УДК 004.415.2 https://doi.org/10.54251/2616-6429.2025.04.n11

Т.Т. Оспанова*, Р.Н. Сериков
к.т.н., доцент, ЕНУ им. Л.Н. Гумилёва, Астана, Казахстан
магистрант, ЕНУ им. Л.Н. Гумилёва, Астана, Казахстан
***Автор для корреспонденции:** tleu2009@mail.ru

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ГОРОДСКОГО ТАКСИ НА ОСНОВЕ FASTAPI

Аннотация

В статье рассматривается разработка информационной системы городского такси и результаты экспериментальной проверки её основных функциональных возможностей. Система реализована с использованием FastAPI и PostgreSQL, что позволило автоматизировать ключевые процессы: регистрацию пользователей, создание заказов, назначение водителей и обработку геолокационных данных в реальном времени. Для определения ближайшего водителя применяется метод вычисления геодезического расстояния, обеспечивающий достаточную точность при городских перемещениях. Механизм управления статусами водителей продемонстрировал устойчивость и корректность переходов между состояниями в ходе экспериментов. Полученные результаты подтвердили эффективность предложенной архитектуры и показали её потенциал для дальнейшего развития, включая расширение функционала, интеграцию аналитических модулей и использование методов оптимизации маршрутов. Представленная система может служить основой для построения полнофункциональных сервисов городского такси.

Ключевые слова: информационная система, городское такси, FastAPI, геолокация, PostgreSQL, оптимизация распределения, цифровые сервисы.

Введение

В условиях цифровизации городских транспортных систем возрастает потребность в программных платформах, обеспечивающих оперативное назначение водителей, снижение времени ожидания пассажиров и оптимизацию логистических процессов. Сервисы такси являются одним из наиболее востребованных видов интеллектуальных транспортных систем, в которых ключевую роль играет корректная обработка геолокационных данных и эффективное распределение транспортных ресурсов.

Современные исследования показывают, что задачи выбора оптимального водителя, формирования маршрутов и минимизации временных затрат относятся к классу NP-трудных проблем, требующих разработки приближённых алгоритмов и адаптивных архитектур. Несмотря на значительное внимание к оптимизационным методам, практическая реализация прикладных информационных систем, интегрирующих эти методы в реальную инфраструктуру, освещена ограниченно.

Целью данной работы является разработка академически обоснованного прототипа информационной системы городского такси, включающего архитектуру, алгоритмы, экспериментальную проверку и обзор применяемых исследовательских подходов.

Материалы и методы

Теоретический анализ

Исследования в области интеллектуальных транспортных систем охватывают широкий спектр задач — от оптимизации маршрутов до моделирования динамики спроса и распределения транспортных средств. Существенный вклад в развитие моделей коллективного использования такси внесён в работе Y. Cao, где предложена оптимизационная модель маршрутизации для ride-hailing с учётом как интересов системы, так и справедливого распределения выгод между участниками [1]. Модель рассматривает ограничения по

вместимости, временным окнам и выгоде водителей и пассажиров, а решение задачи осуществляется с помощью генетического алгоритма.

Другим направлением является учёт социальных факторов и структуры маршрута при формировании совпадений. В работе O. F. Aydin предложен алгоритм ride-sharing, который использует разбиение маршрута на сегменты и вводит показатель социальной совместимости участников (пол, возраст, род занятий, готовность к совместным поездкам) [2]. Показано, что использование разбиения маршрутов и социальных параметров позволяет увеличить число совпадений при дефиците водителей.

С точки зрения алгоритмической сложности проблема распределения поездок формализуется как комбинаторная оптимизационная задача. X. Bei и S. Zhang рассматривают постановку trip–vehicle assignment и показывают, что задача является NP-трудной, предлагая приближённый алгоритм с теоретической оценкой 2,5 и эмпирическим приближением порядка 1,1–1,2 на синтетических данных [3]. Эти результаты подтверждают, что даже относительно простые стратегии выбора ближайшего водителя могут обладать высокой практической эффективностью.

Архитектурные и организационные аспекты систем такси исследуются в рамках мультиагентных моделей. В работе A. Alshamsi городская территория рассматривается как множество взаимодействующих агентов, представляющих районы, которые самостоятельно перераспределяют свободные автомобили. Показано, что самоорганизующаяся система позволяет снизить среднее время ожидания пассажиров по сравнению с традиционными статическими схемами [4].

Экономические особенности сервисов такси и ride-hailing подробно анализируются J. Angrist, которые сравнивают комиссионную модель вознаграждения водителей в Uber с классической арендной схемой традиционного такси [5]. Авторы показывают, что пропорциональная комиссия и отсутствие фиксированной аренды делают работу в цифровой платформе более привлекательной для водителей с низкой и средней загрузкой, что важно учитывать при проектировании новых систем.

Точность работы алгоритмов распределения во многом зависит от корректности геолокационных данных. В исследованиях по IP-геолокации и самонастраивающимся моделям позиционирования показывается, что корректировка координат и использование гибридных методов позволяет повысить точность определения местоположения и, как следствие, улучшить качество маршрутизации и расчёта расстояний [6]. Это особенно актуально для систем, где расстояние между водителем и пассажиром является ключевым параметром при назначении.

Наконец, обобщающие работы по динамическому ride-sharing и оптимизации маршрутов отмечают, что для реальных городских систем необходим баланс между сложностью алгоритмов и возможностью их внедрения в прикладные программные решения [7]. В этом контексте разработка лёгких и расширяемых архитектур серверной части, основанных на современных веб-фреймворках и СУБД, представляет собой важное направление, к которому относится и разрабатываемая в данной статье система.

Разработка информационной системы городского такси опирается на сочетание современных веб-технологий, методов обработки геолокационных данных и алгоритмов распределения транспортных ресурсов. Архитектура сервиса включает серверную часть, реализованную на FastAPI, реляционную базу данных PostgreSQL, а также набор REST-интерфейсов, обеспечивающих взаимодействие между клиентскими приложениями водителей и пассажиров. Данный раздел описывает структурную организацию системы и методы, применяемые при обработке данных.

Архитектура системы

Система построена по принципу клиент–серверного взаимодействия. Серверная часть реализует функциональные модули: регистрацию пользователей, авторизацию, создание заказов, назначение водителей, обновление координат, завершение поездок и хранение данных. Логика системы разделена следующим образом:

1. **Модуль пользователей.**
2. Отвечает за создание и аутентификацию двух категорий пользователей — водителей и пассажиров. Пароли хранятся в хэшированном виде с использованием устойчивого криптографического алгоритма bcrypt, что повышает безопасность учётных данных.
3. **Модуль заказов.**
4. Обеспечивает создание нового заказа, проверку наличия активных поездок и сохранение параметров: координаты отправления и назначения, время создания, статус обработки и идентификатор назначенного водителя.
5. **Модуль обработки геолокации.**
6. Для вычисления расстояния между участниками используется метод geodesic библиотеки *geopy*, основанный на эллипсоидальной модели Земли. Такой подход обеспечивает высокую точность при расчёте расстояний на небольших и средних участках пути, что соответствует требованиям систем такси. Методы коррекции геолокации и важность точности координат обоснованы в исследовании [6].
7. **Модуль назначения водителя.**
8. Реализация опирается на выбор водителя с минимальным расстоянием до пассажира, что соответствует приближённому решению задачи trip–vehicle assignment, описанной в работах [1–3]. Такой метод обеспечивает высокую вычислительную эффективность и надёжен для систем с малым и средним количеством водителей.
9. **Модуль управления статусами.**
10. Система отслеживает состояния водителей: *online, offline, free, busy*. После завершения поездки статус водителя автоматически меняется на *free*, что позволяет обеспечить непрерывное обновление пула доступных транспортных средств.

Структура базы данных

База данных включает три основные таблицы:

- **drivers** — хранит данные о водителях, включая их текущее местоположение и статус;
- **passengers** — содержит информацию о пассажирах;
- **orders** — фиксирует параметры заказа, включая координаты, назначенного водителя, стоимость и время завершения.

Связи между таблицами реализованы через внешние ключи: один пассажир может создавать множество заказов, один водитель может выполнять множество поездок, но только последовательно. Такая реляционная структура типична для транспортных систем и позволяет эффективно выполнять выборки по активным заказам и состояниям участников.

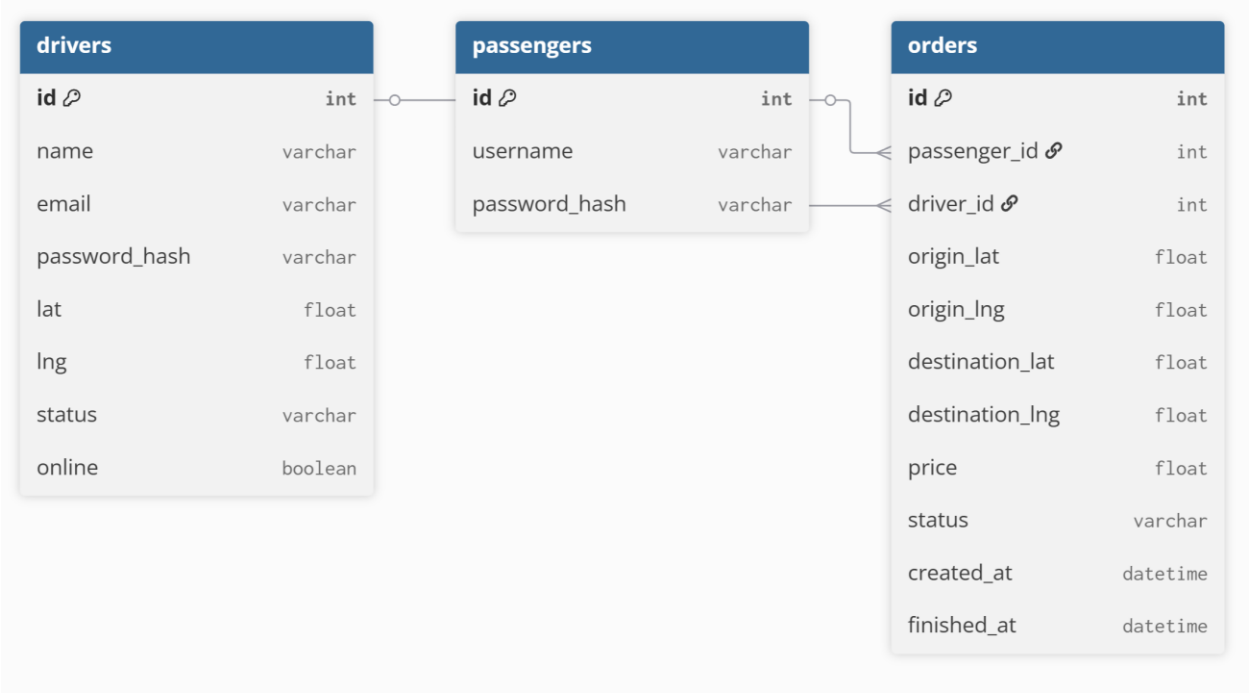

Рисунок 1 – ER-диаграмма базы данных системы городского такси.

Модель назначения водителя

Задача назначения решается на основе минимизации расстояния между пассажиром и каждым доступным водителем. Формально алгоритм определяется выражением:

$$Assign(p) = \arg\min_{d \in D_{free}} D(d_{loc}, p_{loc})$$

где D_{free} — множество свободных водителей,

d_{loc} — координаты водителя,

p_{loc}— координаты пассажира,

$D(\cdot)$ — расстояние, вычисляемое с использованием Google Distance Matrix API.

Подобные модели минимизации описаны в работах [1], [3], а также используются в многоагентных подходах [4]. В отличие от сложных оптимизационных алгоритмов, данный метод обладает линейной вычислительной сложностью и позволяет назначить водителя в режиме реального времени даже при значительном количестве активных пользователей.

Технологическая платформа

Система разработана на основе следующих технологий:

- **FastAPI** — асинхронный фреймворк, обеспечивающий высокую скорость обработки запросов и автоматическую генерацию документации.
- **PostgreSQL** — реляционная СУБД, обеспечивающая транзакционность операций.
- **SQLAlchemy** — ORM-слой, обеспечивающий связь между Python-кодом и базой данных.
- **geopy** — библиотека для точного вычисления расстояний между географическими точками.
- **bcrypt** — криптографическая библиотека для безопасного хранения паролей.

Выбор этих инструментов обусловлен их устойчивостью, открытостью и широким применением в промышленной разработке.

Результаты и их обсуждение

Экспериментальная часть исследования направлена на проверку корректности работы разработанной информационной системы, включая логику создания заказов, авторизации,

назначения водителей, обработки геолокационных данных и обновления состояний участников. Все эксперименты проводились в контролируемой среде с использованием инструментов Insomnia для отправки HTTP-запросов и pgAdmin для наблюдения за состоянием базы данных PostgreSQL.

1. Экспериментальная среда

Система развернута локально и функционирует под управлением FastAPI. База данных PostgreSQL содержит три таблицы: *drivers*, *passengers* и *orders*, структурированные для обеспечения целостности данных и корректного выполнения запросов. Для имитации сценариев работы сервиса были созданы тестовые аккаунты водителей и пассажиров.

Сервер предоставляет REST-интерфейс, что позволяет проводить эксперименты в форме последовательных HTTP-запросов, отражающих реальные действия пользователей.

2. Проверка регистрации и авторизации

Эксперименты начались с тестирования механизмов регистрации и входа. Запросы выполнялись методом:

POST /auth/register

POST /auth/login

Система корректно обрабатывала регистрацию пассажиров и водителей, создавая записи в базе данных и хэшируя пароли с использованием bcrypt. При авторизации водителей дополнительно изменялся статус: *online = True, status = "free"*, что позволяло включить их в пул доступных для назначения.

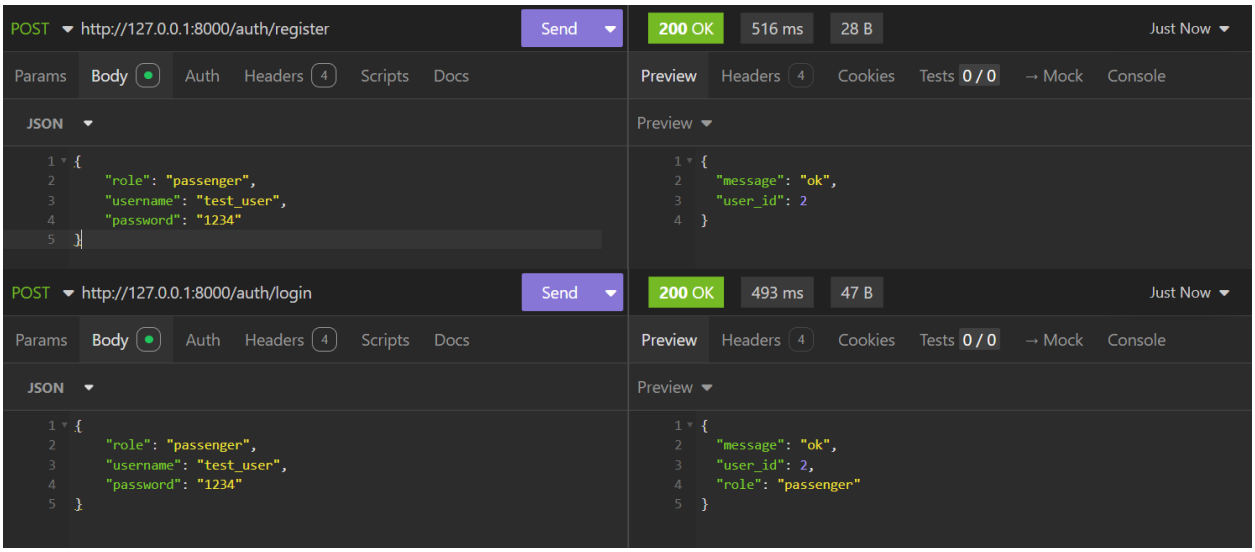

Рисунок 2 — Пример выполнения запросов регистрации и авторизации.

Проверка конкурсных ситуаций (неверный пароль, несуществующий пользователь) показала корректную генерацию исключений и возврат HTTP-ошибок.

3. Создание заказа пассажиром

Создание нового заказа осуществлялось с помощью запроса:

POST /order/create

Пассажир передавал координаты начала и конца пути. Система выполняла следующие проверки:

1. наличие пассажира в базе;
2. отсутствие активного заказа со статусом, отличным от *finished*;
3. корректность координат;

4. создание записи в таблице *orders*.

При успешном создании заказ фиксировался в базе со статусом *waiting*.

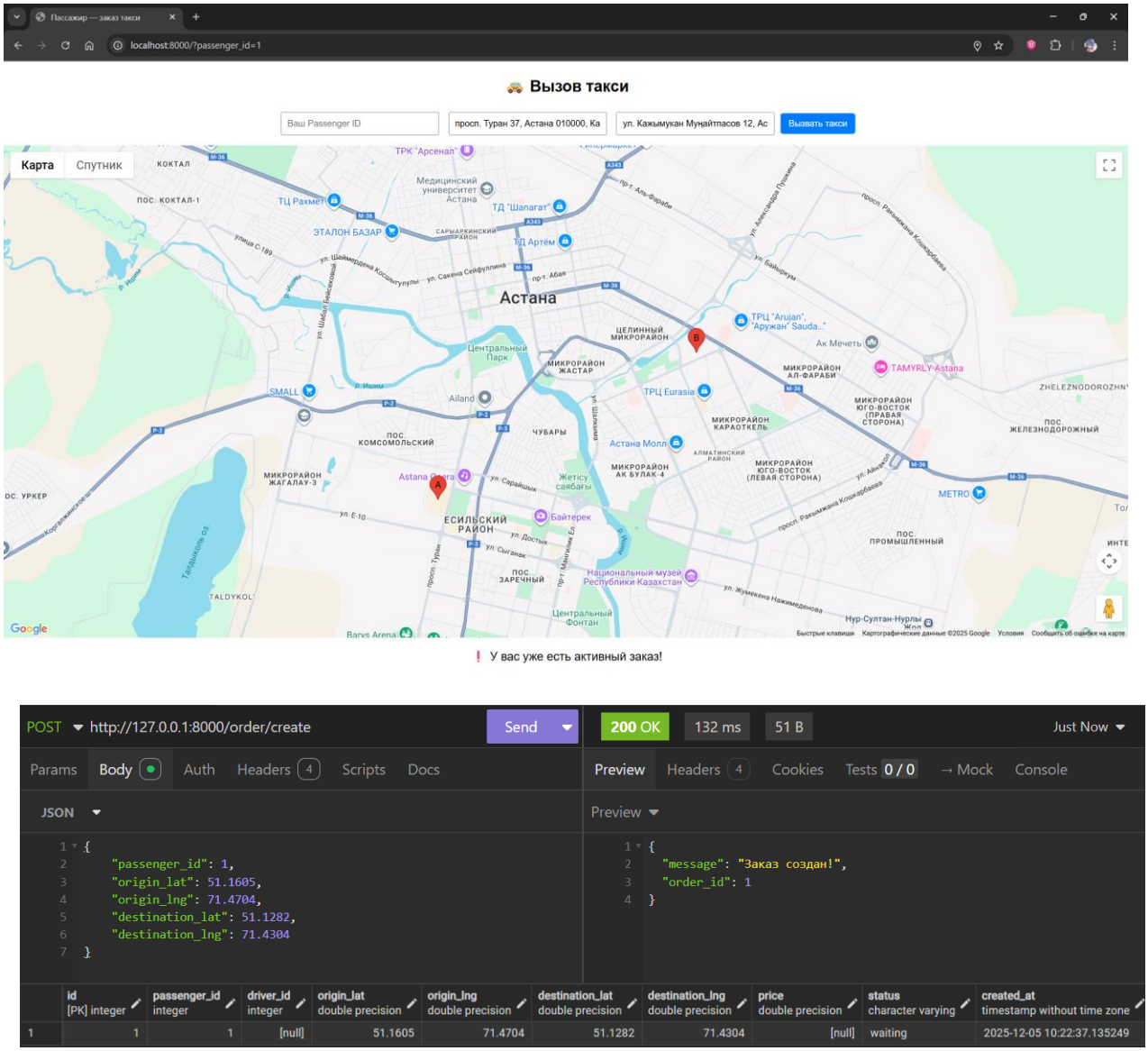

Рисунок 3 — Создание нового заказа и запись в таблице orders.

4. Тестирование алгоритма назначения водителя

Алгоритм назначения водителя проверялся последовательным вызовом метода:
GET /order/find_driver/{order_id}

Система выполняет выбор ближайшего водителя среди тех, кто имеет статус *free* и находится онлайн. Расстояние вычисляется по геодезической метрике:

$$D = geodesic((lat_d, lng_d), (lat_p, lng_p))$$

Если заказ ещё не назначен, система:

1. формирует множество доступных водителей;
2. вычисляет расстояние до каждого;
3. выбирает минимум;
4. обновляет статус водителя на *busy*;
5. сохраняет стоимость поездки в таблицу *orders*.

Стоимость определялась по формуле:

$$Price = \max(500,\ 150 \cdot D)$$

что соответствует тарифной модели прототипа.

Повторные вызовы метода возвращали уже назначенного водителя, что подтверждало устойчивость состояния заказа.

5. Изменение геолокации водителя

Механизм обновления координат тестировался запросом:

POST /drivers/{driver_id}/location

В ходе эксперимента координаты изменялись вручную, чтобы оценить реакцию системы при повторном поиске водителя. Система корректно обновляла значения в таблице *drivers*, что приводило к изменению ближайшего водителя при повторных запросах поиска.

Эксперимент подтвердил стабильную работу механизма динамической геолокации.

6. Завершение поездки

Функция завершения заказа проверялась методом:

PATCH /drivers/{driver_id}/finish

Система:

1. находила активный заказ, назначенный водителю;
2. изменяла статус заказа на *finished*;
3. фиксировала время завершения;
4. возвращала статус водителя к *free*.

Проверка в pgAdmin показала корректное обновление всех связанных полей, отсутствие рассинхронизации статусов и возможность немедленного принятия новых заказов.

7. Наблюдение за состоянием базы данных

В ходе всех экспериментов структура и содержимое таблиц отслеживались через pgAdmin. Особое внимание уделялось:

- корректности статусов водителей (*online*, *free*, *busy*),
- отсутствию более одного активного заказа у пассажира,
- точности расчёта расстояния и сохраняемой стоимости,
- корректности внешних ключей между *orders* и *drivers/passengers*.

	id [PK] integer	passenger_id integer	driver_id integer	origin_lat double precision	origin_lng double precision	destination_lat double precision	destination_lng double precision	price double precision	status character varying	created_at timestamp without time zone
1	1	1	[null]	51.1325301	71.40373029999999	51.1565232	71.47104519999999	[null]	waiting	2025-12-05 04:21:27.86063
2	2	2	[null]	51.1605	71.4704	51.1282	71.4304	[null]	waiting	2025-12-05 04:39:14.398692

Рисунок 4 — Фрагмент таблицы orders после выполнения запросов.

Все тесты показали:

- устойчивую работу REST-интерфейсов;
- корректное назначение водителей по минимальному расстоянию;
- корректное обновление координат;
- правильную логику завершения заказов;
- отсутствие логических ошибок в переходах состояний.

Система демонстрирует стабильность и готовность к дальнейшему расширению, включая добавление динамической маршрутизации, визуализации положения водителей и интеграции дополнительных тарифных моделей.

Результаты экспериментов подтвердили корректность работы ключевых компонентов информационной системы городского такси и показали применимость выбранных алгоритмов распределения и обработки геолокационных данных. Серверная часть стабильно выполняла все этапы жизненного цикла заказа — от его создания до завершения поездки.

Механизмы регистрации и авторизации показали устойчивость: система корректно обрабатывала роли пользователей, обеспечивала безопасное хранение данных и правильное изменение статуса водителя при входе. Создание заказов также прошло без логических конфликтов: пассажир не мог иметь более одного активного заказа, а статусы корректно фиксировались в базе.

Алгоритм назначения водителя стабильно выбирал ближайшего исполнителя на основе геодезического расстояния, что соответствует теоретическим моделям оптимального распределения. Изменение координат водителей приводило к ожидаемому перераспределению, что подтверждает корректность обработки геолокации.

Завершение поездки выполнялось корректно: статус заказа переходил в *finished*, а водитель становился доступным для новых заявок. Это обеспечивало непрерывность работы системы и согласованность данных.

Полученные результаты демонстрируют устойчивость и надёжность архитектуры FastAPI + PostgreSQL. Система может служить основой для дальнейшего развития, включая интеграцию алгоритмов маршрутизации, динамических тарифов и аналитических модулей.

Выводы

В ходе исследования была разработана и экспериментально апробирована информационная система городского такси, ориентированная на автоматизацию ключевых процессов обслуживания пассажиров. Предложенная архитектура, основанная на FastAPI и PostgreSQL, показала себя эффективной для реализации основных функций сервиса: регистрации пользователей, создания заказов, назначения водителей и управления состояниями в реальном времени. Проведённые эксперименты продемонстрировали корректность обработки данных, устойчивость REST-интерфейсов и соблюдение логики жизненного цикла поездки.

Результаты подтверждают, что выбранный алгоритм поиска ближайшего водителя обеспечивает стабильную работу и соответствует эвристикам, рассматриваемым в современных исследованиях задач распределения транспортных средств. Применение геодезической метрики позволило добиться достаточной точности расчёта расстояний для условий городской среды. Важным является и то, что структура базы данных обеспечивает целостность и последовательность обработки заказов.

Разработанная система может служить основой для построения полнофункциональной платформы городского такси, а также для дальнейших научных экспериментов, связанных с оптимизацией маршрутов, прогнозированием спроса, использованием динамических тарифов и моделированием поведения участников транспортной сети. Дальнейшее развитие решения может включать интеграцию картографических сервисов, мобильных клиентов и аналитических модулей.

Список литературы

1. Cao, Y., Wang, S., & Li, J. *The Optimization Model of Ride-Sharing Route for Ride-Hailing Considering Both System Optimization and User Fairness*. Sustainability, 13(902), 2021. 18 p.
2. Aydin, O. F., Gokasar, I., & Kalan, O. *Improving Ride-Sharing by Incorporating Route Splitting and Social Factors*. PLOS ONE, 15(3), 2020. 23 p.
3. Bei, X., & Zhang, S. *Algorithms for Trip-Vehicle Assignment in Ride-Sharing*. Proceedings of the AAAI Conference on Artificial Intelligence, 2018. pp. 3–9.

4. Alshamsi, A., Abdallah, S., & Rahwan, I. *Multiagent Self-Organization for a Taxi Dispatch System*. AAMAS 2009 – Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 21–28.
5. Angrist, J., Caldwell, S., & Hall, J. V. *Uber versus Taxi: A Driver's Eye View*. American Economic Journal: Applied Economics, 13(3), 2021, pp. 272–308.
6. Wang, H., Liu, X., Xu, B., et al. *IP Geolocation Self-Optimizing Model*. arXiv:2004.01531, 2020. 12 p.
7. Zha, L., Yin, Y., Du, Y. *Surge Pricing and Fleet Management in Ride-Sourcing Systems*. Article Text 14826, Journal Publication, 2020. 14 p.

References

1. Cao, Y., Wang, S., & Li, J. *The Optimization Model of Ride-Sharing Route for Ride-Hailing Considering Both System Optimization and User Fairness*. Sustainability, 13(902), 2021. 18 p.
2. Aydin, O. F., Gokasar, I., & Kalan, O. *Improving Ride-Sharing by Incorporating Route Splitting and Social Factors*. PLOS ONE, 15(3), 2020. 23 p.
3. Bei, X., & Zhang, S. *Algorithms for Trip-Vehicle Assignment in Ride-Sharing*. Proceedings of the AAAI Conference on Artificial Intelligence, 2018. pp. 3–9.
4. Alshamsi, A., Abdallah, S., & Rahwan, I. *Multiagent Self-Organization for a Taxi Dispatch System*. AAMAS 2009 – Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 21–28.
5. Angrist, J., Caldwell, S., & Hall, J. V. *Uber versus Taxi: A Driver's Eye View*. American Economic Journal: Applied Economics, 13(3), 2021, pp. 272–308.
6. Wang, H., Liu, X., Xu, B., et al. *IP Geolocation Self-Optimizing Model*. arXiv:2004.01531, 2020. 12 p.
7. Zha, L., Yin, Y., Du, Y. *Surge Pricing and Fleet Management in Ride-Sourcing Systems*. Article Text 14826, Journal Publication, 2020. 14 p.

Т.Т. Оспанова*, Р.Н. Серіков
т.ғ. к., доцент, Л.Н. Гумилев атындағы ЕҰУ, Астана, Қазақстан
магистрант, Л.Н. Гумилев атындағы ЕҰУ, Астана, Қазақстан
***Корреспондент авторы:** tleu2009@mail.ru

FASTAPI НЕГІЗІНДЕ ҚАЛАЛЫҚ ТАКСИГЕ АРНАЛҒАН АҚПАРАТТЫҚ ЖҮЙЕНІ ӘЗІРЛЕУ

Түйін

Мақалада қалалық такси қызметіне арналған ақпараттық жүйені әзірлеу және оның негізгі функционалдық мүмкіндіктерін эксперименттік тұрғыда тексеру нәтижелері қарастырылады. Жүйе FastAPI платформасы мен PostgreSQL дерекқорын пайдалану арқылы жүзеге асырылып, пайдаланушыларды тіркеу, тапсырыстарды жасау, жүргізушілерді тағайындау және геолокациялық деректерді нақты уақыт режимінде өңдеу сияқты негізгі процестерді автоматтандыруға мүмкіндік берді. Жақын жүргізушіні анықтау үшін геодезиялық қашықтықты есептеу әдісі қолданылып, қалалық жағдайларда жеткілікті дәлдік көрсетті. Жүргізуші мәртебелерін басқару механизмі эксперимент барысында күй өзгерістерінің тұрақты және дұрыс орындалатынын дәлелдеді. Алынған нәтижелер ұсынылған архитектураның тиімділігін растап, оның мүмкін болатын кеңею бағыттарын айқындады. Ұсынылған жүйе қалалық такси сервистерінің толыққанды платформасын әзірлеуге негіз бола алады.

Кілттік сөздер: ақпараттық жүйе, қалалық такси, FastAPI, геолокация, PostgreSQL, таратуды оңтайландыру, сандық қызметтер.

T.T. Ospanova*, R.N. Serikov
Cand.Tech.Sci., Associate Professor, L.N. Gumilyov ENU, Astana, Kazakhstan
Master's student, L.N. Gumilyov ENU, Astana, Kazakhstan
***Corresponding author's email:** tleu2009@mail.ru

DEVELOPMENT OF AN INFORMATION SYSTEM FOR URBAN TAXIS BASED ON FASTAPI

Abstract

The article presents the development of an information system for urban taxi services and the results of an experimental evaluation of its key functional capabilities. The system is implemented using the FastAPI framework and PostgreSQL, enabling the automation of essential processes such as user registration, order creation, driver assignment, and real-time geolocation data processing. A geodesic distance calculation method is applied to determine the nearest driver, providing sufficient accuracy for urban environments. The driver status management mechanism demonstrated stability and correct state transitions during testing. The obtained results confirm the effectiveness of the proposed architecture and highlight its potential for further enhancement, including functional expansion, integration of analytical modules, and the application of route optimization methods. The system can serve as a foundation for developing full-scale urban taxi service platforms.

Keywords: information system, urban taxi, FastAPI, geolocation, PostgreSQL, distribution optimization, digital services.