МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра математической теории упругости и биомеханики

Разработка чат-бота для туристической навигации

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 442 группы

направления 09.03.03 - Прикладная информатика

механико-математического факультета

Сухарникова Владислава Владимировича

Научный руководитель
доцент, к.ю.н. Р.В. Амелин

Зав. кафедрой
зав. кафедрой, д.ф.-м.н., профессор Л.Ю. Коссович

Саратов 2026

Введение. Современная туристическая сфера активно использует цифровые инструменты для упрощения ориентирования на незнакомой местности. Растёт спрос на компактные и оперативные решения, которые позволяют путешественникам быстро получать готовые маршруты без необходимости устанавливать отдельные приложения или самостоятельно компоновать точки интереса. Мессенджеры, в частности Telegram, становятся удобной платформой для взаимодействия с навигационными сервисами благодаря мгновенному доступу, поддержке геолокационных данных и привычному для пользователя интерфейсу.

Существующие картографические и туристические приложения предлагают избыточный функционал, требующий ручного планирования маршрутов, фильтрации отзывов и длительного изучения справочной информации. Универсальные навигационные платформы, такие как Яндекс.Карты, требуют самостоятельного построения цепочек точек, специализированные туристические приложения — установки и регистрации, а статичные публикации в Telegram-каналах лишены интерактивности и привязки к геолокации. Это создаёт разрыв между получением информации и её практическим применением, увеличивая время от принятия решения до начала перемещения.

Целью данной дипломной работы является разработка функционального Telegram-бота для туристической навигации, который принимает геолокацию пользователя, предлагает выбрать категорию маршрута и формирует ссылку на Яндекс.Карты с уже построенным путём, включающим несколько ключевых объектов.

В соответствии с поставленной целью предполагается **решение следующих задач**:

1. Провести анализ предметной области и существующих навигационных решений, обосновать целесообразность использования лёгкого мессенджер-ориентированного подхода с заранее подготовленными тематическими маршрутами.
2. Сформулировать функциональные и нефункциональные требования к системе, подготовить техническое задание с учётом специфики работы в среде Telegram и интеграции с картографическим сервисом.

3. Разработать информационную модель данных, спроектировать диалоговый сценарий и архитектуру взаимодействия компонентов с использованием UML-диаграмм.
4. Выбрать технологический стек для реализации серверной части и интеграции с Telegram Bot API и Яндекс.Картами, определить логику формирования динамических навигационных ссылок.
5. Выполнить программную реализацию чат-бота, реализовать обработку геоданных, выбор категории маршрута и генерацию ссылок, провести тестирование работоспособности системы.

Структура работы. Выпускная квалификационная работа состоит из введения, трёх глав, заключения и списка использованных источников.

Первая глава посвящена анализу предметной области и постановке задачи: выполнен обзор существующих решений, выявлены их недостатки, обоснована необходимость разработки бота, сформулировано техническое задание.

Вторая глава содержит проектирование информационной модели: построена ER-диаграмма базы данных, разработаны UML-диаграммы, спроектирована архитектура системы.

Третья глава охватывает программную реализацию: выбор технологического стека, разработка модулей чат-бота, создание диалоговых сценариев и интерфейса, а также тестирование системы.

В первой главе «Анализ предметной области и постановка задачи» проведён комплексный анализ предметной области туристической навигации и обоснована целесообразность создания Telegram-бота для мгновенного предоставления тематических маршрутов. Существующие решения классифицированы на три категории: универсальные картографические платформы (Яндекс.Карты, Google Maps), специализированные туристические приложения с кураторским контентом (izi.TRAVEL) и мессенджер-ориентированные сервисы. Выявлены их системные недостатки: избыточная когнитивная нагрузка при ручном планировании, необходимость установки и регистрации, отсутствие интерактивности и привязки к геолокации в статичных публикациях. На рынке отсутствует лёгкий, контекстно-зависимый инструмент, объединя-

ющий мгновенную обработку геолокации, проверенные тематические подборки и прямую передачу координат в популярный картографический сервис.

На основе анализа трёх типовых профилей пользователей (самостоятельный турист, гость города с ограниченным временем, локальный исследователь) сформулированы ключевые пользовательские сценарии: получение маршрута по геолокации, использование маршрута по умолчанию (город Саратов), смена категории маршрута без повторной отправки геолокации, а также обработка ошибочных ситуаций. Проведён сравнительный анализ по семи критериям: порог входа, интерактивность, привязка к геолокации, поддержка тематических маршрутов, интеграция с картами, время до получения результата и ресурсоёмкость клиентской части. Проектируемый бот демонстрирует нулевой порог входа, высокую интерактивность, автоматическую привязку к геолокации, наличие четырёх тематических категорий (военная история, духовные памятники, культурные объекты, природные маршруты), прямую интеграцию с Яндекс.Картами, время до результата менее 30 секунд и минимальную ресурсоёмкость, так как всё взаимодействие происходит через интерфейс Telegram.

Идентифицированы технические риски (нестабильность Telegram Bot API, изменение формата ссылок Яндекс.Карт), операционные риски (актуальность данных о точках интереса) и внешние риски (зависимость от сторонних сервисов). Предложены меры снижения: использование библиотеки aiogram с обработкой сетевых ошибок, выделение логики генерации ссылок в отдельный модуль, кураторский подход к формированию маршрутов с периодической ручной верификацией, реализация механизмов graceful degradation. Обоснована необходимость разработки: современные инструменты не решают задачу ориентирования в режиме «здесь и сейчас» без избыточной когнитивной нагрузки. Происходит смещение фокуса от вопроса «где искать и как строить?» к оперативному ответу «как быстро попасть в проверенные места отсюда?».

На основе проведённого анализа сформулировано техническое задание, которое определяет функциональные требования (приём и валидация геолокации, предоставление выбора из четырёх категорий через inline-клавиатуру, извлечение трёх точек интереса из локальной базы данных SQLite, динами-

ческая генерация гиперссылки на Яндекс.Карты), нефункциональные требования (время отклика не более 2 секунд при стабильном соединении, отказоустойчивость, конфиденциальность — обработка геоданных только в оперативной памяти), требования к программной среде (Python 3.8+, асинхронный фреймворк aiogram v3, SQLite) и эргономические требования (количество шагов от передачи геолокации до получения ссылки — три действия).

Во второй главе «Проектирование бота» выполнено проектирование информационной модели и архитектуры, разработаны UML-диаграммы и детально описана логика работы конечного автомата состояний.

Проектирование базы данных основывалось на требовании простоты развёртывания и возможности динамического обновления контента без модификации исходного кода. Была разработана единая плоская таблица pois, содержащая все необходимые справочные данные. Ключевая сущность — точка интереса (POI) — включает следующие атрибуты: уникальный идентификатор, категория (military, spiritual, culture, nature), название объекта, географические координаты в формате десятичных градусов (широта и долгота), краткое описание. Выбор плоской таблицы обусловлен архитектурным решением о контролируемой денормализации на этапе минимально жизнеспособного продукта (MVP). Поскольку набор категорий фиксирован и не требует сложной иерархии, хранение идентификатора категории непосредственно в записи точки интереса устраняет необходимость в операциях JOIN, что значительно ускоряет выполнение запросов. Индексация по полю category обеспечивает эффективную выборку за время O(log n). Для каждой из четырёх категорий заданы три объекта с координатами и краткими описаниями (например, для военной категории: Парк Победы на Соколовой горе, памятник «Журавли», Музей боевой славы). На основе описанной модели построена ER-диаграмма, наглядно демонстрирующая схему хранения данных.

Для документирования структуры и поведения системы применён унифицированный язык моделирования UML. Разработаны диаграмма вариантов использования, диаграмма состояний, диаграмма компонентов, диаграмма классов и диаграмма последовательности.

Диаграмма вариантов использования описывает взаимодействие пользователя (туриста) с платформой Telegram. Основными прецедентами выступа-

ют: инициирование диалога, передача геолокационных данных, выбор категории маршрута, получение навигационной ссылки, обработка ошибок ввода, отмена сценария и повторный запуск. Каждый прецедент сопровождается чётко определёнными предусловиями и альтернативными ветками выполнения.

Центральным элементом поведенческой модели является конечный автомат (FSM), управляющий жизненным циклом диалоговой сессии. В реализации на базе фреймворка aiogram состояния системы детерминированы и включают: IDLE (начальное состояние, ожидание команды /start), AWAITING_LOCATION (обработка входящей геолокации), AWAITING_CATEGORY (отображение inline-клавиатуры с четырьмя категориями, ожидание выбора), GENERATING_ROUTE (извлечение данных из базы, формирование URL, подготовка ответа), ERROR (обработка некорректных запросов и системных сбоев). Диаграмма состояний визуализирует все допустимые переходы между состояниями. FSM гарантирует сохранение контекста при асинхронных задержках сети, предотвращает выполнение невалидных последовательностей действий и обеспечивает корректную очистку памяти по завершении диалога.

Компонентная архитектура системы выделяет следующие логические блоки: API Gateway (bot.py, router.py) — модуль взаимодействия с Telegram Bot API, отвечающий за получение обновлений через start_polling, диспетчеризацию событий и отправку сообщений; Session Manager (aiogram.fsm.context) — реализация FSM, хранение временного состояния сессии; Database Layer (database.py) — модуль работы с SQLite, обеспечивающий инициализацию таблицы pois, выполнение SQL-запросов для извлечения точек интереса по категории; Data Access Layer (api_client.py) — абстракция работы с хранилищем, преобразование результатов в типизированные структуры; Link Builder (api_client.py) — формирование стандартизированного URL для Яндекс.Карт с параметрами промежуточных точек (rtext) и режимом построения (rtt=auto); Error Handler — централизованная обработка исключений с логированием и отправкой пользовательских уведомлений. Диаграмма последовательности детализирует обмен сообщениями между участниками: от получения команды /start и отправки геолокации до

выбора категории, извлечения данных из БД, генерации ссылки и отправки структурированного ответа.

Архитектура системы реализована по событийно-ориентированной клиент-серверной модели, адаптированной под среду мессенджеров. Клиентская часть отсутствует в классическом понимании — её роль выполняет инфраструктура Telegram, обеспечивающая интерфейс ввода-вывода, аутентификацию и доставку сообщений. Серверная часть представляет собой легковесное асинхронное приложение на Python с использованием aiogram, обрабатывающее входящие события через механизм Long Polling. Интеграция с Яндекс.Картами выполнена по принципу «тонкого клиента»: бот не выполняет геокодирование и прокладку маршрутов, а преобразует внутренние идентификаторы и координаты в готовую URL-строку. Метод build_route_link агрегирует стартовую точку (геолокацию пользователя или координаты центра Саратова) и координаты трёх POI в параметр rtext формата «lat1,lon1-lat2,lon2-. . . », дополняя его режимом rtt=auto. При клике по ссылке пользователь перенаправляется в веб-интерфейс или мобильное приложение Яндекс.Карт, где вся визуализация и динамический расчёт пути выполняются на стороне профильного сервиса. Конфиденциальность обеспечена вынесением токена бота в файл окружения .env и обработкой геолокации исключительно в оперативной памяти в рамках текущей сессии без записи в журналы или внешние хранилища. Асинхронный цикл asyncio позволяет обрабатывать сотни одновременных сессий без блокировок, а ресурсоёмкость серверного процесса не превышает 40–60 МБ ОЗУ.

В третьей главе «Реализация» подробно описывается программная реализация разработанного Telegram-бота. Базовым языком программирования выступил Python версии 3.10+. Серверная часть построена на асинхронном фреймворке aiogram v3, который обеспечивает обработку входящих обновлений через метод start_polling, поддержку конечных автоматов состояний (FSM) и удобные инструменты формирования интерактивных клавиатур. Управление зависимостями осуществлялось через менеджер пакетов pip, контроль версий — через систему Git.

Архитектура приложения спроектирована по принципу модульности: исходный код разделён на отдельные файлы, каждый из которых решает

свою задачу. Файл bot.py является точкой входа: он инициализирует бота и диспетчер, настраивает систему логирования, вызывает инициализацию базы данных и запускает механизм Long Polling. Модуль config.py с использованием библиотеки python-dotenv загружает токен бота из файла окружения .env, что исключает попадание чувствительных данных в систему контроля версий. Модуль database.py инкапсулирует все операции с SQLite: создание таблицы pois, начальное заполнение данными (12 записей — три объекта в каждой из четырёх категорий), а также функцию get_pois_by_category, выполняющую SQL-запрос с фильтрацией по категории и ограничением в три записи. Центральный модуль router.py содержит обработчики команд и callback-запросов. Реализован класс TourStates с состоянием awaiting_category. Обработчик команды /start очищает состояние и отправляет reply-клавиатуру с двумя кнопками: «Отправить геолокацию» (с параметром request_location=True) и «Использовать Саратов по умолчанию» (координаты 51.5406, 46.0086). При получении объекта location координаты извлекаются и сохраняются в контексте FSM. Функция show_category_keyboard формирует inline-клавиатуру с четырьмя кнопками, каждая из которых имеет callback_data, соответствующий категории. Обработчик process_category при нажатии кнопки извлекает координаты из состояния, вызывает метод get_pois модуля api_client, формирует ссылку через build_route_link и отправляет пользователю структурированное сообщение с перечнем точек, краткими описаниями и активной гиперссылкой. Предусмотрена кнопка «Выбрать другую категорию», которая возвращает пользователя к началу сценария. Модуль api_client.py реализует класс MapsClient. Метод get_pois асинхронно вызывает функцию get_pois_by_category через asyncio.to_thread, чтобы не блокировать основной цикл событий, и преобразует полученные записи в список типизированных объектов POI (dataclass). Метод build_route_link агрегирует стартовые координаты и координаты трёх POI в формат rtext (например, «51.5406,46.0086-51.5856,46.0352-51.5321,46.0125-51.5289,46.0098») и возвращает URL вида `https://yandex.ru/maps/194/saratov/?rtext=...&rtt=auto`.

Клиентская часть бота полностью реализована средствами Telegram: reply-клавиатура для запроса геолокации, inline-клавиатура для выбора ка-

тегории, форматированные сообщения с HTML-разметкой (полужирные заголовки, активные ссылки с отключённым предпросмотром). Пользовательский сценарий линеен и минималистичен: команда /start → нажатие кнопки геолокации (или выбор города по умолчанию) → выбор категории → получение готового маршрута. Все взаимодействия асинхронны, интерфейс адаптирован для использования в движении. В работе приведены скриншоты этапов: экран инициализации с запросом геолокации, интерфейс выбора тематической категории, сообщение со сформированным маршрутом и ссылкой на Яндекс.Карты, а также итоговый маршрут в картографическом сервисе.

Тестирование и оценка эффективности проводились по трём направлениям: проверка модуля работы с базой данных, функциональная проверка серверной логики и тестирование пользовательских сценариев. В ходе тестирования модуля базы данных подтверждена корректность инициализации таблицы pois, выполнение SQL-запросов с фильтрацией по категории (для каждой из четырёх категорий система возвращает ровно три точки интереса с корректными координатами и описаниями), а также обработка отсутствующих данных (при передаче несуществующей категории запрос возвращает пустой результат). Модуль приёма геолокации успешно прошёл проверку: система корректно извлекает широту и долготу из объекта location, валидирует их присутствие и переходит к этапу выбора категории. Модуль формирования ссылок протестирован на различных наборах координат, включая граничные значения; при пустом результате система возвращает ссылку на общую карту города. Среднее время отклика на формирование маршрута составило 0.15–0.3 секунды при стабильном сетевом соединении. Этап извлечения данных из SQLite выполняется с использованием индекса по полю category за время O(log n), формирование ссылки представляет собой простую строковую операцию без сетевых запросов к внешним API. Основная задержка обусловлена исключительно временем доставки сообщений через инфраструктуру Telegram. В ходе комплексного тестирования подтверждена стабильная работа всех компонентов, корректная обработка исключительных ситуаций (отсутствие геолокации, отмена выбора, дублирование запросов) и соответствие системы техническому заданию.

Заключение. В ходе выполнения выпускной квалификационной работы спроектирован и программно реализован Telegram-бот для туристической навигации, обеспечивающий мгновенное преобразование геолокации и выбора категории в готовый маршрут на Яндекс.Картах. Проведённый анализ предметной области выявил системные ограничения существующих решений: избыточную когнитивную нагрузку универсальных карт, необходимость установки и регистрации в специализированных приложениях, отсутствие интерактивности и привязки к геолокации в статичных мессенджер-подборках. На основе выявленных проблем сформулирована цель работы, разработано техническое задание и спроектирована архитектура системы.

Все задачи, поставленные во введении, решены в полном объёме. Информационная модель реализована в виде плоской таблицы pois в СУБД SQLite, содержащей четыре тематические категории по три точки интереса в каждой с географическими координатами и описаниями. Поведение системы смоделировано с использованием конечного автомата (FSM), что гарантирует сохранение контекста диалога при асинхронных задержках сети и предотвращает выполнение невалидных последовательностей действий. Пользовательский сценарий свёрнут к трём шагам: передача геолокации, выбор категории, получение навигационной ссылки. Интерфейс реализован исключительно средствами Telegram (inline-клавиатуры, кнопки шаринга локации, форматированные сообщения), что обеспечивает кроссплатформенную доступность и нулевой порог входа.

Программная реализация выполнена на языке Python с использованием асинхронного фреймворка aiogram v3. Интеграция с картографическим сервисом осуществлена через генерацию стандартизированных URL-параметров для Яндекс.Карт, что позволило переложить задачу визуализации и прокладки маршрута на профильный внешний сервис без использования тяжёлых SDK. Конфигурационные параметры вынесены в переменные окружения, что соответствует принципам безопасности и упрощает развёртывание. Комплексное тестирование подтвердило корректность работы всех функциональных модулей, среднее время отклика на формирование маршрута составляет 0.15–0.3 секунды.

Практическая значимость работы заключается в создании готового к эксплуатации прототипа, который может быть использован как самостоятельный навигационный сервис для туристов в городе Саратов, а также как масштабируемая основа для расширения на другие регионы. Архитектура системы допускает добавление новых категорий, городов и точек интереса через SQL-запросы без рефакторинга ядра. Для дальнейшего развития возможно расширить функционал за счёт интеграции с сервисами отзывов и расписаний, реализовать мультиязычный интерфейс для иностранных туристов, а также внедрить аналитику использования для оценки популярности категорий и оптимизации кураторского контента.

Разработанная система демонстрирует высокую степень готовности к промышленной эксплуатации, соответствует современным требованиям к лёгким диалоговым ассистентам и может служить основой для создания масштабируемых навигационных сервисов нового поколения. Таким образом, цели и задачи выпускной квалификационной работы полностью достигнуты.