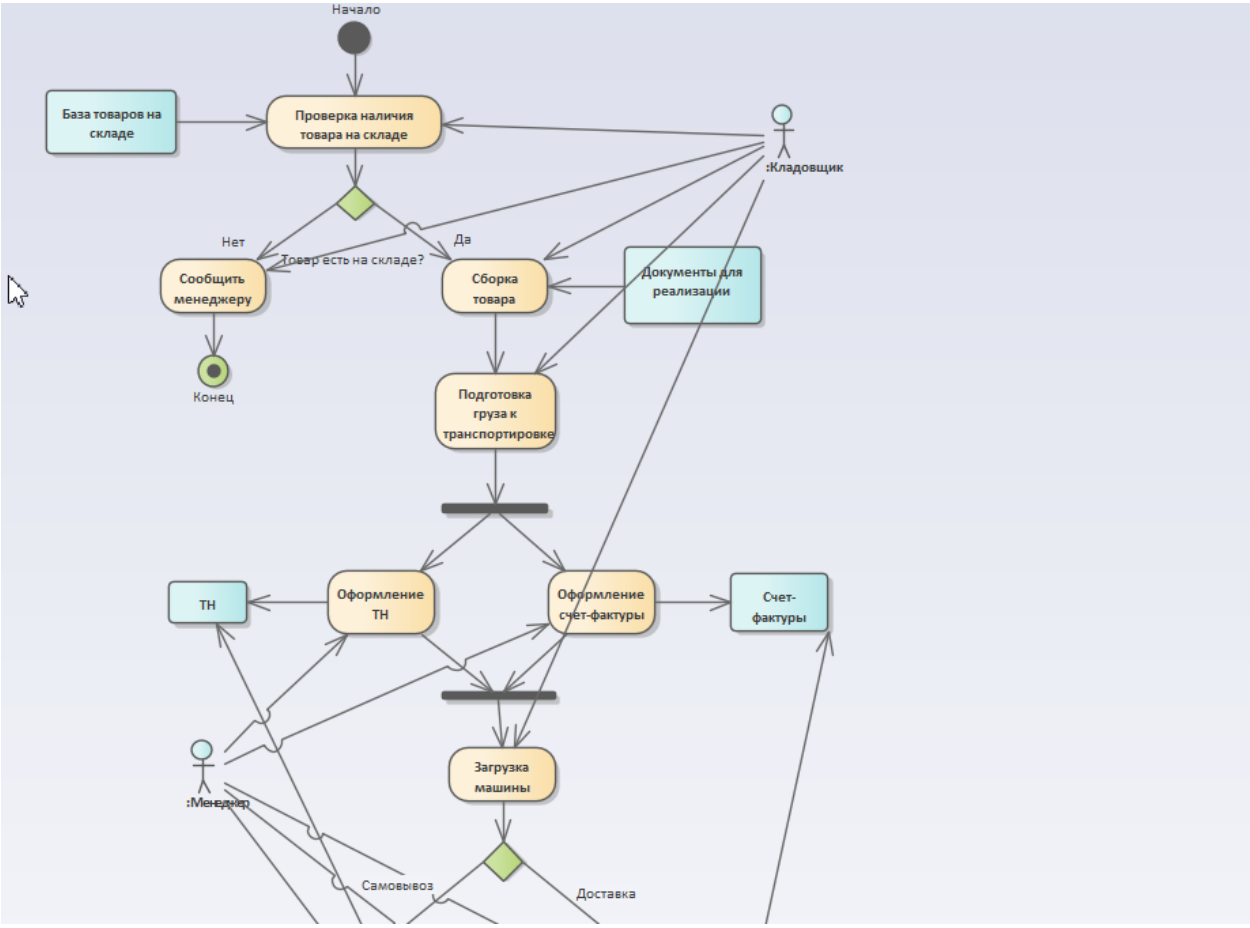
УДК 004

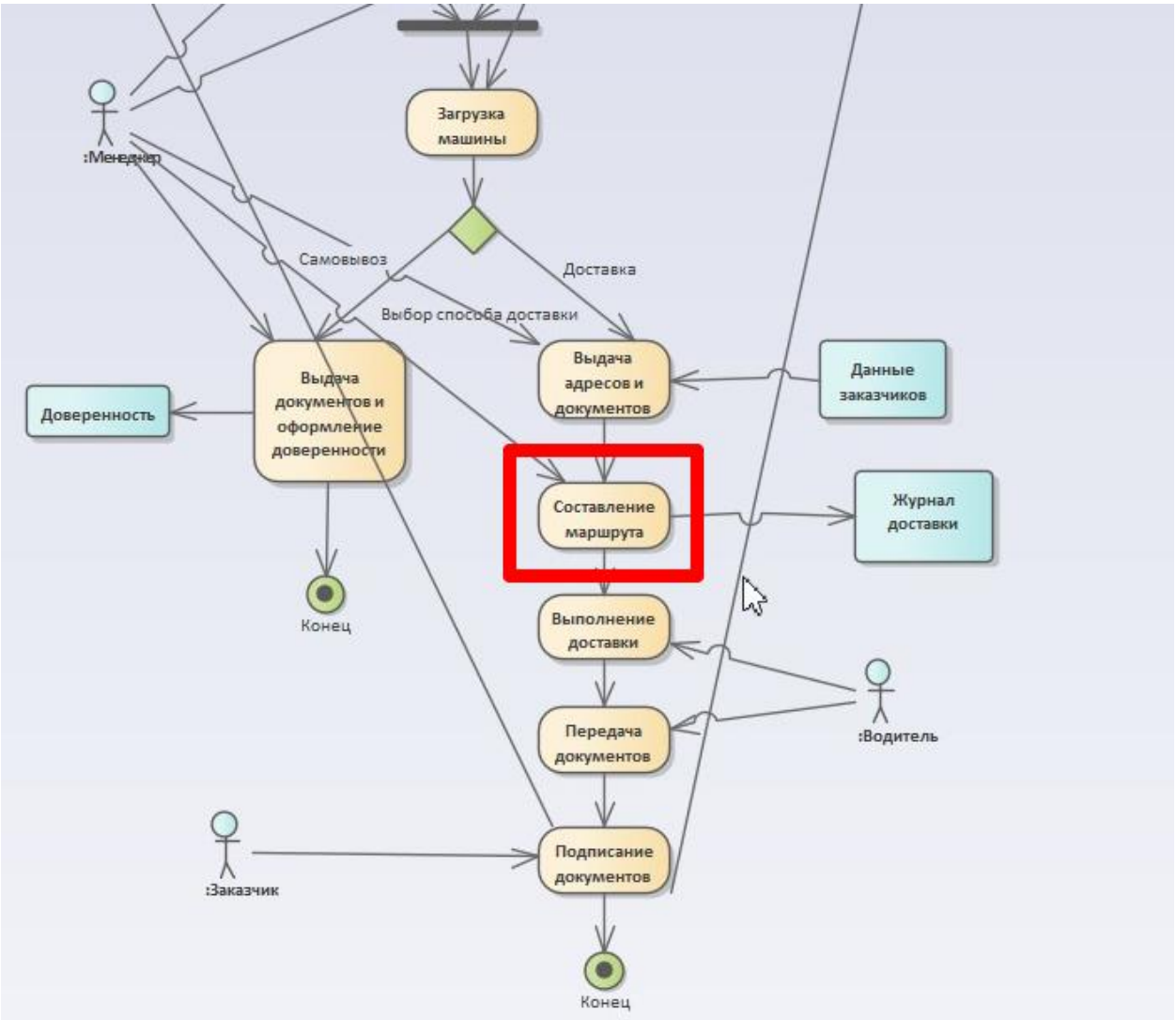
РАЗРАБОТКА СИСТЕМЫ ОПТИМАЛЬНОГО ПОСТРОЕНИЯ МАРШРУТА

Критонова В.Е., студент гр. ИТб-212, 4 курс.
Научный руководитель: Ванеев О.Н., к.н., доцент
Кузбасский государственный технический университет
имени Т.Ф. Горбачева
г. Кемерово

Современные навигационные сервисы широко используются для определения оптимальных маршрутов передвижения на транспорте и пешком. Выбор кратчайшего или быстрейшего пути позволяет сэкономить время в дороге, снизить транспортные расходы и избежать заторов. Несмотря на наличие популярных решений (Google Maps, Яндекс.Карты и др.), создание собственной системы построения маршрутов на основе открытых данных является актуальной задачей. Это позволяет получить независимую платформу навигации, адаптированную под специфические потребности (например, для локального использования или интеграции в специализированные приложения) без зависимости от сторонних API.

Разработка такой системы связана с необходимостью обработки больших объемов картографических данных и реализацией эффективных алгоритмов, что представляет собой индивидуальную инженерную задачу.





В данной работе рассматривается разработка системы оптимального построения маршрута с использованием языка C#, базы данных PostgreSQL/PostGIS и данных OpenStreetMap (OSM). Предлагаемое решение нацелено на построение клиент-серверного приложения с REST API, способного быстро вычислять оптимальные маршруты по дорожной сети.

Анализ существующих решений

На сегодняшний день существует ряд готовых решений для задачи построения маршрутов, как коммерческих, так и с открытым исходным кодом:

- **Коммерческие картографические сервисы** – такие как Google Maps, Яндекс.Карты, Mapbox – предоставляют API для прокладки маршрутов. Эти сервисы обладают высокой точностью и скоростью, учитывают пробки и разнообразные параметры маршрутов. Однако они являются закрытыми проприетарными решениями и их использование часто платное или ограничено по условиям лицензирования.
- **Open Source движки маршрутизации** – существуют открытые проекты, позволяющие строить маршруты на данных OSM. Например, **OSRM (Open Source Routing Machine)** – высокопроизводительный движок на C++ с реализацией алгоритма сокращающих иерархий; **GraphHopper** – маршрутизатор на Java,

эффективно обрабатывающий данные OSM и поддерживающий разные виды транспорта; **Valhalla** – еще один C++ движок, поддерживающий мультимодальные маршруты. Эти решения доступны для развертывания на собственном сервере и дают качественные результаты, однако требуют настройки и ресурсов для работы (предварительная обработка данных, поддержание сервера).

- **Базы данных с поддержкой маршрутизации** – расширение **PgRouting** для PostgreSQL/PostGIS предоставляет алгоритмы кратчайшего пути (Dijkstra, A*, др.), которые можно вызывать напрямую через SQL-запросы по данным дорожного графа, хранящимся в базе. Это упрощает интеграцию поиска маршрутов с хранением геоданных, но может уступать специализированным движкам в производительности на очень больших графах.
- **Библиотеки для разработки** – для языка C# существует библиотека **Itinero** (основанная на OsmSharp) и другие, позволяющие загрузить данные OSM и выполнять расчёт маршрута внутри своего приложения. Они облегчают реализацию, но требуют встраивания полного набора данных и алгоритмов в приложение.

Таким образом, разработчик имеет выбор между использованием готовых решений и созданием собственного. Готовые решения обеспечивают быстрый старт, однако могут быть избыточными или трудно модифицируемыми под специфичные требования. В рамках учебного проекта целесообразно реализовать собственную систему, опираясь на идеи существующих подходов. Это позволит глубже изучить технологию и при необходимости адаптировать алгоритмы и данные под конкретные условия.

Обоснование выбора технологий

Для реализации поставленной задачи выбран следующий стек технологий, исходя из их возможностей и совместимости между собой:

- **Язык программирования: C# (.NET)** – современный объектно-ориентированный язык, позволяющий быстро разработать надежное серверное приложение. C# обладает богатыми возможностями для работы с сетевыми запросами (ASP.NET Web API для создания REST-сервисов) и базами данных (ORM или библиотеки доступа к PostgreSQL). Также для .NET доступны библиотеки для работы с геоданными и алгоритмами, что ускоряет разработку.
- **База данных: PostgreSQL + PostGIS** – PostgreSQL является надежной реляционной СУБД с открытым исходным кодом, а расширение PostGIS добавляет поддержку геопространственных типов данных (точки, линии, полигоны) и функций. PostGIS удобно хранит карту дорог в виде графа: таблицы вершин и ребер с геометрией и атрибутами (длина, ограничения движения). Кроме того, с помощью **PgRouting** база данных может выполнять базовые расчеты маршрутов на стороне SQL, что может использоваться для проверки результатов или в качестве альтернативного метода.
- **Источники картографических данных: OpenStreetMap (OSM)** – открытая глобальная карта, данные которой свободно доступны для использования. OSM предоставляет подробную информацию о дорожной сети (дороги, перекрестки, ограничения поворотов, одностороннее движение и пр.), необходимую для построения графа маршрутов. Данные OSM могут быть загружены в PostGIS с

помощью утилит (например, osm2pgsql) или обработаны напрямую в приложении через библиотеки, что обеспечивает актуальность и детализацию картографической основы системы.

- **Архитектура: клиент-сервер, REST API** – выбран разделенный подход, при котором серверная часть занимается вычислением маршрутов и хранением данных, а клиентская часть отвечает за взаимодействие с пользователем. REST API на сервере позволяет вызывать функциональность построения маршрута через стандартные HTTP-запросы, что делает систему универсальной (клиентом может быть веб-приложение, мобильное приложение, десктоп-программа и т.д.). Такая архитектура облегчает масштабирование: сервер можно развивать и оптимизировать независимо от клиентов, а также обеспечивает возможность многопользовательской работы.

Выбор данных и инструментов обусловлен их открытостью и распространенностью. Использование OSM и PostgreSQL/PostGIS гарантирует невысокие затраты и широкую поддержку сообщества. C# и .NET обеспечивают кроссплатформенность и высокую производительность серверной логики. В совокупности, этот стек технологий позволит реализовать функциональную систему построения маршрутов, которую при необходимости можно расширять (например, альтернативные пути, и пр.).

Архитектура и компоненты системы

Система спроектирована по принципу разделения на несколько компонентов, каждый из которых выполняет свою роль. Основные компоненты системы оптимального построения маршрута:

- **Клиентское приложение** – интерфейс, через который пользователь задает начальную и конечную точки маршрута и получает результат. Это будет мобильное приложение и десктоп-программа. Клиент отображает карту и маршрут, отправляет запросы на сервер через интернет.

- **Серверное приложение (REST API)** – ядро системы, реализованное на C#. На сервере размещена логика обработки запросов: он принимает входные данные (координаты или адреса начала и конца пути), обращается к базе данных и модулю построения маршрута, рассчитывает оптимальный маршрут и возвращает его клиенту в виде удобного формата (например, списка точек polyline, GeoJSON или JSON с инструкциями).

- **База данных геоданных** – PostgreSQL с расширением PostGIS, где хранится модель дорожного графа. В базе содержатся таблицы дорог (ребер графа) с их геометрией, длиной, допустимыми направлениями движения, скоростными ограничениями и др. атрибутами, а также таблица узлов (перекрестков) с координатами. База позволяет эффективно выполнять географические запросы, например, поиск ближайшей к заданной точке дороги (для привязки пользовательского ввода к графу). Также база служит хранилищем для результатов, кэширования маршрутов или статистики (опционально).

- **Модуль алгоритмов маршрутизации** – компонент, отвечающий за выполнение поиска оптимального пути. Он может быть реализован как часть серверного приложения (класс/библиотека внутри C#-сервиса) или в виде функций в базе данных (например, вызов процедур PgRouting). В рамках нашей системы алгоритм

(A* или Contraction Hierarchies) будет интегрирован в серверное приложение, используя данные, полученные из базы. Этот модуль принимает на вход граф (или доступ к графу через БД) и параметры маршрута, а возвращает последовательность ребер от старта до финиша.

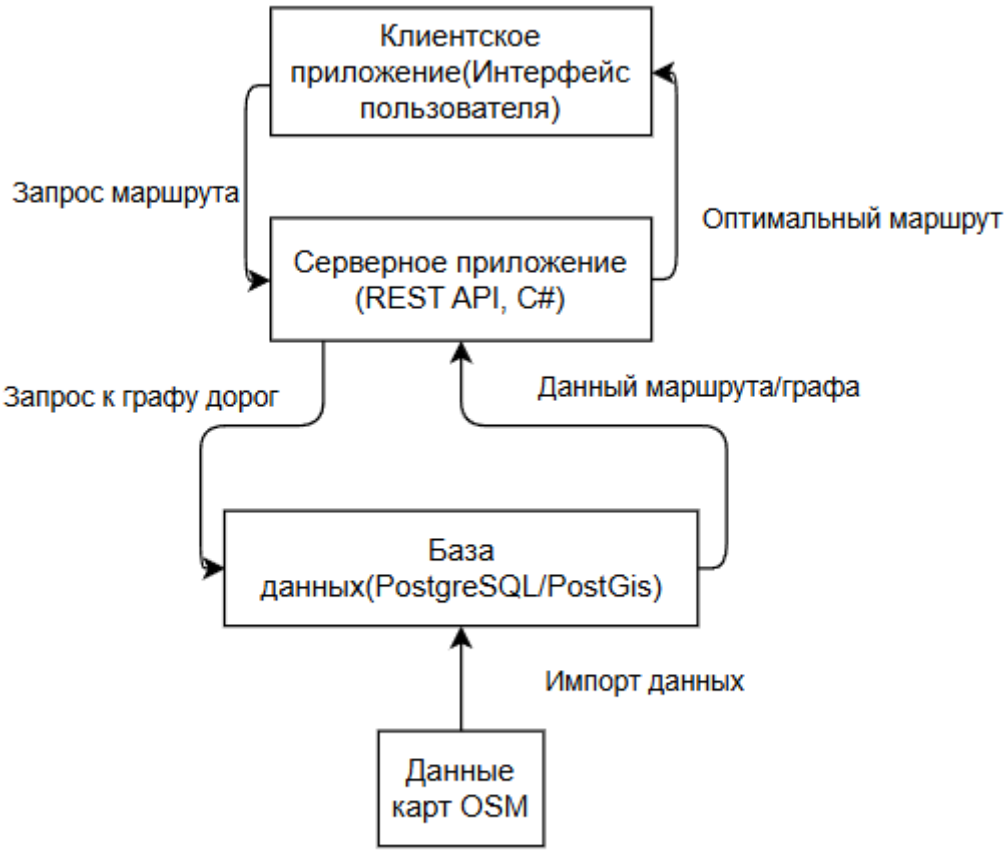


Рис. 1. Архитектура системы оптимального построения маршрута.

Система организована по принципу клиент-сервер: клиентское приложение посылает HTTP-запрос с параметрами маршрута на сервер, сервер обращается к базе данных для получения необходимой информации о графе дорог и использует модуль маршрутизации для расчёта пути. После этого результат в виде оптимального маршрута возвращается на клиент.

На рисунке показана схема взаимодействия компонентов, включая этап предварительного импорта данных OSM в базу данных. Архитектура обеспечивает разделение ответственности: клиент отвечает только за отображение и ввод, вся тяжелая обработка выполняется на сервере, что позволяет масштабировать решение.

Схема взаимодействия модулей:

1. Пользователь задаёт на клиенте параметры маршрута (например, адрес или координаты начальной и конечной точек) и отправляет запрос на построение маршрута – эта информация передается через REST API на сервер.

2. Серверное приложение принимает запрос, преобразует входные адреса в координаты (при необходимости) и определяет ближайшие узлы графа для точек старта и финиша. Затем сервер обращается к базе данных PostGIS, чтобы получить фрагмент графа дорог или необходимую информацию (например, вес ребер, топологию) для алгоритма. Модуль поиска маршрута вычисляет кратчайший путь между указанными точками на основе полученных данных (с использованием выбранного алгоритма).

3. Построенный оптимальный маршрут (последовательность точек или ребер) возвращается сервером клиентскому приложению. Клиент получает данные маршрута через API и отображает маршрут на карте для пользователя, сопровождая его информацией (длина пути, примерное время в пути, пошаговые указания и т.д.). В случае, если маршрут не найден (например, точки находятся на несвязанных участках графа), сервер возвращает сообщение об ошибке или отсутствии пути.

Архитектура допускает обновление картографических данных: при выходе обновлений OSM можно повторно импортировать их в базу, что не затрагивает работу клиентской части. Кроме того, REST API позволяет легко подключать новые типы клиентов и интегрироваться с другими системами (например, передавать вычисленные маршруты в логистические приложения).

Список литературы:

1. Ахо А. В., Хопкрофт Д. Э., Ульман Дж. Д. Структуры данных и алгоритмы. – М.: Мир, 1983. – 440 с.

2. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – 2-е изд. – М.: «Вильямс», 2005. – 1296 с.

3. Официальная документация OpenStreetMap [Электронный ресурс] – Режим доступа: https://wiki.openstreetmap.org (дата обращения: 01.04.2025).

4. Официальная документация PostgreSQL [Электронный ресурс] – Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 01.04.2025).

5. Официальная документация PostGIS [Электронный ресурс] – Режим доступа: https://postgis.net/documentation/ (дата обращения: 01.04.2025).

6. Гудков А. В. ГИС-технологии и базы геоданных: учебное пособие. – СПб.: Питер, 2019. – 320 с.

7. Geisberger R., Sanders P., Schultes D., Delling D. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks // Experimental Algorithms. – Springer, 2008. – С. 319–333.

8. Zheng J., Fu Q. A practical guide to A* and its variants in road networks // Journal of Advanced Transportation, 2020. – Т. 2020. – С. 1–12.

9. Официальная документация Osm2pgsql [Электронный ресурс] – Режим доступа: https://osm2pgsql.org/ (дата обращения: 01.04.2025).

10. Itinero (OsmSharp) – GitHub Repository [Электронный ресурс] – Режим доступа: https://github.com/itinero/itinero (дата обращения: 01.04.2025).