dispatcher – класс, необходимый для маршрутизации и обратки входящих сообщений и событий. Его задача – регистрация обработчиков для определенных типов сообщений. Также нужно импортировать types для работы с объектами и структурами данных из Telegram API. Для работы с ChatPGT 4 необходимо импортировать модуль Open AI из одноименной библиотеки. В рамках работы необходимо создать отдельный поток с созданным ассистентом, который будет опираться только на загруженный файл.

```
        import enum
        import time
        from aiogram import Dispatcher, types
        from openai import OpenAI
    from config import Settings
    from currency_converter import CurrencyConverter

        token          =     '7211174904:AAECPBsvIGrPp0yA1eSgX-
waBoONcp0zoVU'
    API = 'b82323a813707b7593ff50b5f17e7bd1'
```

После этого необходимо задать необходимые промпты в классе Prompts которые необходимы для изначальных настроек ИИ− помощнику. Промпт – обусловленный запрос к нейросети, запрошенный с целью получения ответа. В данной работе благодаря промптам производится первичная настройка интеллектуальной диалоговой системы.

```
    class Prompts(enum.Enum):
        """Промты"""
        MSG_PROMPT = ("Отвечай только на вопросы, связанные с
планированием туристических маршрутов и туризмом"
                      "Предоставляй ответы максимально
развернуто и без воды."
                      "(tourMod.csv).")
```

После этого необходимо создать класс MyOpenAI, состоящий из ряда методов, необходимых для интеграции ChatGPT 4 с интеллектуальной диалоговой системой вуза. Метод __init__ отвечает за инициализацию клиента OpenAI, устанавливая API− ключ доступа, а также путь к датасету, на основе которого чат− бот будет дополнительно выдавать ответы.

```
class MyOpenAI:
    """Класс для взаимодействия с OpenAI"""
    def __init__(self):
        self.client = OpenAI(api_key=Settings.OPENAI_TOKEN)
        self.csv_file =
"/home/PycharmProjects/projects/touristData.csv"
```

Метод test_answer_gpt создан для получения ответа от ChatGPT. В системе указано ключевое слово, указывая которое система создает отдельный поток с сущностью ассистента, который опирается сугубо на загруженный файл. Если в теле письма, который отправляет пользователь данный ключ не используется – вызывается стандартный ИИ- помощник.

```
    async def text_answer_gpt(self, message: str):
        """Ответ текстом от ГПТ"""
        if "psu" in message.split(" "):
            res = await self.thread_run(
                message,
                thread="thread_E3m22YEwMNVH1lnq3p5A2tuN",
                assistant_id="asst_qyUU2LBFQVv2q2aohxIpP3fs"
            )
            return res
        completion = self.client.chat.completions.create(
            model="gpt- 4o",
            messages=[
                {"role":        "system",        "content":
Prompts.MSG_PROMPT.value},
                {"role": "user", "content": message}
            ],
        )
        return completion.choices[0].message.content
```

Метод upload_file необходим для конвертирования CSV− файла в двоичный код, чтобы в дальнейшем передать его на сервер OPENAI. На выходе мы получаем ID загруженного файла.

```
def upload_file(self) - > str:
        """Загрузка файлов в OPenAI"""
        with open(self.csv_file, "rb") as binary_file:
            upload_file = self.client.files.create(
                file=binary_file,
                purpose="assistants"
            )
            return upload_file.id
```

После этого с помощью метода create_assistant необходимо создать ассистента, который на входе изначально получает конкретный датасет, на который будет опираться в виде идентификатора файла, загруженного ранее. В OPENAI можно обратиться как к главной модели, которая содержит в себе обширный датасет, так и создать отдельную версию и дополнительно обучить ее.

```
    def create_assistant(self, file_id: str):
        """Метод для создания ассистента"""
        assistant = self.client.beta.assistants.create(
            instructions=Prompts.MSG_PROMPT.value,
            model="gpt- 4o",
            tools=[{"type": "code_interpreter"}],
            tool_resources={
                "code_interpreter": {
                    "file_ids": [file_id]
                }
            }
        )
        return assistant.id
```

Метод creatr_thread необходима для создания потока, необходимого для вызова ассистента. Метод возвращает идентификатор потока.

```
def create_thread(self, file_id: str, message: str):
        """Метод для создания потока"""
```

```
            thread = self.client.beta.threads.create(
                messages=[
                    {
                        "role": "user",
                        "content": message,
                        "attachments": [
                            {
                                "file_id": file_id,
                                "tools":                [{"type":
"code_interpreter"}]
                            }
                        ]
                    }
                ]

            )
        return thread.id
```

Чтобы отправлять сообщения ассистенту в поток, необходимо реализовать метод thread_run, также в методе происходит ожидание ответа от ассистента после проверки статуса.

```
       async def thread_run(self, message: str, thread: str,
assistant_id: str):
            """Метод для отправки сообщения в поток с
ассистентом и файлом"""
            self.client.beta.threads.messages.create(
                thread,
                role="user",
                content=message
            )
            run = self.client.beta.threads.runs.create(
                thread_id=thread,
                assistant_id=assistant_id
            )
            get_status = self.client.beta.threads.runs.retrieve(
                run_id=run.id,
```

```
                thread_id=thread
            )
            while get_status.status != "completed" and
run.status != "failed":
                time.sleep(5)
                get_status =
self.client.beta.threads.runs.retrieve(
                    run_id=run.id,
                    thread_id=thread
                )
            assistant_response =
self.client.beta.threads.messages.list(
                thread_id=thread
            )
            return
assistant_response.data[0].content[0].text.value
```

Благодаря методу ai_msg_response происходит проверка наличия текста в сообщении пользователя, после чего, если сообщение есть – получаем ответ от модели GPT на текст данного сообщения, после чего ответ отправляется пользователю.

```
    async def ai_msg_response(message: types.Message):
        """метод для обработки текста исходя из
         сообщения со вставкой промта и датасета"""
        if message.text:
            response = await
MyOpenAI().text_answer_gpt(message.text)
            await message.answer(response)
```

Файл ui.py, в котором реализованы все кнопки и клавиатура в целом, которые появляются при выполнении команды /start. Кнопка создается с помощью types.ReplyKeyboardMarkup. Для каждой кнопки мы выдаем собственное название, для дальнейшей расстановки на раскладке. Чтобы объединить несколько кнопок в один ряд, используется функция markup.row. Помимо имени, каждой клавише мы задаем значение в виде текста,

необходимое в дальнейшей для выполнения следующих шагов. Кроме кнопок в данном файле мы задаем приветственное сообщение от бота.

```
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Построить маршрут')
    btn2 = types.KeyboardButton('Узнать погоду')
    btn3 = types.KeyboardButton('Конвертировать валюту')
    markup.row(btn1)
    markup.row(btn2, btn3)
    bot.send_message(message.chat.id, 'Привет! Я ваш личный ассистент по подбору идеального туристического маршрута!', reply_markup=markup)
```

При клике на кнопку для уточнения погоды, запускается функция bot.message_handler, которая подключает API сервиса OpenWeaherMap и предлагает указать указать город. Данный сервис был выбран из– за широкой области покрытия, узнать погоду можно в любой точке мира, открытого API, а также обширного функционала. В случае, если город будет указан не верно, мы получим сообщение об ошибке, и возможность указать место повторно. В случае успеха, мы получаем информация о температуре и скорости ветра в режиме реального времени.

```
@bot.message_handler(func=lambda message: message.text == 'Узнать погоду')
def get_weather(message):
    msg = bot.send_message(message.chat.id, 'Укажите, пожалуйста, город')
    bot.register_next_step_handler(msg, process_weather_step)

def process_weather_step(message):
    city = message.text.strip().lower()
    res = requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API}&units=metric')
```

```
        if res.status_code == 200:
            data = json.loads(res.text)
            bot.reply_to(message,                    f'Температура:
{data["main"]["temp"]}°\nВетер: {data["wind"]["speed"]}м/с')
        else:
            bot.reply_to(message, f'Город указан неверно')
```

В случае выбора клавиши “Конвертировать валюту”, запускается функция bot.message_handler. И алгоритм, позволяющий перевести валюту из одной единицы в другую. Для удобства пользователя, но втором шаге с помощью встроенных кнопок были вынесены наиболее актуальные пары валяют, а также возможность задать собственную пару. Для снижения вероятности появления ошибок, было задано условия, чтобы сумма для перевода была больше 0.

```
    @bot.message_handler(func=lambda  message:  message.text  ==
'Конвертировать валюту')
    def get_money(message):
        bot.send_message(message.chat.id, 'Введите сумму')
        bot.register_next_step_handler(message,
process_summa_step)

    def process_summa_step(message):
        global amount
        try:
            amount = int(message.text.strip())
        except ValueError:
            bot.send_message(message.chat.id,  'Неферный  формат.
Укажите, пожалуйста, сумму')
            bot.register_next_step_handler(message,
process_summa_step)
            return
        if amount > 0:
            markup =types.InlineKeyboardMarkup(row_width=2)
            btn_money1  =  types.InlineKeyboardButton('USD/EUR',
callback_data='usd/eur')
```

```
            btn_money2  =  types.InlineKeyboardButton('EUR/USD',
callback_data='eur/usd')
            btn_money3  =  types.InlineKeyboardButton('USD/GBP',
callback_data='usd/gbp')
            btn_money4   =   types.InlineKeyboardButton('Другое
значение', callback_data='else')
            markup.add(btn_money1,    btn_money2,    btn_money3,
btn_money4)
            bot.send_message(message.chat.id,   'Выберите   пару
валют', reply_markup=markup)
        else:
            bot.send_message(message.chat.id, 'Число должно быть
больше 0. Укажите, пожалуйста, сумму')
            bot.register_next_step_handler(message,
process_summa_step)

    @bot.callback_query_handler(func=lambda call: True)
    def callback(call):
        if call.data != 'else':
            values = call.data.upper().split('/')
            res = currency.convert(amount, values[0], values[1])
            bot.send_message(call.message.chat.id, f'Получается:
{round(res,2)}.')
        else:
            bot.send_message(call.message.chat.id, 'Введите пару
значений через /')
            bot.register_next_step_handler(call.message,
process_mycurrency_step)

    def process_mycurrency_step(message):
        try:
            values = message.text.upper().split('/')
            res = currency.convert(amount, values[0], values[1])
            bot.send_message(message.chat.id,      f'Получается:
{round(res, 2)}.')
        except Exception:
```

```
            bot.send_message(message.chat.id, f'Что- то пошло не
так, впишите значение заново.')
            bot.register_next_step_handler(message,
process_mycurrency_step)
            return
```

ЗАКЛЮЧЕНИЕ

На основании проведенного исследования и анализа представленного материала, можно сделать несколько ключевых выводов.

Во-первых, выбор платформы Telegram для реализации чат− бота является обоснованным благодаря его современному API, широким возможностям для пользователей и разработчиков, а также высокой популярности и широкому охвату аудитории. Telegram предлагает инструментарий, позволяющий создавать сложные и функциональные боты, которые удовлетворяют потребности пользователей.

Во-вторых, использование ChatGPT4 в качестве ИИ-ассистента для чат-бота - это оптимальное решение. Эта модель ИИ обладает способностью поддерживать длинные и связные разговоры, адаптироваться к контексту и предоставлять персонализированные ответы. Кроме того, ChatGPT4 можно настроить под специфические потребности проекта, что делает его универсальным для различных приложений.

В-третьих, выбор Python в качестве языка программирования для разработки чат-бота оправдан ввиду его простоты, читаемости кода, доступности многочисленных библиотек и фреймворков, а также кроссплатформенности. Python обеспечивает эффективную интеграцию с различными API и позволяет сократить время разработки благодаря своей лаконичности и поддержке широкого сообщества.

Таким образом, предложенная система, основанная на чат-боте в Telegram и использующая ChatGPT4 и метод колаборативной фильтрации, представляет собой оптимальное решение для реализации рекомендательного сервиса в сфере туризма. Этот подход позволит улучшить качество и эффективность продукта, сделать его более удобным и полезным для пользователей, а также обеспечить высокую адаптивность и персонализацию предоставляемых рекомендаций.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Jannach D. et al. Recommender Systems: An Introduction. – Cambridge University Press, 2010, p. 360.

2. Chen Y., Zhao X., de Rijke M. Top− N recommendation with highdimensional side information via locality preserving projection //Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2017. – С. 985− 988.

3. Николенко С. Рекомендательные системы //СПб: Изд− во Центр Речевых Технологий. – 2012 – 53 с.

4. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state− of− the− art and possible extensions //IEEE transactions on knowledge and data engineering. – 2005. – Т. 17. – №. 6. – С. 734− 749.

5. Gerbrands J. J. On the relationships between SVD, KLT and PCA //Pattern recognition. – 1981. – Т. 14. – №. 1− 6. – С. 375− 381.

6. Hidasi B. et al. Session− based recommendations with recurrent neural networks //arXiv preprint arXiv:1511.06939. – 2015.

7. Bottou L. Stochastic gradient descent tricks //Neural networks: Tricks of the trade. – Springer, Berlin, Heidelberg, 2012. – С. 421− 436.

8. Звягин Л. С. Метод байесовских сетей и ключевые аспекты байесовского моделирования //Международная конференция по мягким вычислениям и измерениям. – Федеральное государственное автономное образовательное учреждение высшего образования Санкт− Петербургский государственный электротехнический университет ЛЭТИ им. ВИ Ульянова (Ленина), 2019. – С. 30− 34

9. Коцюба И.Ю., Назаренко А.Е. Разработка рекомендательной системы для планирования туристических маршрутов в оптимизационной постановке Моделирование, оптимизация и информационные технологии. 2020;8(2).

10. Михайлов С.А. Интеллектуальная система помощи туристу: сервис− ориентированная архитектура и реализация // Научно− технический вестник информационных технологий, механики и оптики. 2019. Т. 19. № 3. С. 499− 507.

11. Шилов Н.Г. Методология построения проактивных рекомендующих систем для инфомобильных приложений. 2016.

ПРИЛОЖЕНИЕ А

(обязательное)

Рекомендательная система синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений

Листинг программы

config.py

```
# coding utf- 8
import os
class Settings(object):
    """Настройки для бота."""
    token = '7211174904:AAECPBsvIGrPp0yA1eSgX- waBoONcp0zoVU''
    def __str__(self):
        return 'Settings object'
    import enum
    import time
    from aiogram import Dispatcher, types
    from openai import OpenAI
    from config import Settings
    from currency_converter import CurrencyConverter

    token  = '7211174904:AAECPBsvIGrPp0yA1eSgX- waBoONcp0zoVU'
    API = 'b82323a813707b7593ff50b5f17e7bd1'
```

aihandlers.py

```
import enum
import pandas as pd
from aiogram import Dispatcher, types
from openai import OpenAI
from config import Settings
class Prompts(enum.Enum):
    """Промты"""
    MSG_PROMPT = ("Отвечай только на вопросы, связанные с
планированием туристических маршрутов и туризмом"
                  "Предоставляй ответы максимально развернуто и
без воды."
                  " Используй для ответа файл file-
YRxr3nAMkMJ8fPIGCL43qEPO."
                  )
class MyOpenAI:
    """Класс для взаимодействия с OpenAI"""
    def __init__(self):
```

```
        self.client = OpenAI(api_key=Settings.OPENAI_TOKEN)
        self.csv_file =
"/home/PycharmProjects/projects/touristData.csv"

    async def text_answer_gpt(self, message: str):
        """Ответ текстом от ГПТ"""
        completion = self.client.chat.completions.create(
            model="gpt- 4o",
            messages=[
                {"role": "system", "content":
Prompts.MSG_PROMPT.value},
                {"role": "user", "content": message}
            ]
        )
        return completion.choices[0].message.content

    def upload_file(self) - > str:
        """Загрузка файлов в OPenAI"""
        with open(self.csv_file, "rb") as binary_file:
            upload_file = self.client.files.create(
                file=binary_file,
                purpose="assistants"
            )
            return upload_file.id


async def ai_msg_response(message: types.Message):
    """метод для обработки текста исходя из
     сообщения со вставкой промта и датасета"""
    if message.text:
        response = await
MyOpenAI().text_answer_gpt(message.text)
        await message.answer(response)
def ai_handlers(dispatcher: Dispatcher):
    dispatcher.register_message_handler(
        callback=ai_msg_response
```

```
    )
```

rating.py

```
from typing import List
import orjson
import pandas as pd
from pydantic import BaseModel, validator
from app.models.schemas.rwschema import RWSchema
from enum import Enum
    else:
     raise TypeError
    def dump_rating_type_as_int(v, *, default):
     for rating_data in v:
     rating_data.update({'rating':
    rating_data['rating'].val_as_int})
     return orjson.dumps(v, default=default).decode()
    class RatingDataInSave(RWSchema):
     uid: str
     iid: str
     rating: RatingType
    class ListRatingDataInSave(BaseModel):
     __root__: List[RatingDataInSave]
     @validator('__root__', pre=True)
     def uniq_data(cls, v):
     df = pd.DataFrame(v)
     df_duplicated = df[df.duplicated(['uid', 'iid'])]
     if not df_duplicated.empty:
     raise ValueError('Together uid and iid must be unique')
     return v
     class Config:
     json_dumps = dump_rating_type_as_int
```

users.py

```
from typing import Optional
from pydantic import BaseModel, EmailStr, HttpUrl
```

```
from app.models.domain.users import User
from app.models.schemas.rwschema import RWSchema
    class UserInLogin(RWSchema):
     email: EmailStr
     password: str
    class UserInCreate(UserInLogin):
     username: str
    class UserInUpdate(BaseModel):
     username: Optional[str] = None
     email: Optional[EmailStr] = None
     password: Optional[str] = None
     bio: Optional[str] = None
     image: Optional[HttpUrl] = None
    class UserWithToken(User):
     token: str
    class UserInResponse(RWSchema):
     user: UserWithToken
```

rwschema.py

```
from app.models.domain.rwmodel import RWModel
class RWSchema(RWModel):
 class Config(RWModel.Config):
 orm_mode = True
```

jwt_meta.py

```
from datetime import datetime
from pydantic import BaseModel
    class JWTMeta(BaseModel):
     exp: datetime
     sub: str
    class JWTUser(BaseModel):
     username: str
```

jwt.py

```
from datetime import datetime, timedelta
from typing import Dict
import jwt
```

```
from pydantic import ValidationError
from app.models.domain.users import User
from app.models.schemas.jwt import JWTMeta, JWTUser
    JWT_SUBJECT = "access"
    ALGORITHM = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES = 60 * 24 * 7 # one week
    def create_jwt_token(
     *, jwt_content: Dict[str, str], secret_key: str,
    expires_delta: timedelta,
    ) - > str:
     to_encode = jwt_content.copy()
     expire = datetime.utcnow() + expires_delta
     to_encode.update(JWTMeta(exp=expire,
    sub=JWT_SUBJECT).dict())
     return jwt.encode(to_encode, secret_key,
    algorithm=ALGORITHM).decode()
    def create_access_token_for_user(user: User, secret_key: str)
    - > str:
     return create_jwt_token(
     jwt_content=JWTUser(username=user.username).dict(),
     secret_key=secret_key,

    expires_delta=timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES),
     )
    def get_username_from_token(token: str, secret_key: str) - >
    str:
     try:
     return JWTUser(**jwt.decode(token, secret_key,
    algorithms=[ALGORITHM])).username
     except jwt.PyJWTError as decode_error:
     raise ValueError("unable to decode JWT token") from
    decode_error
     except ValidationError as validation_error:
     raise ValueError("malformed payload in token") from
    validation_error
```

cms.py

```
from collections import ChainMap
from typing import List
import orjson
from aioredis import Redis
    from httpx import AsyncClient
    async def get_cms_entities(_redis: Redis, ids: List[str],
    language: str, paging: Pageable):
     resp = {
     'page': paging.page,
     'pageSize': paging.size,
     'total': len(ids)
     }
     keys = ["{0}_{1}".format(_id, language.lower()) for _id
    in ids]
     if not keys:
     resp['items'] = []
     return resp
    98
     data = [orjson.loads(c) for c in await
    _redis.mget(*keys) if c is not None]
     data_keys = list(ChainMap(*data))
     for key in data_keys:
     unzip = {item['id']: item for sublist in
 [item_list[key] for item_list in data if key in item_list
and isinstance(item_list[key],
list)]
 for item in sublist if 'id' in item}
 resp[key] = list(unzip.values())
 return resp
```

sync_cms_tour.py

```
import re
from concurrent.futures.thread import ThreadPoolExecutor
from typing import List
from urllib.parse import urljoin
```

```
import requests
from loguru import logger
from redis import Redis
from sqlalchemy import create_engine
from sqlalchemy.sql import text
from app.core.config import DATABASE_URL, CMS_DETAILS_URL,
CMS_HEADERS,
CMS_HOST, CMS_DEFAULT_LANGUAGE
from app.models.domain.cms import EntityType
from app.utils import redis_utils, sql_utils
    CMS_TOUR_URL = urljoin(CMS_HOST, 'v1/screens/tours')
    PAGE_SIZE = 100
    REQUEST_THREAD_NUMBER = 20
    TAG_RE = re.compile(r'<[^>]+>')
    99
    def _remove_tags(_text: str) - > str:
     return TAG_RE.sub('', _text)
    def sync_tours() - > None:
     page = 1
     total = 0
     cms_languages =
    sql_utils.get_cms_languages(DATABASE_URL.url)
     logger.info("Start sync tours...")
     engine = create_engine(DATABASE_URL.url)
     with engine.connect() as con, redis_utils.connect() as
    r_con:
     while True:
     payload = {'language': CMS_DEFAULT_LANGUAGE, 'page': page,
    'pageSize':
    PAGE_SIZE}
     r = requests.get(CMS_TOUR_URL, params=payload,
    headers=CMS_HEADERS)
     data = r.json()
     if data['pageSize'] > 0 and 'items' in data:
     args = [(con, r_con, cms_languages, item['id'], ) for item
    in data['items']]
```

```
    with ThreadPoolExecutor(REQUEST_THREAD_NUMBER) as executor:
    executor.map(lambda p: _processing_tour(*p), args)
    total += len(data['items'])
    if total >= data['total']:
    break
    logger.info("End sync tours...")
```

main.py

```
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Построить маршрут')
    btn2 = types.KeyboardButton('Узнать погоду')
    btn3 = types.KeyboardButton('Конвертировать валюту')
    markup.row(btn1)
    markup.row(btn2, btn3)
    bot.send_message(message.chat.id, 'Привет! Я ваш личный
ассистент по подбору идеального туристического маршрута!',
reply_markup=markup)

@bot.message_handler(func=lambda message: message.text ==
'Узнать погоду')
def get_weather(message):
    msg = bot.send_message(message.chat.id, 'Укажите,
пожалуйста, город')
    bot.register_next_step_handler(msg, process_weather_step)

def process_weather_step(message):
    city = message.text.strip().lower()
    res =
requests.get(f'https://api.openweathermap.org/data/2.5/weather?q
={city}&appid={API}&units=metric')
    if res.status_code == 200:
        data = json.loads(res.text)
        bot.reply_to(message, f'Температура:
{data["main"]["temp"]}°\nВетер: {data["wind"]["speed"]}м/с')
```

```
    else:
        bot.reply_to(message, f'Город указан неверно')
@bot.message_handler(func=lambda message: message.text ==
'Конвертировать валюту')
def get_money(message):
    bot.send_message(message.chat.id, 'Введите сумму')
    bot.register_next_step_handler(message, process_summa_step)
def process_summa_step(message):
    global amount
    try:
        amount = int(message.text.strip())
    except ValueError:
        bot.send_message(message.chat.id, 'Неферный формат.
Укажите, пожалуйста, сумму')
        bot.register_next_step_handler(message,
process_summa_step)
        return
    if amount > 0:
        markup =types.InlineKeyboardMarkup(row_width=2)
        btn_money1 = types.InlineKeyboardButton('USD/EUR',
callback_data='usd/eur')
        btn_money2 = types.InlineKeyboardButton('EUR/USD',
callback_data='eur/usd')
        btn_money3 = types.InlineKeyboardButton('USD/GBP',
callback_data='usd/gbp')
        btn_money4 = types.InlineKeyboardButton('Другое
значение', callback_data='else')
        markup.add(btn_money1, btn_money2, btn_money3,
btn_money4)
        bot.send_message(message.chat.id, 'Выберите пару валют',
reply_markup=markup)
    else:
        bot.send_message(message.chat.id, 'Число должно быть
больше 0. Укажите, пожалуйста, сумму')
        bot.register_next_step_handler(message,
process_summa_step)
```

```
@bot.callback_query_handler(func=lambda call: True)
def callback(call):
    if call.data != 'else':
        values = call.data.upper().split('/')
        res = currency.convert(amount, values[0], values[1])
        bot.send_message(call.message.chat.id, f'Получается:
{round(res,2)}.')
    else:
        bot.send_message(call.message.chat.id, 'Введите пару
значений через /')
        bot.register_next_step_handler(call.message,
process_mycurrency_step)
def process_mycurrency_step(message):
    try:
        values = message.text.upper().split('/')
        res = currency.convert(amount, values[0], values[1])
        bot.send_message(message.chat.id, f'Получается:
{round(res, 2)}.')
    except Exception:
        bot.send_message(message.chat.id, f'Что- то пошло не
так, впишите значение заново.')
        bot.register_next_step_handler(message,
process_mycurrency_step)
        return
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Рекомендательная система синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений

Презентация

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА СИНТЕЗА ОПТИМАЛЬНЫХ МНОГОЦЕЛЕВЫХ МАРШРУТОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ТУРИСТИЧЕСКИХ ПРЕДПОЧТЕНИЙ

ВЫПОЛНИЛ: ПРОКОФЬЕВ Н.А.
РУКОВОДИТЕЛЬ: ФИНОГЕЕВ А.Г.

Задачи и цели работы

Цель работы – необходимо разработать рекомендательную систему синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений. Для достижения цели необходимо выполнение следующих задач:

- Анализ предметной области,
- Обзор аналогов разрабатываемой подсистемы,
- Выбор платформы
- Выбор языка программирования.
- Проектирование подсистемы
- Разработка подсистемы

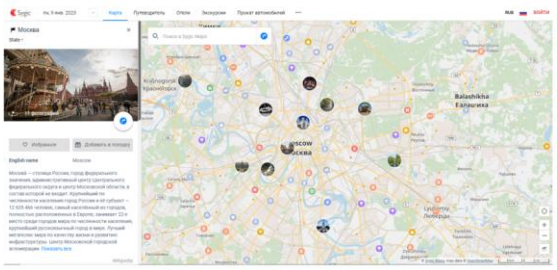
Актуальность работы

Важность таких систем нельзя недооценивать: они предоставляют возможность не только экономить время при планировании поездок, но и обеспечивают более глубокое взаимодействие между туристами и местами, которые они посещают, делая путешествие более полезным и гармоничным.

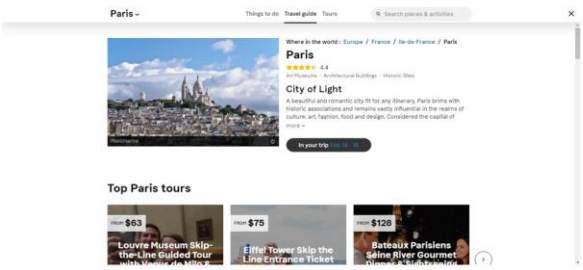
Полученная система позволяет решить проблему информационного пузыря, а дополнительный функционал и искусственный интеллект позволяет упростить планирование путешествия.

Поэтому разработка рекомендательной системы синтеза оптимальных многоцелевых маршрутов на базе интеллектуального анализа туристических предпочтений.

Обзор аналогов

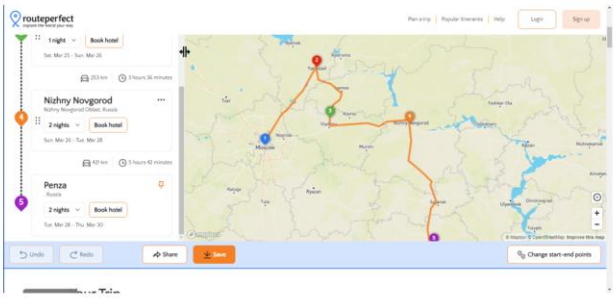


Интерфейс системы Tripomatic



Интерфейс системы Inspirock

Обзор аналогов



Интерфейс системы Routeperfect

5

Достоинства и недостатки аналогов

Основные достоинства рассмотренных аналогов:

- Интуитивно понятный интерфейс
- Поддержка несколько языков

Недостатки аналогов:

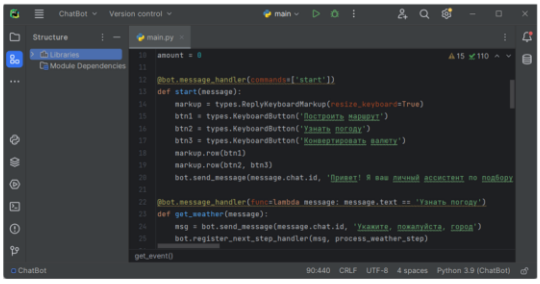
- Отсутствие мобильного приложения
- Слабая адаптация для использования на территории России
- Ограниченный функционал

6

Выбор средств разработки

В качестве языка разработки был выбран Python – это один из самых популярных языков программирования, который идеально подходит для разработки чат-ботов для Телеграма по причине простоты и чистоты код

В качестве среды разработки был выбран PyCharm – это оптимальная среда разработки (IDE) для Python.



PyCharm

7

Выбор средств разработки

Основной причиной выбора Telegram в качестве платформы для создания программыы является его современная и хорошо документированная API для разработки ботов. Также, по сравнению с другими мессенджерами, Telegram предлагает больше возможностей как для пользователей, так и для разработчиков, что делает его интерфейс привлекательным для создания и использования чат-ботов.

Telegram чат-бот

8

Интерфейс системы

Основное окно

На рисунке показан главный экран чат-бота, который видит пользователь при переходе. На экране отображено окно, в котором описываются возможность бота, поле для ввода текста, а так же кнопка вызова меню, через которую воспользоваться быстрыми командами, например /start, для начала работы с чат-ботом.

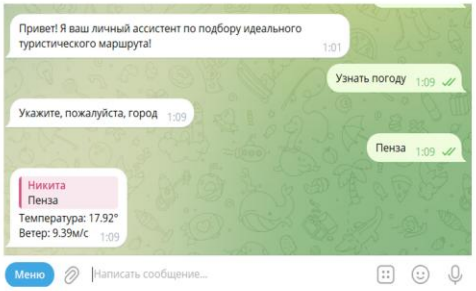
Интерфейс системы

Процесс построения маршрута

Вывод результата

Интерфейс системы

Алгоритм конвертации валют



Алгоритм получения информации о погоде

11

Заключение

Был произведён анализ литературы по исследуемой предметной области, а также осуществлен обзор аналогов проектируемой системы.

Разработана структура системы и ее компоненты, обеспечивающие работу рекомендательной системы и вывод результата подбора маршрута на экран для пользователя, с дальнейшим взаимодействием с ним.

В дальнейшем предполагается развитие системы для расширения её функционала, увеличения точности расчётов и выводимого результата, а также интеграция картографического сервиса, для наглядного построения маршрута.

12