МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Факультет (институт, филиал) 3, Институт информационных систем и технологий**

**Кафедра ИСТ**

**Специальность 09.04.01 «Информатика и вычислительная техника» Группа 3ВТИ-1ДМ-052**

**Квалификация магистр**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА**

**(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

На тему: Анализ технологии построения системы обработки геоданных на примере

web-сервиса «Coordinate».

Автор диссертации Нагорный Андрей Александрович ( )

(фамилия, имя, отчество)

Научный руководитель Павлов Виталий Юрьевич ( )

(фамилия, имя, отчество)

Рецензент ( )

(фамилия, имя, отчество)

**К з а щ и т е д о п у с т и т ь**

Завкафедрой Павлов В. Ю. ( )

(фамилия, инициалы)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

Москва 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Факультет (институт, филиал) 3, Институт информационных систем и технологий**

**Кафедра ИСТ**

**Специальность 09.04.01 «Информатика и вычислительная техника» Группа 3ВТИ-1ДМ-052**

**Квалификация магистр**

**РАЗДАТОЧНЫЙ МАТЕРИАЛ**

**К**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ МАГИСТРА**

**(МАГИСТЕРСКОЙ ДИССЕРТАЦИИ)**

На тему: Анализ технологии построения системы обработки геоданных на примере

web-сервиса «Coordinate».

Автор диссертации Нагорный Андрей Александрович ( )

(фамилия, имя, отчество)

Научный руководитель Павлов Виталий Юрьевич ( )

(фамилия, имя, отчество)

Рецензент ( )

(фамилия, имя, отчество)

**К з а щ и т е д о п у с т и т ь**

Завкафедрой Павлов В. Ю. ( )

(фамилия, инициалы)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

Москва 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**О Т З Ы В**

**НАУЧНОГО РУКОВОДИТЕЛЯ**

**Научный руководитель** Павлов В.Ю., к.ф.м.н., зав. кафедрой «ИСТ».

(фамилия, инициалы, ученая степень, ученое звание, должность и место работы)

**Студент** Нагорный Андрей Александрович

**Факультет (институт, филиал) 3, Институт информационных систем и технологий**

**Кафедра ИСТ**

**Специальность 09.04.01 «Информатика и вычислительная техника» Группа 3ВТИ-1ДМ-052**

**Квалификация (степень) магистр**

**Тема диссертации:** Анализ технологии построения системы обработки геоданных на

примере web-сервиса «Coordinate».

Работа проверена на объем заимствования. % заимствования -

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 20 г. Руководитель

(подпись)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Факультет (институт, филиал) 3, Институт информационных систем и технологий**

**Кафедра ИСТ**

**Специальность 09.04.01 «Информатика и вычислительная техника» Группа 3ВТИ-1ДМ-052**

**Квалификация магистр**

**УТВЕРЖДАЮ**

Завкафедрой Павлов В. Ю.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20 г.

*П Л А Н Р А Б О Т Ы*

**над выпускной квалификационной работой магистра (магистерской диссертацией)**

**Студент** Нагорный Андрей Александрович

(фамилия, имя, отчество)

**Руководитель** Павлов Виталий Юрьевич

(фамилия, имя, отчество

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_к.ф.-м.н., зав. кафедрой «ИСТ»

ученая степень, ученое звание, должность и место работы)

**1. Наименование темы:** Анализ технологии построения системы обработки геоданных на примере web-сервиса «Coordinate».

**2. Срок сдачи студентом законченной работы**

**3. Перечень подлежащих разработке разделов и этапы выполнения работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№**  **п/п** | **Наименование раздела или этапа** | **Трудоёмкость в % от**  **полной трудоёмкости**  **работы** | **Срок выполнения** | **Примечание** |
| **1** | **Введение** | **10** | **07.04.17** |  |
| **2** | **Постановка задач** | **10** | **07.04.17** |  |
| **3** | **Описание системы** | **15** | **07.04.17** |  |
| **4** | **Исследование системы** | **40** | **28.04.17** |  |
| **5** | **Модернизация системы** | **20** | **26.05.17** |  |
| **6** | **Заключение** | **5** | **26.05.17** |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**4. Перечень иллюстративно-графических материалов:**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование** | **Количество листов** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**5. Исходные материалы и пособия**

1. Руби С.,Томас Д., Хэнссон Д. Rails 4. Гибкая разработка веб-приложений. изд. М. и др.: Питер, 2014, 448с.
2. Крис Дж. Д. SQL и реляционная теория. Как грамотно писать код на SQL. изд. М. и др.: Символ-Плюс, 2010, 480с.
3. Т.Кормен, Ч.Лейзерсон, Р.Ривест. Алгоритмы. Построение и анализ. изд. М. и др.: Вильямс, 2012, 1290с.
4. Г. Шилд. Java 7. Полное руководство. М. и др.: Вильямс, 2012, 1104с.
5. Немет Э., Снайдер Г. Unix и Linux. Руководство системного администратора. изд. М. и др.: Вильямс, 2012, 1300с.

**6. Дата составления плана**

Руководитель

(подпись)

Студент

(подпись)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**ЗАКЛЮЧЕНИЕ**

**РЕЦЕНЗЕНТА**

**О ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ МАГИСТРА**

**(МАГИСТЕРСКОЙ ДИССЕРТАЦИИ)**

**Студента Нагорного Андрея Александровича**

(фамилия, имя, отчество)

**Факультет (институт, филиал) 3, Институт информационных систем и технологий**

**Кафедра ИСТ**

**Специальность 09.04.01 «Информатика и вычислительная техника» Группа 3ВТИ-1ДМ-052**

**Квалификация (степень) магистр**

**Рецензент**

(фамилия, инициалы, ученая степень, ученое звание, должность и место работы)

**Отмеченные достоинства:**

**Отмеченные недостатки:**

**Заключение:**

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 20 г. Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Аннотация**

Web-сервис “Coordinate” был разработан автором данной работы в ходе выпускной квалификационной работы бакалавра от 2015 года. Web-сервис “Coordinate” позволяет производить мониторинг транспорта, снаряженным специальной программно-аппаратной частью системы: трекером, представляющее собой устройство под управлением ОС Android с установленном специальным ПО, отвечающее за сбор GPS координат и отправкой их на сервере. Web-сервис предназначен для сбора информации о передвижении личного транспорта пользователей, вычислении статистических и аналитических данных и предоставления доступа к ним.

Сервис окончательно введен в эксплуатацию в 2015 году. За период эксплуатации был выявлен ряд проблем, анализ и решение которых послужили основанием данной выпускной квалификационной работы.

**Оглавление**

[1. Введение 4](#_Toc482545661)

[2. Постановка задач 4](#_Toc482545662)

[3. Описание системы 5](#_Toc482545663)

[3.1 Описание объектов исследования. 5](#_Toc482545664)

[3.2 Опрос респондентов 6](#_Toc482545665)

[3.3 Анализ востребованности системы. 9](#_Toc482545666)

[3.4 Основные технологии построения систем мониторинга 12](#_Toc482545667)

[3.5 Описание существующей системы “Coordinate" 13](#_Toc482545668)

[4. Исследование системы “Coordinate” 16](#_Toc482545669)

[4.1 Методология исследования 16](#_Toc482545670)

[4.2 Анализ хранения геоданных в системе “Coordinate” 17](#_Toc482545671)

[4.3 Анализ обработки геоданных в системе “Coordinate” 30](#_Toc482545672)

[4.4 Анализ сбора геоданных в системе “Coordinate” 39](#_Toc482545673)

[5. Экономический анализ системы 46](#_Toc482545674)

[5.1 Расчет экономических затрат 46](#_Toc482545675)

[5.2 Расчет экономической рентабельности 46](#_Toc482545676)

[6. Заключительная часть 46](#_Toc482545677)

[6.1 Результаты исследования 46](#_Toc482545678)

[6.2 Заключение 48](#_Toc482545679)

[7. Список сокращений и условных обозначений 50](#_Toc482545680)

[8. Список литературы 50](#_Toc482545681)

[Приложение А 51](#_Toc482545682)

[Приложение Б 57](#_Toc482545683)

[Приложение В 66](#_Toc482545684)

[Приложение Г 67](#_Toc482545685)

[Приложение Д 70](#_Toc482545686)

[Приложение Е 73](#_Toc482545687)

# Введение

Наиболее важной проблемой сервиса оказалось быстродействие системы в целом. Несмотря на ограниченное число пользователей, за год использования накопился большой объем данных, обработка которых стала причиной дополнительной нагрузки на сервер, что привело к увеличению времени выполнения запросов, возникновении ошибок системы и даже, потери данных. Решения, найденные в ходе ВКР и представленные в данном отчете позволяют увеличить как быстродействие системы в целом, так и отдельных компонентов.

Еще одной важной проблемой стал рост данных, при постоянных характеристиках серверных компонент. Серверу необходимо произвести масштабирование, расчет которого проведен в данной работе.

Кроме того, в работе сервиса были выявлены некоторые архитектурные ошибки, исправление которых способно повысить быстродействие и надежность системы. В работе был приведен анализ таких возможных улучшений.

Модернизация улучшение системы является актуальной задачей, способной повысить спрос на систему, экономические показатели и обеспечить более высокий уровень безопасности и оптимизации личных данных пользователей.

# Постановка задач

Целью данной работы является исследование существующей системы мониторинга транспорта, с ее последующей модернизацией. Необходимо провести исследование объектов системы, содержащие наиболее критичные проблемы, скопившиеся за годы функционирования системы, и найти способы их решения, или обосновать, почему эти решения невозможны.

Пользовательское ПО должно предоставлять санкционированный доступ к пользовательским данным, вне зависимости от платформы и рабочего окружения пользователя (должна быть обеспечена кроссплатформенность). Сервис в целом должен обеспечивать надежную и бесперебойную работу в круглосуточном режиме.

Кроме того, необходимо уделить внимание аспектам безопасности конфиденциальных данных пользователей, соответствию правовым положениям по хранению персональных данных, а также экономической рентабельности данных.

Для достижения этих целей был выделен ряд задач:

* анализ существующих решений, используемых в сервисе “Coordinate”;
* анализ аналогичных продуктов;
* анализ структурной модели сервиса;
* анализ и модернизация ПО для трекера;
* анализ и модернизация БД;
* анализ и модернизация ПО для сервера;
* анализ и модернизация пользовательского интерфейса;
* проработка вопросов безопасности;
* тестирование;
* документирование.

# Описание системы

## Описание объектов исследования.

Объектами данной выпускной работы служат как сама информационная система “Coordinate”, так и ее основные компоненты: СУБД, серверная часть, мобильное приложение-трекер, WEB-сайт для отображения данных конечному пользователю. В данном контексте, под информационной системой “Coordinate” подразумевается совокупность всех ее компонент и коммутирующих узлов, соединяющих эти компоненты.

Субъектами данной работы является ее автор, в качестве проектировщика, аналитика и разработчика, а также потенциальные пользователи, использующие ИС “Coordinate” на частных и коммерческих правах.

Под пользователями, использующих ИС “Coordinate” на частных правах подразумеваются физические лица, пользующиеся каждым компонентом ИС для личного пользования и не получающие прибыль за это использование.

Под пользователями, использующих ИС “Coordinate” на коммерческих правах подразумеваются физические и юридические лица, пользующиеся компонентами ИС для коммерческой деятельности, приносящую прибыль за это использование.

Регионом исследования является Московская область Российской Федерации, включая г. Москва.

## Опрос респондентов

Для постановки функциональных задач и выделения областей, нуждающихся в последующей модернизации, а также для анализа соответствия системы потребностям потенциальных пользователей системы был произведен опрос через анкетирование.

В анкете были заданы следующие вопросы:

1. Максимальное время загрузки страницы, при превышении которого пользования пользование сайтом становится некомфортным.
2. 700 мс. (10%)
3. 1 сек. (20%)
4. 1.5 сек. (65%)
5. 3 сек. (5%)
6. Наиболее оптимальное время загрузки страницы.
7. 0-300 мс. (15%)
8. 300-700 мс. (45%)
9. 700-1000 мс. (30%)
10. 1 - 1.5 сек. (10%)
11. Тяжелые страницы должны загружаться сразу, предоставляя полную информацию. Пока данные загружается, контент должен быть недоступен, а на странице должен отображаться состояние загрузки.
12. Согласен. (20%)
13. Не согласен. (80%)
14. Тяжелые страницы должны загружаться постепенно, пользователю должен быть доступен основной контент сразу, а дополнительный должен загружаться по времени загрузки.
15. Согласен. (75 %)
16. Не согласен. (25%)
17. Тяжелые страницы должны быть разделены на основной контент и дополнительный. Основной контент должен загружаться сразу, а дополнительный - по запросу пользователю (например, при наведении мышкой на элемент).
18. Согласен. (65%)
19. Не согласен. (35%)
20. Допустима ли анимация построения графических элементов (графики, диаграммы) на странице?
21. Допустима, если не создает нагрузку на сайт. (55%)
22. Недопустима, т.к. подобные анимации отвлекают внимание. (25%)
23. Анимация построения графических элементов в наше время обязательна. (20%)
24. Наиболее комфортный способ представления числовых данных
25. Таблицы. (30%)
26. Графики/диаграммы/гистограммы. (65%)
27. Выгрузка в выходной файл. (5%)
28. Удобнее, когда сайт представляет собой одностраничное приложение, без перехода по ссылкам
29. Согласен. (55%)
30. Не согласен. (45%)
31. Удобнее, когда информационные сообщения и формы ввода информации размещаются в модальных окнах и могут быть вызваны из любого контекста сайта.
32. Согласен. (80%)
33. Не согласен (20%)
34. Удобнее, когда формы входа размещены на статичных страницах сайта, на которых есть постоянная ссылка.
35. Согласен. (40%)
36. Не согласен. (60%)
37. Удобнее, когда информационные сообщения размещаются в фиксированной зоне шаблона сайта.
38. Согласен. (10%)
39. Не согласен. (90%)
40. Я бы хотел, чтобы сайт полнофункционально работал с отключенным JavaScript.
41. Согласен. (10%)
42. Не согласен. (15%)
43. Мне все равно (я не знаю, что это). (75%)
44. Сайт не должен иметь доступ к куки браузера.
45. Согласен. (10%)
46. Не согласен. (20%)
47. Мне все равно (я не знаю, что это). (70%)
48. Представим, что Вы пользователь системы, которая производит отслеживание за перемещением Вашего автомобиля. В начале поездки Вы ее активируете, а в конце - отключаете. Какую бы информацию о поездке Вы бы хотели получить от системы? (маршрут, длину маршрута, среднюю скорость и т.п.).

Общий ответ: *среднюю скорость, расстояние, время в пути, максимальную скорость, среднее ускорение, время простоя, примерный расход бензина, общая статистика по автомобилю за все время пользования системой.*

1. Если у сервиса есть мобильное предложения, мне удобнее работать с ним, чем заходить с мобильного браузера на сайт.
2. Согласен. (90%)
3. Не согласен. (10%)

Всего было опрошено порядка 150 респондентов на различных онлайн-ресурсах. С каждым вариантом ответа указан процент респондентов, выбравших данный вариант.

На основании ответов, можно судить, что текущее состояние системы не соответствует потребностям пользователей, по следующим критериям:

* Загрузка данных и отображение страниц происходит недостаточно быстро, для корректной работы с сайтом системы.
* Система не предоставляет необходимый объем статистических данных.
* Мобильное приложение системы не может функционально заменить веб-сайт системы

## Анализ востребованности системы.

Системы мониторинга подвижных объектов в настоящее время широко востребованы и используется во многих сферах деятельности. Выделяется два основных направления использования систем мониторинга по типу использования:

* для личного пользования;
* для коммерческого использования;
* для государственного назначения.

Системы мониторинга для личного пользования могут использоваться в решении самых разных задач:

* мониторинг маршрута пешей или велосипедной прогулки;
* мониторинг и анализ автомобильной поездки;
* определение местоположения личного автотранспорта;
* системы сигнализации о начале несанкционированного движения личного автомобиля.

Главное требование для таких систем: доступность конечному пользователю. Программно-аппаратные части системы не должны быть слишком дорогими, канал передачи данных должен быть доступен обычным людям на территории пользования и т.д.

При проектировании ИС “Coordinate” одной из главной целью было доступность пользователю, поэтому система использует обычный мобильный телефон в качестве аппаратной части трекера, системой геопозиционирования является доступный в РФ GPS, а передача данных осуществляется по мобильным сетям 2G/3G/4G.

С системами мониторинга для коммерческого использования все несколько сложнее, так как они могут, а в некоторых случаях и обязаны использовать специальную аппаратуру и среду передачи данных, кроме то объем передаваемой информации заметно выше, а ее конфиденциальность может носить куда более строгий характер. Таким системам способны решать следующие задачи:

* мониторинг логистических сообщений, транспортировки грузов;
* обратная связь с курьером, службой доставки, расчёт предполагаемого времени доставки, оперативное оповещения о сбоях;
* мониторинг и учет автопарка компании;
* поиск ближайшего такси, расчет времени подачи автомобиля;
* гарантировать сохранность автомобиля при аренде, в т.ч. и при кратковременной.

Система “Coordinate” и ее технологии могут быть взяты за основу при реализации такой системы, хоть и изначально, при проектировании ИС “Coordinate”, цели соответствия требованиям, предъявляемым к коммерческим системам не было. Тем не менее, система может быть модернизирована таким образом, чтобы соответствовать этим требованиям.

Системы мониторинга государственного назначения оказывают спектр услуг в государственной сфере и способны решать следующие задачи:

* оптимизация работы и минимизация расходов спецтранспорта;
* слежение за местоположением транспорта специального назначения (конвои, инкассации и т.п.)
* повышение функциональности работы экстренных служб;
* сбор статистической информации о ситуации на дорогах.

Системы государственного назначения сложны при вводе в эксплуатацию, так как требуют сертификацию ПО и аппаратуры и разрешение на внедрение.

Система “Coordinate” может быть сертифицирована, но это затребует переработки архитектуры системы, что может занять довольно большой промежуток времени.

Таким образом, можно сделать вывод, что система “Coordinate” может быть использована для личного и коммерческого использования, во сферах где к ПО и аппаратной части не предъявляют особых требований. Для использования в структурах государственного назначения система “Coordinate” существующем виде не считается подходящей.

## Основные технологии построения систем мониторинга

Система мониторинга подвижных объектов – система, построенная на основе средств спутниковой навигации, оборудования и сотовой связи. Такие системы, в настоящее время, используются для решения задач транспортной логистики, таких как управление перевозками, автопарком. Принцип работы заключается в отслеживании и анализе пространственных и временных координат транспортного средства.

Существует 2 основных варианта мониторинга: online – когда информация передается, по мере обновления координат и offline – информация накапливается в памяти спутникового передатчика (трекера) и передается в некоторой контрольной точке (диспетчерский пункт, и т.п).

Основным недостатком online-мониторинга является возможность потери данных, при обрыве соединения. Этот недостаток можно устранить, храня данные в памяти трекера, до ответа от сервера об успешном сохранении координат. Но такой подход значительно усложняет логику взаимодействия трекера с сервером, и, кроме того, допустимый объем для хранения данных в таких трекерах не велик, что может привести к заполнению модуля памяти и дальнейшей потере данных. Более того, усложнение логики и увеличение объема хранение данных влечет за собой усложнение аппаратной конструкции трекера, что сказывается на стоимости устройства, а постоянное соединение с сервером требует более мощные и дорогие модули связи.

Основным недостатком offline-мониторинга является неактуальность полученных данных. Данные на сервер приходят пулами, за, достаточно большой, промежуток времени. Такой подход совершенно неприемлем, если нужно производить слежение в реальном времени. Зато, offline-мониторинг позволяет использовать сервис даже в тех местах, где покрытие мобильной сети нестабильно, и имеет место возможность потери сигнала. На транспортном средстве устанавливается мобильный модуль (трекер), включающий в себя: спутниковый модуль-приемник, позволяющий определять географические координаты со спутниковых сигналов, модуль хранения, модуль передачи данных. Кроме того, к трекеру могут быть подключены различные внешние датчики, предоставляющие информацию о внешних факторах среды (температура, обороты двигателя и т.п.).

## Описание существующей системы “Coordinate"

Система Coordinate была спроектирована и реализована таким образом, что совмещает в себе положительные стороны обеих технологий построения систем мониторинга: пока присутствует соединение с сервером, данные передаются в онлайн режиме, но как только соединение с сервером прерывается, новые данные начинают накапливаться в буфере трекера, что позволило достичь актуальность данных в режиме реального времени, и не допустить потерю данных в случае потери соединения с сервером.

Архитектура системы не обязывает в качестве GPS-трекера использовать мобильный телефон, так как коммуникация с серверной компонентой системы происходит в формате JSON обычным HTTP запросом с поддержкой шифрования SSL. Таким образом, в качестве GPS-трекера может служить любое устройство, способное получать GPS-координаты и передавать их в определенном формате на сервер HTTP-запросом, что дает возможность использовать в качестве GPS-трекера встраиваемые системы.

Вариант использования мобильного телефона в качестве трекера позволил не использовать специальное оборудование, которое может быть недоступно конечному пользователю, а также полностью решает проблему хранения несинхронизированных данных в локальном буфере, так как на текущий момент даже телефоны низкой ценовой категории способны хранить относительно большой объем информации.

Реализованная система представляет собой программный стек, все элементы которого способны взаимодействовать между собой. Различается, как минимум, 4 программных уровня (рис. 3.1):

**СУБД** (PostgreSQL) – на этом уровне происходит хранение пользовательских данных. Представляет из себя развернутый сервер PostgreSQL 9.4.

**Backend** (Rails 4.2) – на этом уровне реализуется основная логика сервиса, предоставляется санкционированный доступ к пользовательским данных, а также происходит сбор необходимой информации.

**Frontend** (Web-сайт, мобильное приложение и прочие потенциальные пользовательские приложения) – уровень, непосредственно используемый пользователем для предоставления и отображения данных.

**Tracker Application** (Мобильное-приложение-трекер) – приложение под ОС Android, выполняющее сбор и передачу GPS координат на сервер, а так же серверный программный интерфейс (API) между ними.

Серверное ПО представляет собой Web-приложение на языке ruby 2.0.1 с использованием программного комплекса (фреймворка) Ruby on Rails 4.2. В качестве HTTP-сервера используется сервер приложений Apache 2.2 с используемым модулем Passenger для развертывания Ruby on Rails - приложений. Серверное ПО размещено на личном виртуальном сервере (VPS) под управлением CentOS 6.5. с публичным доступом по сети Интернет.

Пользовательское ПО представляет собой динамический Web-сайт, с использованием современных технологий, таких как JQuery v1.11.2, Bootstrap v3.3.2, HTML5, SCSS. Сайт корректно отображается во всех современных браузерах, за счет чего достигается полная кроссплатформенность клиентского приложения.



Рисунок . Функциональная схема системы Coordinate.

Реализованное ПО для трекера представляет собой мобильное приложение для популярной мобильной ОС Android (>2.0), тем самым достигается требование доступности пользователю. Использование актуальной ОС Android, также, позволяет использовать приложение-трекер как в смартфоне пользователя, так и в интегрированном в автомобиль устройстве, таком как GPS-навигатор, портативный автомобильные ПК или видеорегистратор. Кроме того, для пользователя открыта спецификация формата коммуникации с сервером, что дает возможность реализовывать свое ПО для неподдерживаемых архитектур.

Коммуникация между уровнями защищена SSL шифрованием, и использует протокол HTTPS. SSL шифрование осуществляет алгоритм SHA-256 (по открытому ключу(RSA)), сертификат выдан и подписан компанией StartCom Ltd. Приложение на Ruby on Rails обеспечивает корректное получение, обработку, хранение, санкционированный доступ к данным, вариацию и верификацию данных, и аутентификацию пользователей.

# Исследование системы “Coordinate”

## Методология исследования

Методы исследования позволяют наиболее эффективно достичь целей исследовательской работы. Основным методом исследования данной работы является анализ – исследование объекта исследования с учетом его характеристик, индивидуальных свойств и признаков.

Для анализа необходимо заполнить базу объектов исследования и их моделей. База может быть собрана исходя из реальных данных, полученных в ходе нормальной эксплуатации системы, экспериментальных полученных в ходе эксперимента, и формальных - вручную сгенерированных моделей данных, максимально похожих на реальные данные.

Анализ проводится как над самим объектом исследования, так и над его моделями. Анализ может быть проведён на основе измерительных расчётов отдельных свойств, и числовых показателей (рис. 4.1), так и на основе сравнении конкретных свойств объектов исследования (или их моделей (рис. 4.2)). Отдельные свойства могут быть синтезированы и рассмотрены при анализе, как одно целое (рис. 4.3).

Результаты сравнительного анализа по каждому из набору свойств объекта могут быть обобщены, и приведены к конечному заключению. Однако, индуктивный метод исследования позволяет исходя из общего заключения, провести анализ и частных решений, что дополнит полноту исследования.

Объект

*А*

Измерения объекта *F(A)*

Рисунок .. Измерение свойств объекта А

Измерение модели объекта

*F (А′)*

Измерение объекта

*F (А)*

Сравнительный анализ *C(F(A), F(А′))*

Рисунок .. Сравнение свойств объекта и модели

Измерение

синтеза свойств

модели

объекта

*F (S(А′))*

Измерение

синтеза свойств

объекта

*F (S(А))*

Сравнительный анализ *C(F(S(A)), F(S(А)′))*

Рисунок .. Сравнение синтеза свойств объекта и модели

## Анализ хранения геоданных в системе “Coordinate”

Данные, полученные от GPS-трекера хранятся в БД под управлением СУБД PostgreSQL 9.3. Схема структуры таблиц изображена на рис. 4.4. Таблицы и отношения между ними создаются фреймворком Ruby on Rails благодаря механизму ORM - ActiveRecord. Данный подход позволяет абстрагироваться от типа и версии СУБД, но имеет ряд существенных недостатков:

* все связи в между таблицами построены на уровне языка, и совершенно не затрагивают БД, превращая ее в набор несвязанных между собой таблиц;
* нарушается принцип единой ответственности – доменный объект должен иметь только одну зону ответственности, то есть содержать только свою бизнес-логику. Вызывая его для сохраненных в БД, данных, происходит размытие зоны ответственности, что влечет за собой увеличение сложности объекта и усложнение поддержки и тестирования проекта;
* описание объекта значительно усложняется, если свойства объекта не соответствуют полям таблицы;
* низкая эффективность формирования сложных запросов: составные запросы формируется не самым эффективным образом, а некоторые запросы и вовсе невозможно сформировать. Приходится использовать SQL-команды вместе с синтаксисом ORM, что приводит к усложнению поддержки кода и заметно понижает его понимание.

Эксплуатация показала, что для сервиса "Coordinate" наиболее важными проблемами в части хранения данных являются:

* неоптимальная структура межтабличных связей;
* неэффективные запросы, сгенерированные в результате работы ORM.



Рисунок .. Схема БД сервиса “Coordinate”

На схеме, изображенной на рис. 4.4 можно заметить межтабличные отношения четырех видов:

1. один к одному (1 → 1). Простейшее отношения между таблицами через внешний ключ (FK), позволяющее связать две сущности. используется в тех случаях, когда одна сущность может иметь не более одного отношения с другой, например, каждому пользователю свойственен только один экземпляр объекта профиля, с информацией о нем;
2. один ко многим (1 → ∞). Используется, когда одна сущность, может иметь сколь угодно много отношений с другой сущностью. Такой тип отношения используется, к примеру, для автомобилей, которые могут иметь сколь угодно много треков за весь период пользования сервисом;
3. многие ко многим (∞ → ∞). Это связь, при которой множественным записям из одной таблицы могут соответствовать множественные записи из другой. Достигается за счет использование составной таблицы. Не используется в системе Coordinate;
4. наследование Rails. Нереляционное отношение, позволяющие хранить сущности с одинаковым набором полей в одной таблице. Реализовано на уровне фреймворка Rails. Используется для того, чтобы разделить геолокацию трека от прочих.

Связь один ко многим

Связь один к одному

Связь многие ко многим

Наследование Rails

Условные обозначения

Реляционные отношения в БД способны уменьшить объем таблиц и упорядочить данные в определенную структуру, а также позволяют реализовать паттерн “ленивой” загрузки данных (Lazy load). Паттерн Lazy load подразумевает отказ от загрузки дополнительных данных, когда в этом нет необходимости. Вместо этого ставится маркер о том, что данные не загружены и их надо загрузить в случае, если они понадобятся.

Существует четыре основных варианта ленивой загрузки:

1. lazy Initialization (ленивая инициализация) использует специальный маркер (обычно null), чтобы пометить поле, как не загруженное. При каждом обращении к полю проверяется значение маркера и, если значение поля не загружено - оно загружается;
2. virtual Proxy (виртуальный прокси) - объект с таким же интерфейсом, как и настоящий объект. При первом обращении к методу объекта, виртуальный прокси загружает настоящий объект и перенаправляет выполнение;
3. value Holder (контейнер значения) - объект с методом getValue. Клиент вызывает метод getValue, чтобы получить реальный объект. getValue вызывает загрузку;
4. ghost (призрак) - объект без каких-либо данных. При первом обращении к его методу, призрак загружает все данные сразу.

ActiveRecord гарантирует использование ленивой загрузки с через механизм виртуального прокси. Поэтому, зависимые данные не загружаются при загрузки основного объекта до тех пор, пока не будет явного обращения к ним, что значительно уменьшает нагрузку на БД и повышает производительность.

Но межтабличные связи требуют и хранение дополнительных данных: внешних ключей и индексов, что значительно увеличивает размер хранимых данных. В приложении А.1 приведен запрос, позволяющий узнать общую информацию о размерах таблиц БД. Результаты выполнения этого запроса в 2015 и в 2017 годах представлены в таблице 4.1. Данные были собраны в результате нормальной работы сервиса, начиная с 01.03.2015 и актуальны для одного реального пользователя, при частоте пользования сервисом не реже чем раз в неделю.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Июнь 2015 | | Март 2017 | |
| Общий размер | Индексы, VM, FSM | Общий размер | Индексы, VM, FSM |
| locations | 16 MB | 3800 kB | 46 MB | 12 MB |
| logged\_exceptions | 1272 kB | 1024 kB | 2528 kB | 2280 kB |
| tracks | 64 kB | 48 kB | 96 kB | 56 kB |
| active\_admin\_comments | 40 kB | 40 kB | 80 kB | 72 kB |
| simple\_captcha\_data | 144 kB | 112 kB | 72 kB | 56 kB |
| cars | 64 kB | 56 kB | 64 kB | 56 kB |
| users | 64 kB | 56 kB | 64 kB | 56 kB |
| profiles | 48 kB | 40 kB | 48 kB | 40 kB |
| reset\_passwords | 24 kB | 24 kB | 48 kB | 40 kB |
| api\_tokens | 48 kB | 40 kB | 48 kB | 40 kB |
| users\_mails | 32 kB | 24 kB | 32 kB | 24 kB |
| images | 32 kB | 24 kB | 32 kB | 24 kB |
| schema\_migrations | 32 kB | 24 kB | 32 kB | 24 kB |
| verification\_users | 32 kB | 24 kB | 32 kB | 24 kB |

Таблица .. Размеры таблиц БД системы “Coordinate”

Под общим размером таблицы подразумевается реальный объем данных, занимаемый отношением на дисковом пространстве. Это значение включает в себя как реальные данные, так и системные, используемы для построения и представления отношения.

Системные данные включают в себя:

* **Индексы** – объекты, создаваемые с целью повышения навигации по данным и, как следствие – повышение производительности поиска данных. Таблицы в БД могут иметь большое количество строк, и их поиск по указанному критерию путем последовательного просмотра таблицы может занимать очень много времени. Индекс формируется из значения одного из нескольких столбцов таблицы и указателей на соответствующие строки таблицы, таким образом позволяя выполнять поиск строк, удовлетворяющих критерии поиска. В PostgreSQL есть несколько типов индексов, но основное распространение получили индексы со структурами B-дерева и хеш-функции.
* **VM** (Visibility Map) – карта видимости. Каждое табличное отношение в PostgreSQL имеет карту видимости для отслеживания страниц, содержащих только кортежи, видимые для всех активных транзакций, а также VM содержит информацию о страницах, в которых содержатся только активные кортежи. Карта хранит по 2 бита на страницу таблицы. Первый бит, если он установлен - показывает, что вся страница видна всем активным транзакциям, и не содержит кортежей, нуждающихся в очистке. Установленный второй бит указывает на то, что на странице нет активных кортеже, и все они заморожен, что означает, что процесс очистки, для предотвращения зацикливания, не должен больше посещать данную страницу.
* **FSM** (Free Space Map) – карта свободного пространства. Любое табличное и индексное отношение, за исключение хеш-индексов, в PostgreSQL имеет карту свободного пространства, для отслеживания доступного места. Она хранится рядом с данными главного отношения в отдельном слое, имя которого формируется именем файлового узла отношения с суффиксом \_fsm. Карта свободного пространства представляет собой дерево страниц FSM. Страницы FSM хранят информацию о свободном пространстве, доступном на каждой странице таблицы (индекса), используя 1 байт для представления каждой такой таблицы. Внутри каждой FSM таблицы имеется бинарное дерево, хранящееся в массиве, где на каждый узел дерева выделяется 1 байт.

Исходя из полученных данных, можно выделить таблицу locations, которая применяется для хранения координат в различные промежутки времени трека автомобиля, так как при почти неизменном количестве родительских сущностей (пользователи, автомобили), размер хранимых в ней данных имеет значительный прирост.

Физически, в БД нет связи между таблицами locations (геолокации) и users (пользователи), но эта связь достигается в процессе нормальной работы бизнес-логики сервиса.

Добавляя в профиле автомобили, создается связь между таблицами users и cars. Затем, при получении геолокаций от GPS-трекера, создается сущность в таблице tracks (маршруты), привязанная к конкретному автомобилю. Все последующие геолокации, в контексте этого маршрута создаются с привязкой к этому маршруту (логика разделения маршрутов и создания новых рассматривается далее в работе).

Рисунок 4.5. Организации связи “Пользователь – Геолокация”

На рис. 4.5 изображена связь между пользователем сервиса и геолокацией. Очевидно, что каждая связь “один ко многим” вырождается в линейную зависимость по количеству записей между родительской таблице и дочерней. Соответственно, каждое сочетание из нескольких таких связей будет вырождаться в экспоненциальную зависимость дочерней таблицы от корневой. Это особенно заметно на таблице 4.2, в которой указано реальное число строк в таблицах для одного пользователя за период использования в два года.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблицы | users | cars | tracks | locations |
| Число записей | 1 | 3 | 470 | 117046 |

Таблица .. Зависимость числа записей в таблицах

Рассчитаем примерное число записей в таблице locations при большом количестве пользователей, учитывая, что каждый пользователь не использует более одного автомобиля. Для получения среднего числа маршрутов на автомобиль, воспользуемся запросом из приложения А.2, а для получения среднего числа на геолокационных точек на маршрут, воспользуемся запросом из приложения А.3. Результаты вычислений представлены в таблице 4.3.

На основе полученных были построены гистограмма (рис. 4.6) и график зависимости (рис. 4.7). Данные графики наглядно демонстрирует экспоненциальный рост данных в таблице locations при линейном росте в таблице users.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблицы | users | | Cars | | tracks | | locations | |
| N | Ln(N) | N | Ln(N) | N | Ln(N) | N | Ln(N) |
| Число записей | 1 | 0,000 | 1 | 0,000 | 235 | 5,460 | 58515 | 10,977 |
| 100 | 4,605 | 100 | 4,605 | 23500 | 10,065 | 5851500 | 15,582 |
| 1000 | 6,908 | 1000 | 6,908 | 235000 | 12,367 | 58515000 | 17,885 |
| 10000 | 9,210 | 10000 | 9,210 | 2350000 | 14,670 | 585150000 | 20,187 |
| 50000 | 10,820 | 50000 | 10,820 | 11750000 | 16,279 | 2925750000 | 21,797 |
| 100000 | 11,513 | 100000 | 11,513 | 23500000 | 16,973 | 5851500000 | 22,490 |
| 250000 | 12,429 | 250000 | 12,429 | 58750000 | 17,889 | 14628750000 | 23,406 |

Таблица .. Результаты расчета зависимости роста записей

Рисунок .. Гистограмма, изображающая зависимость роста данных

Рисунок .. Экспоненциальная зависимость при росте данных

Для получения более подробной информации о размере таблицы, воспользуемся запросом из приложения А.4. Результат применения данного запроса к таблице locations представлен в таблице 4.4.

Таблица 4.4 Информация о таблице “locations”

Следующим шагом будет анализ индексов таблицы locations. Их объем составляет 12МБ при 34МБ чистых данных, что, считается, не так уж много. Для просмотра информации о индексах, воспользуемся запросом из приложения А.5.

Результат отображен в таблице 4.5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Индекс | Размер индекса | Уникальность | Число использований | Прочитано кортежей, с помощью индекса | Выбрано кортежей, с помощью индекса |
| track\_id\_lat\_long\_time\_index | 8920 kB | Y | 1560 | 404460 | 376250 |
| locations\_pkey | 4088 kB | Y | 0 | 0 | 0 |
| locations\_time\_idx | 2592 kB | N | 0 | 0 | 0 |

Таблица .. Информация об индексах таблицы “locations”

Исходя из данных, можно сделать вывод, что несмотря на достаточный размер таблицы, размеры индексов не превышают допустимых. Работоспособность можно проверить на плане построения запроса, сгенерированным при вычислении одного из нескольких статистических запросов системы “Coordinate”.

PostgreSQL позволяет без труда построить план построения, для этого необходимо добавить лишь директиву EXPLAIN (ANALYZE) перед запросом. В приложениях А.6 и А.7 приведен запрос, позволяющий получить план построения вложенного запроса и его результат. Вложенный запрос, вызываемый итерационно для каждого автомобиля пользователя, возвращает расстояние каждого маршрута автомобиля за весь период пользования сервисом. Затем, уже в бекенде, эти данные агрегируются и получается общее пройденное расстояние автомобилем.

План подзапроса в приложении А.6 представляет собой дерево вложенных операций, применяемых СУБД для построения запроса. Обходить это дерево следует из глубины: в таком случае сохранится хронология действий СУБД.

Первой операцией является ***Seq Scan on tracks*** – фильтрация таблицы “tracks”, используя последовательное сканирование. Последовательное сканирование очень дорогая операция, т.к. сводится к последовательному перебору таблицы. Значение в скобках *(cost=0.00..10.88 rows=403 width=4)* означают абстрактную стоимость поиска первого и последнего элемента, число обработанных строк и столбцов. Параметр *Filter* говорит о том, что данная операция является фильтрацией по полю *car\_id*. Так как фильтрация с помощью последовательного сканирования очень трудоемкий процесс, целесообразно в таком случае использовать индекс, по полю *car\_id.* Тем не менее, временные затраты на эту операцию – всего 0.5мс.

Второй операцией является ***Hash*** – подсчет хеш-функции для каждой записи, получившейся в результате первой операции. Данная операция занимает 32 кБ. ОЗУ, и затрачивает около 1мс. Общего времени запроса.

Третья операция ***Seq Scan on locations*** выполняет фильтрацию таблицы *locations* по полю *state* опять же, без использования индекса. Данная операция используется для реализации наследования таблиц ORM ActiveRecord. Затраты на эту операцию составляют ~220 мс., что составляет чуть более половины общего времени запроса.

Следующая операция ***Hash Join***выполняет соединение результатов третьей и второй операции, по условию *locations.track\_id = tracks.id.* Временные затраты составляют ~150мс.

Последняя, пятая операция ***HashAggregate*** выполняет группировку данных. Временные затраты - ~80мс.

Можно выделить 2 основные проблемы данного запроса:

* отсутствие индексов при фильтрации, и как следствие, использование последовательного перебора записей;
* трудоемкий запрос, использующий ORM ActiveRecord для реализации наследования track\_locations от locations.

Причем, в приложении А.6 представлен лишь один запрос из многих, посылаемых асинхронно при получении статистики пользователем.

Первая проблема решается автоматически, добавлением индексов на соответствующие поля.

Решением второй проблемы послужить пересмотр концепций хранения данных с последующей переработкой бизнес-логики, а именно – отказ от наследования моделей Active Record между классами Locations (базовый класс геолокационных данных) и TrackLocations (классы геоданных маршрута). Дело в том, что данное наследование достигается за счет использования общей таблицы locations и введения в нее дополнительного столбца state с текстовым классификатором – именем класса, к которому принадлежи конкретная строка таблицы. Таким образом, для того что бы сделать выборку из всех точек маршрута автомобиля, необходимо задавать условий выборки *“WHERE ″locations.state″ = ″TrackLocations″”*, что само по себе не является быстрым и оптимальным решением.

Так как, в настоящее геоданные, не привязанные к маршруту не предусмотрены, и их добавление в скором времени не планируется, было принято решние отказаться от наследования совсем. В последствии, если потребуется вводить немаршрутные геоданные (точки статичных объектов), то возможна реализация наследования, уже на уровне PostgreSQL: в таком случае, будут созданы несколько физических таблиц, с разным набором индексов и общей базовой структурой.

Для достижения удаления наследования была изменена конфигурация моделей *TrackLocations* и *Locations*, удален столбец *″locations.state″*,что дало существенный прирост производительности БД.

При применении CRUD операций к объектам БД приводит к появлению недействующим кортежам (см. таблицу 4.4), что замедляет работу индексов. Для уничтожения недействующих кортежей и перерасчета индексов в PostgreSQL применяется команда *″VACUUM FULL″*.

Результат выполнения запроса из приложения А.4 после оптимизации таблицы locations приведен на таблице 4.6.

Запрос с из приложения А.6 после оптимизации таблицы locations примет вид запроса, представленного в приложении А.8, а результат его выполнения приведен в приложении А.9.

Таким образом, анализируя результаты, полученные после оптимизации таблицы “locations” можно говорить о росте производительности в более чем 2 раза, и уменьшении объема занимаемых данных на почти на 3% в запросах, использующих данную таблицу при формировании выборок данных.

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | В байтах | В удобночитаемом виде | Байт на строку |
| Общий размер отношения | 34594816 | 33 MB | 295 |
| Карта видимости | 0 | 0 bytes | 0 |
| Размер свободного пространства | 0 | 0 bytes | 0 |
| Размер таблицы, включая TOAST | 34603008 | 33 MB | 295 |
| Размер индексов | 11182080 | 11 MB | 95 |
| Общий размер, включая TOAST и индексы | 45785088 | 44 MB | 390 |
| Кол-во используемых строк в текстовом представлении | 33598218 | 32 MB | 286 |
| Число строк | 117202 |  | |
| Число действующих кортежей | 117202 |
| Число недействующих кортежей | 0 |

Таблица . Информация о таблице “locations”

## Анализ обработки геоданных в системе “Coordinate”

Полученные по протоколу HTTPS данные на сервере обрабатываются приложением Ruby On Rails, в котором и размещается бизнес-слой системы. Приложение Ruby On Rails реализует паттерн MVC (Model View Controller – Модель Представление Контроллер) и четко разделяет контексты декларации объектов системы, описания их поведения и представления.

Декларирование объектов состоит из описания набора полей свойств, конфигурации хранения объекта. Описанный объект называется моделью (Model).

Модель не только описывает набор вариантов использования объекта, но и содержит информацию о том, как привести БД к виду, необходимому для хранения данных, и как эти действия откатить. За это отвечает встроенный в ORM Active Record механизм миграций. Основная конфигурация БД происходит именно через него.

Представления описывает интерфейсы модели. Под интерфейсом подразумевается не только HTML-страница WEB-сайта, но и представление модели в виде форматах сериализации, таких как XML, JSON, а также в виде электронных документов, изображений и пр. Сервис Coordinate использует только HTML представление для WEB-сайта и JSON для API.

Контроллер позволяет обрабатывать входящие HTTP(HTTPS) запросы, манипулируя моделями и их методами.

Связка модель-представления-контроллер определяет понятие ресурса системы. Серверная часть системы “Coordinate” реализует архитектуру REST и является RESTful приложением. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

Маршрутизация в системе происходит по белому списку разрешенных URL, который конфигурируется в файле ″$COORDINATE\_APP/config/routes.rb″. Каждый запрос обрабатывается в соответствии с форматом маршрута, описанным в конфигурационном файле и извлеченные параметры запроса передаются на вход соответствующего метода контроллера.

В таблице 4.7 указаны контроллеры, используемые WEB-сервисом “Coordinate” для обработки входящих запросов. Для выявления наиболее нагруженных контроллеров системы, был проведен анализ журналов, результат которого приведен в приложении Б.

|  |  |  |  |
| --- | --- | --- | --- |
| Пространство имен | Контроллер | Формат | Назначение |
| admin | dashboard | HTML | Главная страница панели администратора. |
| admin | users | HTML | Страница администрирования пользователей. |
| admin | exceptions | HTML | Страница администрирования исключений и ошибок сервиса. |
| admin | cars | HTML | Страница администрирования транспортных средств пользователей. |
| admin | tickets | HTML | Страница администрирования пользовательских обращений. |
| admin | comments | HTML | Страница администрирования комментариев к ресурсам системы. |
| - | static\_pages | HTML | Статичные страницы сайта: "О системе", "FAQ", и др. |
| - | users | HTML, JS, JSON | Профиль пользователя, панель настроек и пр. |
| - | sessions | HTML | Управление HTTP(HTTPS) сессией, при пользовании WEB-сайтом. |
| - | cars | HTML, JS, JSON | Страницы WEB-сайта, связанные с управлением автотранспортом пользователями. |
| - | users\_mails | HTML | Управление email-рассылкой пользователей. |
| maps | cars | JSON | Предоставление информации об автомобилях для работы с картой. |
| maps | tracks | JSON, JS | Предоставление информации о маршрутах для работы с картой. |
| maps | locations | JSON | Предоставление информации о геолокациях для работы с картой. |
| api/v1/ | sessions | JSON | Управление сессиями для устройств, использующих API (мобильное приложение-трекер) |
| api/v1/ | geodata | JSON | Получение и обработка геолокационных данных. |

Таблица .. Перечень контроллеров WEB-сервиса “Coordinate”

Для анализа быстродействия серверной компоненты системы “Coordinate” был проведен комплекс мер по обработке журналов системы за отдельно взятый промежуток времени работоспособности системы, длиной в неделю. В течении данного времени, все запросы к сервису системы записывались в отдельный журнал, результат анализа которого приведен в приложении Б. Исходя из этого анализа, можно выделить наиболее ресурсоемкие методы:

1. ″Api::V1::GeodataController#recive”. Данный метод используется для обработки входящих геоданных из трекера. Агрегированные данные по данному контроллеру приведены в таблице 4.8.
2. ″UsersController#charts\_controller″. Используется для получения информации о треках пользователя. Общие данные о нем приведены в таблице 4.9.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| recive (GeodataController) | Общее время | | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Отработка действия контроллера | 3m42s | 100% | 231ms | 111ms | 5.80s | 132ms-564ms |
| Обработка запросов к БД | 18.49s | 8% | 19ms | 5ms | 270ms | 8ms-48ms |
| Построение представления | 2.35s | 1% | 2ms | 0ms | 1.10s | 0ms-5ms |

Таблица . Информация о методе ″Api::V1::GeodataController#recive″ по результатам анализа журнала.

Действие ″Api::V1::GeodataController#recive” получает геоданные с трекеров пользователей и выполняет их первичную обработку. Алгоритм его работы приведен в приложении В. По таблице 4.8. видно, что время на выполнение запросов к БД и на построения представления в сумме составляют порядка 10% от общего времени выполнения действия контроллера, когда как на выполнения алгоритмических операций в методе контроллера приходится остальные 90%. Таким образом, основная оптимизация данного контроллера заключается в оптимизации бизнес-логики алгоритма данного метода.

Оптимизация бизнес-логики действия ″Api::V1::GeodataController#recive” сводится к исправлению двух ключевых факторов, влияющих на производительность алгоритма:

* улучшение фильтрации принимаемых параметров, с целью отсеивания некорректных данных;
* изменения формата хранения данных в БД, с целью уменьшения объема хранимых данных и трудоемкости поиска.
* оптимизация алгоритма поиска подходящего маршрута find\_track.

Наиболее частым примеров некорректных данных являются точки маршрута, скорость которых равна нулю. Такой вариант возможен, если автомобиль располагается вне зоны устойчивого GPS сигнала без движение (например, располагается в гараже или на крытой парковке). В таком случае, координаты расположения автомобиля будут беспорядочно колебаться, в диапазоне параметра accuracy. Очевидно, что тем ниже значение данного параметра, тем меньше разброс геоданных, а, следовательно, выше их точность. Для GPS допускается погрешность в 15 метров.

К сожалению, опираться на один лишь параметр accuracy невозможно, так как его значения зависит от множества индивидуальных факторов, таких как погодные условия или технические характеристики устройства сбора данных.

Ситуация с параметром speed (скорость) совершенно обратная: любые геоданные, содержащие нулевое значение данного параметра означают, что транспортное средство пользователя находится в состоянии покоя без движения, и случай, когда на сервер приходят данные с различными координатами, но нулевой скорость является неккоректными, с точки зрения системы, так как отображает не реальное перемещение автомобиля, а разброс координат, получаемый в результате погрешности измерения (рис. 4.8).

Для оценки весомости данного фактора, были проанализированы данные, полученные во временном диапазоне, когда проводился анализ, результаты которого приведены в приложении Б. Анализ этих данных показал, что из 975 полученных геоданных 59 из них (~6%) является некорректными. Всего же, за все время эксплуатации системы, число таких данных составляет 8389 (~7%).

Таким образом, для предотвращения данной ситуации, необходимо дополнить метод ″Api::V1::GeodataController#recive” проверкой на нулевой параметр speed и прерывать обработку таких геоданных в системе. Такую же проверку целесообразно добавить в мобильное приложение-трекер для предотвращения передачи заведомо неверных данных.



Рисунок .. Отображение некорректных геоданных с нулевой скоростью на карте системы.

Следующим шагом является оптимизация формата хранения данных. До модернизации, приведенной в данной работе, такие параметры, как accuracy, speed, distance хранились в виде вещественных чисел. Так система “Coordinate” использует метрическую систему измерения, целесообразно округление данных параметров до целых, с целью уменьшения объема, хранимых данных и дополнительных преобразований.

Завершающим этапом, оптимизации метода получения и первичной обработки геоданных ″Api::V1::GeodataController#recive” является оптимизация алгоритма поиска целевого трека (метод ″Api::V1::GeodataController#find\_track”), с которым необходимо связать полученные геоданные.

Алгоритм поиска целевого трека представлен в Г.1. Суть его работы заключается в подборе наиболее подходящего трека для полученных геоданных, причем последовательность геоданных может быть хронологически нарушена, так как мобильное приложение-трекер способно локально хранить полученные геоданные, в случае отсутствия соединения с системой. В таком случае, данные будут обработаны алгоритмом параллельно с данными, которые могут относиться к другим трекам. Помимо этого, алгоритм гарантирует выполнение следующих условий:

* Если временной интервал, между последней геолокацией и полученной геолокацией превышает интервал в 15 минут, будет создан новый трек для полученных геолокаций.
* Если временной интервал между двумя треками становится меньше 15 минут, треки будут объединены в один.
* Если полученные геоданные входят в временной интервал между началом и концом существующего трека, они будут добавлены в него.

Существующая реализация алгоритма поиска целевого трека допускает ряд ошибочных ситуаций, поведение которых критично для системы и требует немедленного исправления:

* Создание нового трека (или объединение треков) и сохранение новой геолокации происходит в разных транзакциях, что может привести к мусорным данным в БД.
* При объединении треков, не происходит перерасчет дистанции нового трека.
* В случае асинхронной загрузки геолокаций, возможна ситуация, когда геолокация попадает в два разных трека, и последующее объединение этих треков, становится невозможным, так как нарушается уникальность индекса.

Обновленная реализация алгоритма поиска трека опирается не на время начала/окончание трека, а на конкретную геолокацию, сохраненную в бд, находящейся в интервале *X-15 мин. … X+15 мин.*, где X – время полученной геолокации. Полученные геолокации, группируются по трекам, и если

* Таких треков нет (геолокации в указанном интервале отсутствуют) – необходимо создать новый трек.
* Такой трек единственный – присвоить его новой геолокации.
* Таких треков больше одного – присвоить геолокации трек, полученный в результате объединения этих треков.

Отказавшись от привязки ко времени начала/конца трека были не только решены проблемы с корректным поиском трека, но и ускорена работа, за счет полного переноса логики из серверной части в запрос БД, а также, появилась возможность создания нового трека и сохранения геолокации в рамках одной транзакции.

Обновленная версия алгоритма поиска целевого трека охватывает случаи расположения новой геолокации, относительно существующих треков, представленные на рисунке 4.9. Обновленный алгоритм приведен в приложении Г.2.

Алгоритм объединения двух треков представлен в приложении Г.3. При необходимости объединения, все геолокации меньшего трека Y (трека, с наименьшим число геолокаций) переносятся в больший трек X, затем, обновляются временные метки трека X (дата начала, дата конца) и выполняется перерасчет дистанции. Заключительным этапом работы алгоритма становится уничтожение трека Y и возврат трека X. Для объединения более чем двух треков, алгоритм может быть вызван из рекурсивной функции обхода списка треков.

Новая геолокация

X

< 15 мин.

> 15 мин.

б) Найден единственный трек X, в интервалах 15 минут – добавление в этот трек.

Новая геолокация

X

> 15 мин.

< 15 мин.

> 15 мин.

> 15 мин.

Новая геолокация

а) Трек, в интервалах 15 минут не найден – создание нового трека.

Новая геолокация

X

< 15 мин.

Y

< 15 мин.

в) Найдено несколько треков (допускается более двух). Объединение найденных треков и присвоение его новой геолокации.

Рисунок . Возможные варианты расположения полученных геолокаций, относительно существующих в системе треков.

Данные о действии к методу ″*UsersController*#*charts\_controller*″ приведены в таблице 4.9. В отличие от метода ″Api::V1::GeodataController#recive” здесь наиболее трудоемкими операциями являются обращения к БД. При этом, данный метод используется для предоставлении разных данных о треках текущего пользователя, в зависимости от передаваемого параметра. Более корректным и оптимальным решением будет разделение данного метода на более мелкие функции, результатом которых будет конкретная информация по треку, в зависимости от запроса, и перенос этих функций в отдельный контроллер.

Запросы, используемы данным методом были оптимизированы в ходе данной работы на этапе раздела 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| charts\_controller (UsersController) | Общее время | | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Отработка действия контроллера | 1m14s | 100,00% | 1.27s | 15ms | 5.20s | 26ms-4.74s |
| Обработка запросов к БД | 34.55s | 46,69% | 639ms | 20ms | 4.01s | 28ms-3.44s |
| Построение представления | 548ms | 0,74% | 10ms | 0ms | 41ms | 0ms-30ms |

Таблица .. Информация о методе ″*UsersController*#*charts\_controller*″ по результатам анализа журнала.

Вынос метода в отдельный контроллер, с методами для каждого типа запроса, дал возможность посылать данные запросы асинхронно, а также с использование “ленивой” загрузки (отправка запроса лишь в момент требования конкретных пользователю, а не при старте загрузки страницы.)

В результате вышеперечисленных мероприятий, нацеленных на оптимизацию серверной части системы, загрузка наиболее сложной страницы WEB-сайта системы “Профиль пользователя” ускорилась почти в 5 раз и составило время, наиболее комфортное для пользователей. Результаты измерения загрузки страницы профиля пользователя приведены в приложении Д.

## Анализ сбора геоданных в системе “Coordinate”

Сбор геоданных осуществляется мобильным приложением-трекером на устройствах, под управлением систем семейства Android.

Мобильное приложение обеспечивает определение местоположения и передачу геоданных на сервер. Система коммуникации сервиса не ориентирована на конкретную ОС трекера, и может работать с любой ОС, под которую существует соответствующий клиент. Для коммуникации в Rails приложении был создан API. Передача данных осуществляется в формате JSON по зашифрованному SSL каналу с использованием протокола HTTPS. Трекер должен передавать основные данные о местоположении пользователя в фоновом режиме. Полученные от трека данные отображаются на карте. Длительность пути и юридический адрес не предоставляются трекером, а определяются в серверном приложении.

На рисунке 4.10 представлена функциональная схема мобильного приложения “Coordinate-Tracker”.

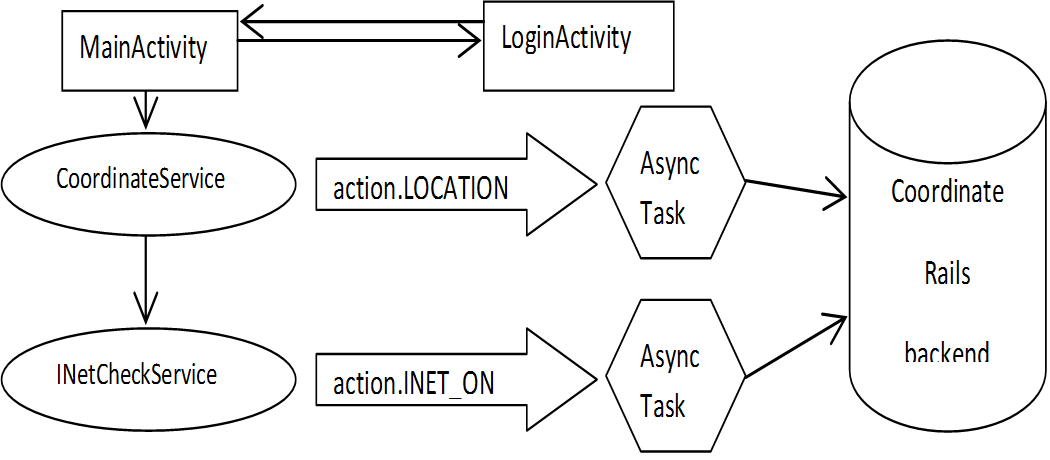


Рисунок . Функциональная схема мобильного приложения-трекера.

Авторизация позволяет уникально идентифицировать автомобиль пользователя при первом запуске приложения-трекера на мобильном устройстве пользователя.

Авторизация выполняется в несколько этапов, и является реализацией алгоритма oAuth 2.0 в несколько усовершенствованной форме.

Пользователю, впервые запустившему мобильное приложение-трекер предоставляется LoginActivity, в котором он должен заполнить данные учетной записи (email, пароль) и уникальный идентификатор, выданный при создании автомобиля, представляющий собой случайно сгенерированную шестнадцатеричную последовательность, длиной 8.

Затем, email и пароль хешируются по алгоритму SHA256 и хеш-сумма, полученная в результате, передается на сервер вместе с уникальным идентификатором по протоколу HTTPS, использующий SSL шифрование. По хеш-сумме находится пользователь системы, а из списка автомобилей пользователя находится автомобиль, с полученным уникальным идентификатором. Для данного автомобиля генерируется уникальный API-токен, который передается мобильному приложению-трекеру со статусом успешной авторизацией.

Данный API-токен сохраняется в локальном хранилище мобильного приложения и передается с каждым запросом к сервису в заголовке, тем самым достигая успешную аунтефикацию пользователя.

Пользователь имеет возможность отвязать трекер от системы, выбрав соответствующий пункт меню (LOGOUT), при этом все API-токен будет уничтожен, и пользователю будет вновь предложен ввод регистрационных данных на LoginActivity.

После успешной авторизации происходит запуск MainActivity, в котором содержится область с информацией о работе сервисов и кнопки управления. Кроме того, при старте MainActivity стартует сервис CoordinateService, управляющий фоновыми процессами приложения.

При старте, CoordinateService выполняет старт INetCheckService – сервиса, проверяющего доступность API сервиса, по специальному адресу, в некоторый промежуток времени (раз в минуту). Если получен ожидаемый ответ - генерируется событие action.INET\_ON, означающее наличие соединения с сервисом. Если же по причине плохого соединения или внутренней серверной ошибки ожидаемый ответ не получен – сервис генерирует событие action.INET\_OFF.

Кроме того, CoordinateService активирует деятельность, опрашивающую текущее местоположение в GPS координатах по определенному таймингу. При получении таких координат, и, в случае их корректности, активируется событие action.LOCATION, содержащие основные параметры геолокации (широта, долгота, время, скорость, точность).

Асинхронный обработчик события action.LOCATION отправляет геолокацию на сервер в HTTPS – запросе, используя шифрование SSL. Если ожидаемый ответ от сервера не получен (ошибка соединения, внутренняя ошибка сервера, и т.п.), то геолокация сохраняется в локальное хранилище мобильного приложения – трекера.

Асинхронный обработчик события action.INET\_ON передает содержимое локального хранилища мобильного приложения – трекера на сервер, и в случае ожидаемого ответа, удаляет советующую запись из локального хранилища.

За период пользования мобильного приложения – трекера, был выявлен ряд задач, требующих переработки:

* Работа CoordinateService не должна быть прозрачной для пользователя: пользователь должен видеть текущее состояние сервиса CoordinateService, значение его основных параметров при фоновой работе.
* Сетевые подключения не используют достоинства HTTP/2, получившего широкое распространение с начала 2016 года.
* Для снижении ресурсоемкости, количество запросов должно быть снижено, а многочисленные соединений должны быть собраны в пул запросов.

Прозрачность фоновой работы сервиса CoordinateService была достигнута за счет добавления нескрываемым push-уведомлением с информацией о работе сервисов. Теперь, при запуске сервиса CoordinateService в трее появляется уведомление, по которому пользователь способен не только увидеть некоторую информацию о текущем состоянии сервиса (статус подключения, число геолокаций в локальном хранилище, примерный адрес местоположения, примерная скорость), но и менять состояние сервиса (приостанавливать отслеживание, например).

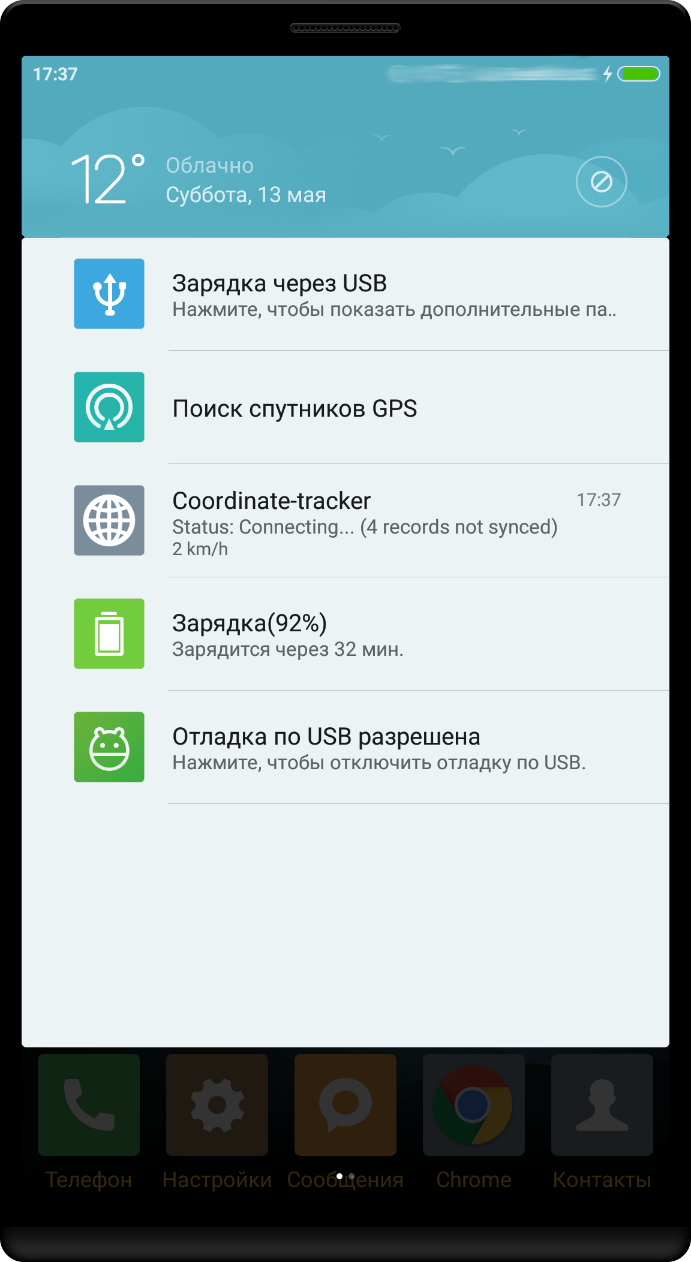


Рисунок . Уведомление с информацией о фоновой работе CoordinateService.

Обновление модуля сетевых подключений до версии, поддерживающей стандарт HTTP/2 и снижение ресурсоемкости было достигнуто за счет переработки поведения коммуникаций мобильного приложения – трекера с сервером.

Основные коммуникации с сервером происходят в трех функциональных частях мобильного-приложения:

1. В сервисе INetCheckService при отправки HTTP – запросов для проверки соединения с сервером.
2. В асинхронном обработчике при получении новой геолокации для отправки геоданных на сервер.
3. В асинхронном обработчике при восстановлении соединения с сервером для синхронизации локального хранилища геоданных с сервером.
4. При авторизации пользователей.

Во всех четырех случаях, в рамках обновления ПО был изменен API, отвечающий за предоставления подключений с Apache httpclient (<https://hc.apache.org/>) на okHTTP (<http://square.github.io/okhttp/>), предоставляющий возможности HTTP/2.

Так как процесс авторизации выполняется достаточно редко, и содержит единственный запрос за итерацию, оптимизация сетевых подключений в нем не является приоритетным.

Несмотря на то, что HTTP – запросы в сервисе INetCheckService выполняются на всем протяжении работы мобильного приложения-трекера, объем передаваемых данных в запросе минимален, а запрос так же единственный на всю итерацию, что дает основания говорить, о том, что данная коммуникация с сервером не нуждается в оптимизации.

Таким образом, в оптимизации нуждаются два основных алгоритма мобильного приложения – трекера: алгоритм передачи геоданных и алгоритм синхронизации локального хранилища.

Алгоритм передачи приведен в приложении Е.1. Несмотря на его минимальность, он наделен одной, очень неприятной чертой – многочисленный вызовы. Данный алгоритм отрабатывает всякий раз, когда провайдер GPS – провайдер предоставляет координаты, удовлетворяющие следующим условиям:

1. Разница по времени с последними полученными координатами не должна быть меньше 5 сек.
2. Расстояние между последними координатами не должно превышать 10м.

Исходя из расчётов, при скорости движения в 30 км./ч. за 1 минуту алгоритм должен отработать около 50 раз, т.е должно быть сделано около 50 соединений с сервером. Такое количество соединений является неприемлемым так кая является очень ресурсоемким, особенно в вопросе энергопотребления.

Для уменьшения количества соединений необходимо воспользоваться пулом сетевых соединений и пулом геолокационных данных. Пул сетевых соединений ограничит число одновременно доступных подключений и позволит выделить ресурсы системы под подключения один раз при инициализации, а не каждый раз при создании подключения. За пул сетевых подключений отвечает API okHTTP. Пул геолокаций реализован самостоятельно, с помощью системного хранилища, и позволяет передавать полученные координаты массивами. Наиболее оптимальными параметрами являются 5 для активных соединений и 20 для геолокаций.

Алгоритм отправки геолокаций после оптимизации приведен в приложении Е.2. В нем данные передаются на сервер только в том случае, когда пул геолокаций полон (более 20) и присутствует соединение с API сервиса. В таком виде количество открытых соединений будет ниже, а, значит, и энергопотребление меньше. В условиях приведенного выше примера, при движении в 30км/ч. за минуту, число подключений будет снижено всего до трех.

Алгоритм синхронизации локальных данных с сервером приведен в приложении Е.3. Аналогично с алгоритмом передачи геолокаций, здесь используется большое число подключений, равное количеству геоданных в локальном хранилище. Но ситуация с ресурсоемкостью здесь еще хуже, так как каждая итерация обхода локального хранилища подразумевает создания нового потока, что многократно повышает потребление памяти и энергоресурсов. Оптимизация алгоритма синхронизации локальных данных заключается в передачи всего массива данных в единственном запросе, что сокращает количество выделяемых ресурсов системы порядка на 40% процентов, но повышает объем трафика между мобильным приложением – трекером, так как данная модификация подразумевает более объемный ответ от сервера, со списком идентификаторов геолокаций, которые были успешно обработаны системой и нуждаются в очистке из локального хранилища.

Оптимизированный алгоритм синхронизации локальных данных приведен в приложении Е.4.

Таким образом, оптимизация основных алгоритмов коммуникаций с сервером позволила сократить потребление энергоресурсов и выделения ресурсов системы, как показала опытная эксплуатация, более чем на 30%. Использование пула соединений позволило сократить выделение ресурсов в каждый момент соединения с сервером, и ограничить максимально возможное число одновременно активных подключений к серверу, что понизило нагрузку как на мобильное приложение-трекер, так и на сервер системы.

# Экономический анализ системы

## Расчет экономических затрат

Рассчитываем затраты на разработку и модернизацию проекта + на внедрение с учетом масштабирования.

## Расчет экономической рентабельности

Исходя из п. 3.1 рассчитываем экономическую рентабельность проекта.

# Заключительная часть

## Результаты исследования

В ходе проведенной работы по анализу информационной системы “Coordinate”, был выделен ряд задач, решение которых должно было положительно сказаться на оптимизации системы. Наиболее критичные задачи были решены в рамках данной работы, и их результат привел к положительным результатам. Время отклика системы стало на порядок ниже, а объем используемых данных стал более оптимально структурирован, что привело к заметному уменьшению объема хранимых и обрабатываемых данных.

|  |  |  |  |
| --- | --- | --- | --- |
| Фактор | Значение до модернизации | Значение после модернизации | Прирост производительности |
| Оптимизация БД | 100,00% | 83,33% | 16,67% |
| Оптимизация серверных процессов | 100,00% | 60,00% | 40,00% |
| Оптимизация построения представления | 100,00% | 80,00% | 20,00% |
| Объем трафика GPS-трекера | 100,00% | 60,00% | 40,00% |

Таблица . Общие результаты оптимизации компонентов системы “Coordinate”

Таблица 6.1 наглядно демонстрирует приблизительный прирост производительности компонентов системы, в результате исследования и анализа системы “Coordinate” в ходе текущей работы. Можно заметить, что общий прирост производительности составляет порядка 30%, что является высоким значением.

По завершению модернизации системы, ее работоспособность стала удовлетворять запросам пользователей, выявленных в ходе опроса, результаты которого приведены в разделе 3.2. Кроме того, повысилась надежность системы, исправлен ряд критических ошибок.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Задача | Критичность | Решена | Решение найдено | Решение не найдено |
| Оптимизация запросов БД | Критичная | + |  |  |
| Оптимизация формата хранения данных | Критичная | + |  |  |
| Оптимизация алгоритма обработки геоданных | Критичная | + |  |  |
| Уменьшение времени взаимодействия с сервером | Критичная | + |  |  |
| Кеширование промежуточных результатов | Важная |  |  | + |
| Повышение эргономичности интерфейса | Важная | + |  |  |
| Исправление пула ошибок | Критичная | + |  |  |
| Добавление новой статистической информации | Нормальная |  | + |  |
| Реализация возможности сброса статистики | Важная |  | + |  |
| Повышение точности сбора геоданных | Критичная | + |  |  |
| Оптимизация взаимодействия трекера с сервером | Критичная | + |  |  |
| Переработка архитектуры мобильного трекера | Критичная |  | + |  |
| Модификация мобильного приложения-трекера до полнофункционального клиента сервиса (предоставление всех функций, предоставляемых Web-сайтом) | Важная |  |  | + |

Таблица .. Задачи, выявленные в ходе анализа системы “Coordinate”

В таблице 6.2 перечислен перечень задач, выявленных в ходе исследования сервиса Coordinate в рамках данной работы. Наиболее критичные задачи были решены и функционал внедрен в систему. Решения задач, реализация которых не критична для функционала системы и достижения поставленных целей, но при этом обладающих высокой трудоемкостью, вынесены за рамки данной работы. Остальные задачи были подвержены анализу, на основании которого было предложено наиболее подходящее решение.

Таким образом, в ходе работы удалось достичь поставленных целей по модернизации сервиса “Coordinate” и в конечном итоге, оправдать желания опрошенных пользователей.

## Заключение

Для выпускной квалификационной работы, была поставлена задача исследовать сервис мониторинга автотранспорта. Для достижении этой цели были изучены современные паттерны построения Web-приложений и архитектуры (MVC, REST), методологии ведения проектов и построения информационных стстем, изучены специфичные средства и инструменты, проработаны вопросы надежности и безопасности. Обновленное и проработанное приложение было развернуто и открыто для общего доступа в сети интернет.

В ходе исследования системы были обнаружены и исправлены критические ошибки, добавлен новый функционал, а существующие решения были оптимизирована. Система полностью удовлетворяет результатам опроса респондентов, что делает ее готовой к внедрению и эксплуатации.

Между тем, остался ряд актуальных вопросов, которые могут послужить для дальнейшей модернизации системы с целью расширения функционала.

Все затраты, возникшие в ходе текущей работы признаются обоснованными, а модернизация системы считается успешной.

Цели выпускной квалификационной работы достигнуты. В работе решены следующие задачи:

* проведены анализ и изучение существующих решений;
* проведен опрос пользователей;
* модернизирована структурная модель сервиса;
* модернизирована архитектура БД;
* оптимизирована серверная компонента на ruby on rails
* переработан Web-сайт системы;
* усовершенствован API для мобильного приложения
* модернизирован алгоритм получения и передачи координат;
* обновлены все компоненты системы;
* модернизированная система внедрена и предоставлен доступ конечным пользователям.
* проведено комплексное тестирование;
* проведено документирование в виде оформления отчета выпускной квалификационной работы, а также частичное документирование исходного кода с использованием RDoc для ruby.

В рамках работы автор получил богатый объем опыта и знаний в области проектирования и анализа ИС, ознакомился с современными инструментами и методологиями, используемых при построении картографических сервисов.

# Список сокращений и условных обозначений

* БД – База Данных;
* ГБ - Гигабайт;
* ГГц - Гигагерц;
* ЛВС - Локальная вычислительная сеть;
* ОС - Операционная Система;
* СУБД - Система Управления Базами Данных;
* API - Application Programming Interface;
* GPS - Global Positioning System;
* HTML - HyperText Markup Language;
* HTTP - HyperText Transfer Protocol;
* HTTPS - HyperText Transfer Protocol Secure;
* JPEG - Joint Photographic Experts Group;
* JSON - JavaScript Object Notation;
* PNG - Portable Network Graphics;
* RHEL - Red Hat Enterprise Linux;
* SQL - Structured Query Language;
* SSL - Secure Sockets Layer;
* USB - Universal Serial Bus.

# Список литературы

Приложение А

**Методы анализа БД**

А.1. Получение информации о таблицах БД

1 SELECT

2 relname as "Таблица",

3 pg\_size\_pretty(pg\_total\_relation\_size(relid)) As "Общий размер",

4 pg\_size\_pretty(pg\_total\_relation\_size(relid)-pg\_relation\_size(relid)) as "Индексы, VM, FSM"

5 FROM pg\_catalog.pg\_statio\_user\_tables ORDER BY pg\_total\_relation\_size(relid) DESC;

А.2. Расчет среднего числа маршрутов на автомобиль

1 SELECT

2 ROUND(avg) AS "срд. знач. марш./авт."

3 FROM (

4 SELECT

5 AVG(count)

6 from (

7 SELECT

8 COUNT(\*)

9 FROM

10 tracks AS t LEFT JOIN cars AS c ON t.car\_id = c.id

11 WHERE

12 user\_id = ?

13 GROUP BY c.id

14 ) AS cars

15 ) AS cars\_grouped;

**А.3. Расчет среднего числа точек на маршрут**

1 SELECT

2 ROUND(avg) AS "срд. знач. лок./марш."

3 FROM (

4 SELECT

5 AVG(count)

6 FROM (

7 SELECT

8 COUNT(l.id)

9 FROM locations l LEFT JOIN tracks t ON t.id = l.track\_id

10 WHERE

11 t.car\_id IN (

12 SELECT

13 c.id

14 FROM

15 cars AS c

16 WHERE c.user\_id = ?

17 )

18 GROUP BY t.id

19 ) AS tracks

20 ) AS tracks\_grouped;

А.4 Подробная информация по таблице

1 WITH x AS (

2 SELECT count(\*) AS ct

3 ,sum(length(t::text)) AS txt\_len

4 ,'locations'::regclass AS tbl

5 FROM locations t

6 )

7 , y AS (

8 SELECT ARRAY [

9 pg\_relation\_size(tbl)

10 ,pg\_relation\_size(tbl, 'vm')

11 ,pg\_relation\_size(tbl, 'fsm')

12 ,pg\_table\_size(tbl)

13 ,pg\_indexes\_size(tbl)

14 ,pg\_total\_relation\_size(tbl)

15 ,txt\_len

16 ] AS val

17 ,ARRAY [

18 'Общий размер отношения'

19 ,'Карта видимости'

20 ,'Размер свобоного пространства'

21 ,'Размер таблицы, включая TOAST'

22 ,'Размер индексов'

23 ,'Общий размер, включая TOAST и индексы'

24 ,'Кол-во используемых строк в текстовом представлении'

25 ] AS name

26 FROM x

27 )

28 SELECT unnest(name) AS "'Параметр'"

29 ,unnest(val) AS "В байтах"

30 ,pg\_size\_pretty(unnest(val)) AS "В удобночитаемом виде"

31 ,unnest(val) / ct AS "Байт на строку"

32 FROM x,y

33 UNION ALL

34 SELECT '----------'::text, NULL::int8, '----'::text, NULL::int8

35 UNION ALL

36 SELECT 'Число строк'::text, ct,

37 NULL::text, NULL::bigint FROM x

38 UNION ALL

39 SELECT 'Число действующих кортежей'::text, pg\_stat\_get\_live\_tuples(tbl),

40 NULL::text, NULL::bigint FROM x

41 UNION ALL

42 SELECT '

А.5 Подробная информация по индексам таблицы

1 SELECT

2 i.indexname AS "Индекс",

3 i.tablename AS "Таблица",

4 i.num\_rows AS "Число кортежей",

5 i.table\_size AS "Размер таблицы",

6 i.index\_size AS "Размер индекса",

7 i.unique AS "Уникальность",

8 i.number\_of\_scans AS "Число использований",

9 i.tuples\_read AS "Прочитано, с помощью индекса",

10 i.tuples\_fetched AS "Выбрано, с помощью индекса"

11 FROM (

12 SELECT

13 t.tablename,

14 indexname,

15 c.reltuples AS num\_rows,

16 pg\_size\_pretty(pg\_relation\_size(quote\_ident(t.tablename)::text)) AS table\_size,

17 pg\_size\_pretty(pg\_relation\_size(quote\_ident(indexrelname)::text)) AS index\_size,

18 CASE WHEN indisunique THEN 'Y'

19 ELSE 'N'

20 END AS UNIQUE,

21 idx\_scan AS number\_of\_scans,

22 idx\_tup\_read AS tuples\_read,

23 idx\_tup\_fetch AS tuples\_fetched

24 FROM pg\_tables t

25 LEFT OUTER JOIN pg\_class c ON t.tablename=c.relname

26 LEFT OUTER JOIN

27 ( SELECT c.relname AS ctablename, ipg.relname AS indexname, x.indnatts AS number\_of\_columns, idx\_scan, idx\_tup\_read, idx\_tup\_fetch, indexrelname, indisunique FROM pg\_index x

28 JOIN pg\_class c ON c.oid = x.indrelid

29 JOIN pg\_class ipg ON ipg.oid = x.indexrelid

30 JOIN pg\_stat\_all\_indexes psai ON x.indexrelid = psai.indexrelid )

31 AS foo

32 ON t.tablename = foo.ctablename

33 WHERE t.schemaname='public' AND c.reltuples > 0

34 ORDER BY number\_of\_scans DESC, 1 ASC, 2 ASC

А.6 Выборка маршрута автомобиля с максимальной дистанцией (до оптимизации)

1 EXPLAIN (ANALYZE) SELECT

2 MAX("locations"."distance") AS maximum\_distance,

3 "locations"."track\_id" AS locations\_track\_id

4 FROM

5 "locations"

6 WHERE

7 "locations"."state" IN ('TrackLocation') AND

8 "locations"."track\_id" IN (

9 SELECT

10 id

11 FROM

12 tracks

13 WHERE

14 car\_id = ?

15 )

16 GROUP BY "locations"."track\_id";

А.7 План построения запроса: выборка маршрута автомобиля с максимальной дистанцией (до оптимизации)

HashAggregate (cost=7774.21..7778.75 rows=454 width=12) (actual time=454.555..454.676 rows=403 loops=1)

-> Hash Join (cost=15.91..7253.63 rows=104116 width=12) (actual time=1.385..370.820 rows=102941 loops=1)

Hash Cond: (locations.track\_id = tracks.id)

-> Seq Scan on locations (cost=0.00..5792.15 rows=117292 width=12) (actual time=0.397..225.732 rows=117046 loops=1)

Filter: ((state)::text = 'TrackLocation'::text)

Rows Removed by Filter: 2

-> Hash (cost=10.88..10.88 rows=403 width=4) (actual time=0.953..0.953 rows=403 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 32kB

-> Seq Scan on tracks (cost=0.00..10.88 rows=403 width=4) (actual time=0.048..0.563 rows=403 loops=1)

Filter: (car\_id = ?)

Rows Removed by Filter: 67

Total runtime: 454.983 msby Filter: 67

12 Total runtime: 454.983 ms

А.8 Выборка маршрута автомобиля с максимальной дистанцией (после оптимизации)

1 EXPLAIN (ANALYZE) SELECT

2 MAX("locations"."distance") AS maximum\_distance,

3 "locations"."track\_id" AS locations\_track\_id

4 FROM

5 "locations"

6 WHERE

7 "locations"."track\_id" IN (

8 SELECT

9 id

10 FROM

11 tracks

12 WHERE

13 car\_id = ?

14 )

15 GROUP BY "locations"."track\_id";

А.9 План построения запроса: выборка маршрута автомобиля с максимальной дистанцией (после оптимизации)

HashAggregate (cost=7375.99..7380.57 rows=458 width=12) (actual time=198.302..198.430 rows=406 loops=1)

-> Hash Join (cost=15.99..6856.52 rows=103895 width=12) (actual time=0.582..158.951 rows=103097 loops=1)

Hash Cond: (locations.track\_id = tracks.id)

-> Seq Scan on locations (cost=0.00..5395.02 rows=117202 width=12) (actual time=0.026..96.341 rows=117202 loops=1)

-> Hash (cost=10.91..10.91 rows=406 width=4) (actual time=0.497..0.497 rows=406 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 32kB

-> Seq Scan on tracks (cost=0.00..10.91 rows=406 width=4) (actual time=0.029..0.244 rows=406 loops=1)

Filter: (car\_id = 1)

Rows Removed by Filter: 67

Total runtime: 198.651 ms

Приложение Б

**Результаты анализа журналов входящих запросов сервера**

Б.1. Общая информация о проведенном анализе

|  |  |
| --- | --- |
| Анализируемый файл | /var/log/httpd/coordinate/production.log |
| Прочитано строк | 7898 |
| Пропущено строк | 169 |
| Анализировано запросов | 1842 |
| Пропущено запросов | 0 |
| Дата первого запроса | 02.04.2017 12:50 |
| Дата последнего запроса | 10.04.2017 2:57 |
| Анализируемое время | 8 дней |

Б.2. Среднее число запросов по часам

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Время | 0:00 | 1:00 | 2:00 | 3:00 | 4:00 | 5:00 | 6:00 | 7:00 | 8:00 | 9:00 | 10:00 | 11:00 |
| Число запросов | 28 | 4 | 16 | 1 | 1 | 18 | 20 | 6 | 4 | 1 | 1 | 2 |
| Время | 12:00 | 13:00 | 14:00 | 15:00 | 16:00 | 17:00 | 18:00 | 19:00 | 20:00 | 21:00 | 22:00 | 23:00 |
| Число запросов | 40 | 75 | 2 | 2 | 15 | 9 | 2 | 1 | 1 | 1 | 1 | 4 |

Б.3. Частота запросов к различным методам контроллеров

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемый метод | Число запросов | Процентное соотношение |
| 1 | Api::V1::GeodataController#recive.JSON | 973 | 52.8% |
| 2 | Api::V1::SessionsController#ping.JSON | 284 | 15.4% |
| 3 | #. | 253 | 13.7% |
| 4 | UsersController#charts\_controller.JSON | 66 | 3.6% |
| 5 | StaticPagesController#home.HTML | 43 | 2.3% |
| 6 | Maps::TracksController#info.JS | 39 | 2.1% |
| 7 | Maps::LocationsController#index.JSON | 36 | 2.0% |
| 8 | CarsController#info.JSON | 32 | 1.7% |
| 9 | Maps::CarsController#index.JSON | 24 | 1.3% |
| 10 | Maps::TracksController#index.JSON | 24 | 1.3% |
| 11 | StaticPagesController#home.\*/\* | 21 | 1.1% |
| 12 | UsersController#show.HTML | 17 | 0.9% |
| 13 | Maps::LocationsController#show.JSON | 10 | 0.5% |
| 14 | StaticPagesController#about.\*/\* | 6 | 0.3% |
| 15 | Admin::DashboardController#index.HTML | 2 | 0.1% |
| 16 | SessionsController#create.HTML | 2 | 0.1% |
| 17 | StaticPagesController#faq.\*/\* | 2 | 0.1% |
| 18 | Admin::TicketsController#index.HTML | 1 | 0.1% |
| 19 | Admin::UsersController#show.HTML | 1 | 0.1% |
| 20 | SessionsController#new.HTML | 1 | 0.1% |

Б.4. Блокирующие запросы (длительность более 1сек.)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемый метод | Число запросов | Процентное соотношение |
| 1 | UsersController#charts\_controller.JSON | 25 | 41.0% |
| 2 | UsersController#show.HTML | 10 | 16.4% |
| 3 | Api::V1::GeodataController#recive.JSON | 9 | 14.8% |
| 4 | CarsController#info.JSON | 6 | 9.8% |
| 5 | Maps::TracksController#index.JSON | 5 | 8.2% |
| 6 | StaticPagesController#home.HTML | 2 | 3.3% |
| 7 | Maps::LocationsController#index.JSON | 2 | 3.3% |
| 8 | #. | 1 | 1.6% |
| 9 | Admin::DashboardController#index.HTML | 1 | 1.6% |

Б.5. Трудоемкость запросов (по сумме временных значений)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод контроллера | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Api::V1::GeodataController#recive.JSON | 959 | 3m42s | 231ms | 111ms | 5.80s | 132ms-564ms |
| Api::V1::SessionsController#ping.JSON | 282 | 2.61s | 9ms | 1ms | 248ms | 1ms-30ms |
| UsersController#charts\_controller.JSON | 58 | 1m14s | 1.27s | 15ms | 5.20s | 26ms-4.74s |
| StaticPagesController#home.HTML | 41 | 14.06s | 342ms | 68ms | 2.35s | 77ms-2.37s |
| Maps::TracksController#info.JS | 37 | 9.92s | 267ms | 42ms | 612ms | 55ms-612ms |
| Maps::TracksController#index.JSON | 22 | 19.20s | 872ms | 336ms | 2.04s | 331ms-2.05s |
| Maps::LocationsController#index.JSON | 22 | 7.81s | 354ms | 43ms | 1.29s | 42ms-1.31s |
| Maps::CarsController#index.JSON | 22 | 1.22s | 55ms | 12ms | 307ms | 11ms-311ms |
| StaticPagesController#home.\*/\* | 21 | 7.76s | 369ms | 114ms | 907ms | 112ms-916ms |
| #. | 16 | 6.42s | 401ms | 12ms | 1.05s | 11ms-1.06s |
| UsersController#show.HTML | 16 | 17.85s | 1.12s | 55ms | 2.73s | 54ms-2.74s |
| CarsController#info.JSON | 11 | 13.55s | 1.23s | 228ms | 2.34s | 225ms-2.37s |
| Maps::LocationsController#show.JSON | 10 | 620ms | 62ms | 17ms | 216ms | 16ms-218ms |
| StaticPagesController#about.\*/\* | 6 | 1.84s | 307ms | 196ms | 379ms | 194ms-383ms |
| StaticPagesController#faq.\*/\* | 2 | 577ms | 288ms | 282ms | 295ms | 277ms-296ms |
| SessionsController#create.HTML | 2 | 453ms | 226ms | 171ms | 283ms | 168ms-287ms |
| Admin::DashboardController#index.HTML | 2 | 2.87s | 1.43s | 967ms | 1.90s | 961ms-1.92s |
| Admin::TicketsController#index.HTML | 1 | 445ms | 445ms | 445ms | 445ms | 443ms-458ms |
| Admin::ExceptionsController#index.HTML | 1 | 433ms | 433ms | 433ms | 433ms | 429ms-443ms |
| Admin::UsersController#show.HTML | 1 | 385ms | 385ms | 385ms | 385ms | 383ms-396ms |

Б.6. Трудоемкость запросов (по средним значениям)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод контроллера | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Api::V1::GeodataController#recive.JSON | 959 | 3m42s | 231ms | 111ms | 5.80s | 132ms-564ms |
| UsersController#charts\_controller.JSON | 58 | 1m14s | 1.27s | 15ms | 5.20s | 26ms-4.74s |
| StaticPagesController#home.HTML | 41 | 14.06s | 342ms | 68ms | 2.35s | 77ms-2.37s |
| Maps::TracksController#info.JS | 37 | 9.92s | 267ms | 42ms | 612ms | 55ms-612ms |
| Maps::TracksController#index.JSON | 22 | 19.20s | 872ms | 336ms | 2.04s | 331ms-2.05s |
| Maps::LocationsController#index.JSON | 22 | 7.81s | 354ms | 43ms | 1.29s | 42ms-1.31s |
| StaticPagesController#home.\*/\* | 21 | 7.76s | 369ms | 114ms | 907ms | 112ms-916ms |
| #. | 16 | 6.42s | 401ms | 12ms | 1.05s | 11ms-1.06s |
| UsersController#show.HTML | 16 | 17.85s | 1.12s | 55ms | 2.73s | 54ms-2.74s |
| CarsController#info.JSON | 11 | 13.55s | 1.23s | 228ms | 2.34s | 225ms-2.37s |
| StaticPagesController#about.\*/\* | 6 | 1.84s | 307ms | 196ms | 379ms | 194ms-383ms |
| StaticPagesController#faq.\*/\* | 2 | 577ms | 288ms | 282ms | 295ms | 277ms-296ms |
| SessionsController#create.HTML | 2 | 453ms | 226ms | 171ms | 283ms | 168ms-287ms |
| Admin::DashboardController#index.HTML | 2 | 2.87s | 1.43s | 967ms | 1.90s | 961ms-1.92s |
| Admin::TicketsController#index.HTML | 1 | 445ms | 445ms | 445ms | 445ms | 443ms-458ms |
| Admin::ExceptionsController#index.HTML | 1 | 433ms | 433ms | 433ms | 433ms | 429ms-443ms |
| Admin::UsersController#show.HTML | 1 | 385ms | 385ms | 385ms | 385ms | 383ms-396ms |
| StaticPagesController#about.HTML | 1 | 264ms | 264ms | 264ms | 264ms | 260ms-269ms |
| SessionsController#new.HTML | 1 | 224ms | 224ms | 224ms | 224ms | 221ms-229ms |
| Admin::ExceptionsController#show.HTML | 1 | 216ms | 216ms | 216ms | 216ms | 214ms-221ms |

Б.7. Трудоемкость генерации представлений (по сумме временных значений)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Представление | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Api::V1::GeodataController#recive.JSON | 959 | 2.35s | 2ms | 0ms | 1.10s | 0ms-5ms |
| Api::V1::SessionsController#ping.JSON | 277 | 1.02s | 3ms | 0ms | 27ms | 0ms-14ms |
| UsersController#charts\_controller.JSON | 54 | 548ms | 10ms | 0ms | 41ms | 0ms-30ms |
| StaticPagesController#home.HTML | 41 | 10.66s | 260ms | 51ms | 1.86s | 51ms-1.86s |
| Maps::TracksController#info.JS | 37 | 2.40s | 64ms | 4ms | 405ms | 4ms-409ms |
| Maps::TracksController#index.JSON | 22 | 17.52s | 796ms | 270ms | 1.73s | 269ms-1.75s |
| Maps::CarsController#index.JSON | 22 | 712ms | 32ms | 4ms | 209ms | 4ms-211ms |
| Maps::LocationsController#index.JSON | 22 | 4.75s | 215ms | 16ms | 1.03s | 16ms-1.04s |
| StaticPagesController#home.\*/\* | 21 | 6.05s | 288ms | 86ms | 782ms | 85ms-792ms |
| #. | 16 | 1.05s | 65ms | 1ms | 646ms | 1ms-653ms |
| UsersController#show.HTML | 15 | 5.96s | 397ms | 106ms | 820ms | 105ms-831ms |
| StaticPagesController#about.\*/\* | 6 | 1.44s | 240ms | 150ms | 283ms | 148ms-287ms |
| StaticPagesController#faq.\*/\* | 2 | 438ms | 219ms | 209ms | 229ms | 207ms-232ms |
| Admin::DashboardController#index.HTML | 2 | 2.30s | 1.15s | 757ms | 1.54s | 755ms-1.56s |
| Admin::ExceptionsController#index.HTML | 1 | 392ms | 392ms | 392ms | 392ms | 389ms-402ms |
| Admin::UsersController#show.HTML | 1 | 319ms | 319ms | 319ms | 319ms | 316ms-326ms |
| Admin::TicketsController#index.HTML | 1 | 239ms | 239ms | 239ms | 239ms | 236ms-244ms |
| StaticPagesController#about.HTML | 1 | 223ms | 223ms | 223ms | 223ms | 221ms-229ms |
| Admin::ExceptionsController#show.HTML | 1 | 177ms | 177ms | 177ms | 177ms | 177ms-182ms |
| SessionsController#new.HTML | 1 | 176ms | 176ms | 176ms | 176ms | 174ms-179ms |

Б.8. Трудоемкость генерации представлений (по средним значениям)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод контроллера | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| UsersController#charts\_controller.JSON | 54 | 548ms | 10ms | 0ms | 41ms | 0ms-30ms |
| StaticPagesController#home.HTML | 41 | 10.66s | 260ms | 51ms | 1.86s | 51ms-1.86s |
| Maps::TracksController#info.JS | 37 | 2.40s | 64ms | 4ms | 405ms | 4ms-409ms |
| Maps::TracksController#index.JSON | 22 | 17.52s | 796ms | 270ms | 1.73s | 269ms-1.75s |
| Maps::CarsController#index.JSON | 22 | 712ms | 32ms | 4ms | 209ms | 4ms-211ms |
| Maps::LocationsController#index.JSON | 22 | 4.75s | 215ms | 16ms | 1.03s | 16ms-1.04s |
| StaticPagesController#home.\*/\* | 21 | 6.05s | 288ms | 86ms | 782ms | 85ms-792ms |
| #. | 16 | 1.05s | 65ms | 1ms | 646ms | 1ms-653ms |
| UsersController#show.HTML | 15 | 5.96s | 397ms | 106ms | 820ms | 105ms-831ms |
| StaticPagesController#about.\*/\* | 6 | 1.44s | 240ms | 150ms | 283ms | 148ms-287ms |
| StaticPagesController#faq.\*/\* | 2 | 438ms | 219ms | 209ms | 229ms | 207ms-232ms |
| Admin::DashboardController#index.HTML | 2 | 2.30s | 1.15s | 757ms | 1.54s | 755ms-1.56s |
| Admin::ExceptionsController#index.HTML | 1 | 392ms | 392ms | 392ms | 392ms | 389ms-402ms |
| Admin::UsersController#show.HTML | 1 | 319ms | 319ms | 319ms | 319ms | 316ms-326ms |
| Admin::TicketsController#index.HTML | 1 | 239ms | 239ms | 239ms | 239ms | 236ms-244ms |
| StaticPagesController#about.HTML | 1 | 223ms | 223ms | 223ms | 223ms | 221ms-229ms |
| Admin::ExceptionsController#show.HTML | 1 | 177ms | 177ms | 177ms | 177ms | 177ms-182ms |
| SessionsController#new.HTML | 1 | 176ms | 176ms | 176ms | 176ms | 174ms-179ms |
| UsersController#reset\_password.\*/\* | 1 | 88ms | 88ms | 88ms | 88ms | 88ms-91ms |
| Maps::TracksController#info\_all.JS | 1 | 19ms | 19ms | 19ms | 19ms | 19ms-20ms |

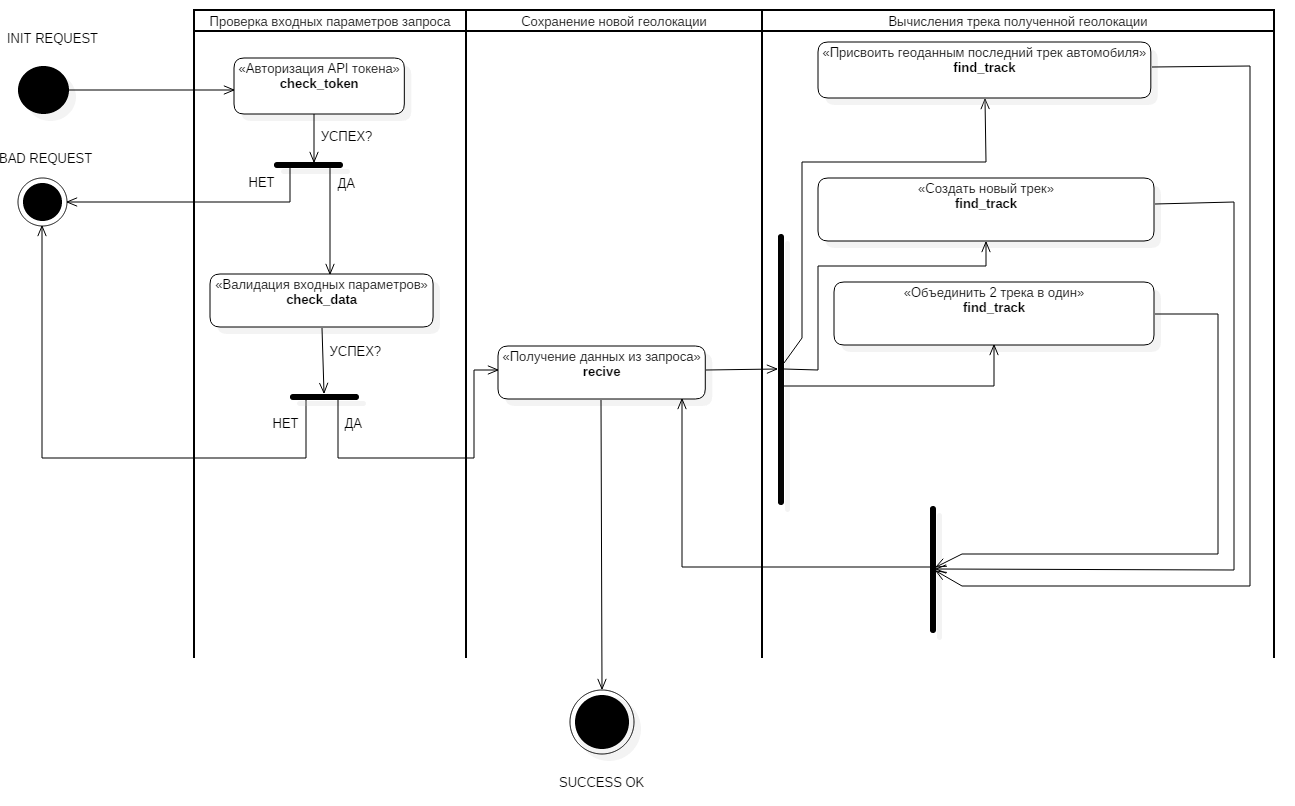
Б.9. Трудоемкость обращений к БД (по сумме значений)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Представление | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Api::V1::GeodataController#recive.JSON | 959 | 18.49s | 19ms | 5ms | 270ms | 8ms-48ms |
| Api::V1::SessionsController#ping.JSON | 277 | 60ms | 0ms | 0ms | 23ms | 0ms-0ms |
| UsersController#charts\_controller.JSON | 54 | 34.55s | 639ms | 20ms | 4.01s | 28ms-3.44s |
| StaticPagesController#home.HTML | 41 | 1.34s | 32ms | 9ms | 161ms | 9ms-163ms |
| Maps::TracksController#info.JS | 37 | 1.11s | 30ms | 5ms | 87ms | 6ms-88ms |
| Maps::TracksController#index.JSON | 22 | 411ms | 18ms | 3ms | 183ms | 3ms-185ms |
| Maps::LocationsController#index.JSON | 22 | 478ms | 21ms | 3ms | 98ms | 3ms-99ms |
| Maps::CarsController#index.JSON | 22 | 96ms | 4ms | 1ms | 29ms | 0ms-29ms |
| StaticPagesController#home.\*/\* | 21 | 731ms | 34ms | 12ms | 142ms | 12ms-143ms |
| #. | 16 | 1.29s | 80ms | 0ms | 609ms | 0ms-612ms |
| UsersController#show.HTML | 15 | 5.36s | 357ms | 125ms | 765ms | 124ms-767ms |
| CarsController#info.JSON | 11 | 9.23s | 838ms | 39ms | 1.99s | 38ms-2.02s |
| Maps::LocationsController#show.JSON | 10 | 58ms | 5ms | 1ms | 13ms | 1ms-13ms |
| StaticPagesController#about.\*/\* | 6 | 191ms | 31ms | 14ms | 63ms | 14ms-64ms |
| StaticPagesController#faq.\*/\* | 2 | 51ms | 25ms | 24ms | 26ms | 24ms-26ms |
| Admin::DashboardController#index.HTML | 2 | 264ms | 132ms | 113ms | 150ms | 112ms-153ms |
| StaticPagesController#about.HTML | 1 | 23ms | 23ms | 23ms | 23ms | 23ms-24ms |
| Maps::TracksController#info\_all.JS | 1 | 45ms | 45ms | 45ms | 45ms | 45ms-47ms |
| Admin::UsersController#show.HTML | 1 | 28ms | 28ms | 28ms | 28ms | 27ms-28ms |
| Admin::TicketsController#index.HTML | 1 | 55ms | 55ms | 55ms | 55ms | 55ms-57ms |

Б.10. Трудоемкость обращений к БД (по средним значениям)

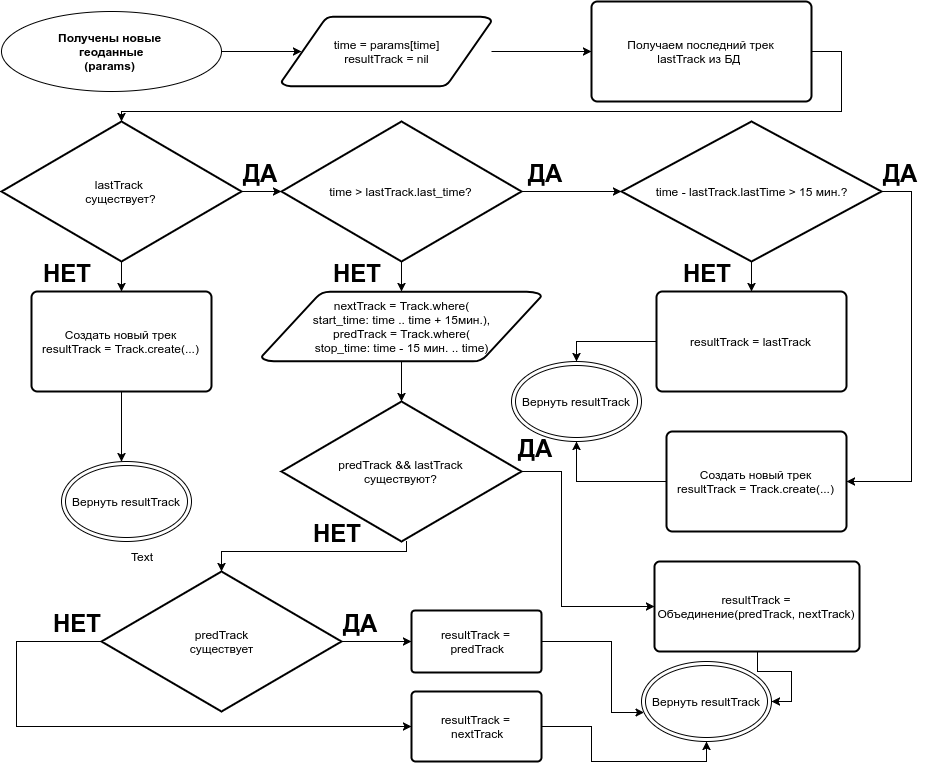
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод контроллера | Число запросов | Общее время | Среднее время | Минимальное время | Максимальное время | Статистическая оценка |
| Api::V1::GeodataController#recive.JSON | 959 | 18.49s | 19ms | 5ms | 270ms | 8ms-48ms |
| UsersController#charts\_controller.JSON | 54 | 34.55s | 639ms | 20ms | 4.01s | 28ms-3.44s |
| StaticPagesController#home.HTML | 41 | 1.34s | 32ms | 9ms | 161ms | 9ms-163ms |
| Maps::TracksController#info.JS | 37 | 1.11s | 30ms | 5ms | 87ms | 6ms-88ms |
| Maps::LocationsController#index.JSON | 22 | 478ms | 21ms | 3ms | 98ms | 3ms-99ms |
| Maps::TracksController#index.JSON | 22 | 411ms | 18ms | 3ms | 183ms | 3ms-185ms |
| StaticPagesController#home.\*/\* | 21 | 731ms | 34ms | 12ms | 142ms | 12ms-143ms |
| #. | 16 | 1.29s | 80ms | 0ms | 609ms | 0ms-612ms |
| UsersController#show.HTML | 15 | 5.36s | 357ms | 125ms | 765ms | 124ms-767ms |
| CarsController#info.JSON | 11 | 9.23s | 838ms | 39ms | 1.99s | 38ms-2.02s |
| StaticPagesController#about.\*/\* | 6 | 191ms | 31ms | 14ms | 63ms | 14ms-64ms |
| Admin::DashboardController#index.HTML | 2 | 264ms | 132ms | 113ms | 150ms | 112ms-153ms |
| StaticPagesController#faq.\*/\* | 2 | 51ms | 25ms | 24ms | 26ms | 24ms-26ms |
| Admin::TicketsController#index.HTML | 1 | 55ms | 55ms | 55ms | 55ms | 55ms-57ms |
| Maps::TracksController#info\_all.JS | 1 | 45ms | 45ms | 45ms | 45ms | 45ms-47ms |
| Admin::UsersController#show.HTML | 1 | 28ms | 28ms | 28ms | 28ms | 27ms-28ms |
| StaticPagesController#about.HTML | 1 | 23ms | 23ms | 23ms | 23ms | 23ms-24ms |
| Admin::ExceptionsController#show.HTML | 1 | 20ms | 20ms | 20ms | 20ms | 20ms-20ms |
| UsersController#reset\_password.\*/\* | 1 | 18ms | 18ms | 18ms | 18ms | 17ms-18ms |
| SessionsController#new.HTML | 1 | 17ms | 17ms | 17ms | 17ms | 17ms-18ms |

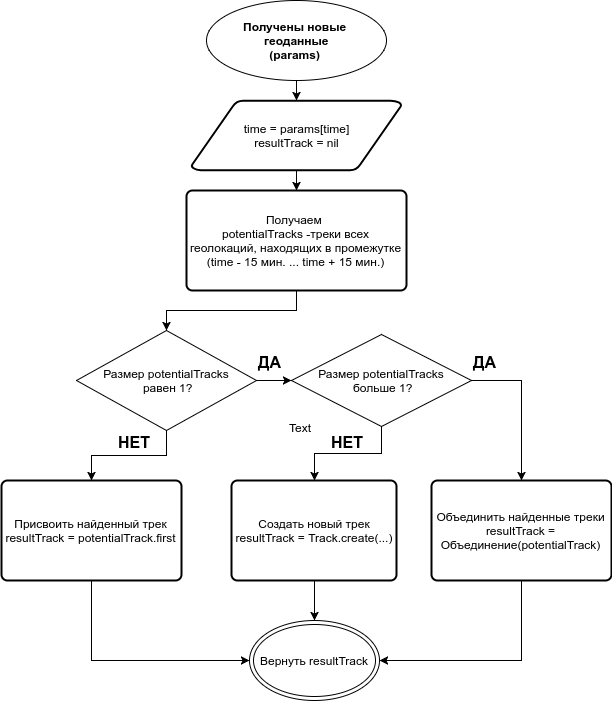
Приложение В

**Алгоритм получения новых геоданных и распределение их по маршрутам**

Приложение Г

**Г.1. Алгоритм поиска целевого трека до оптимизации.**

****

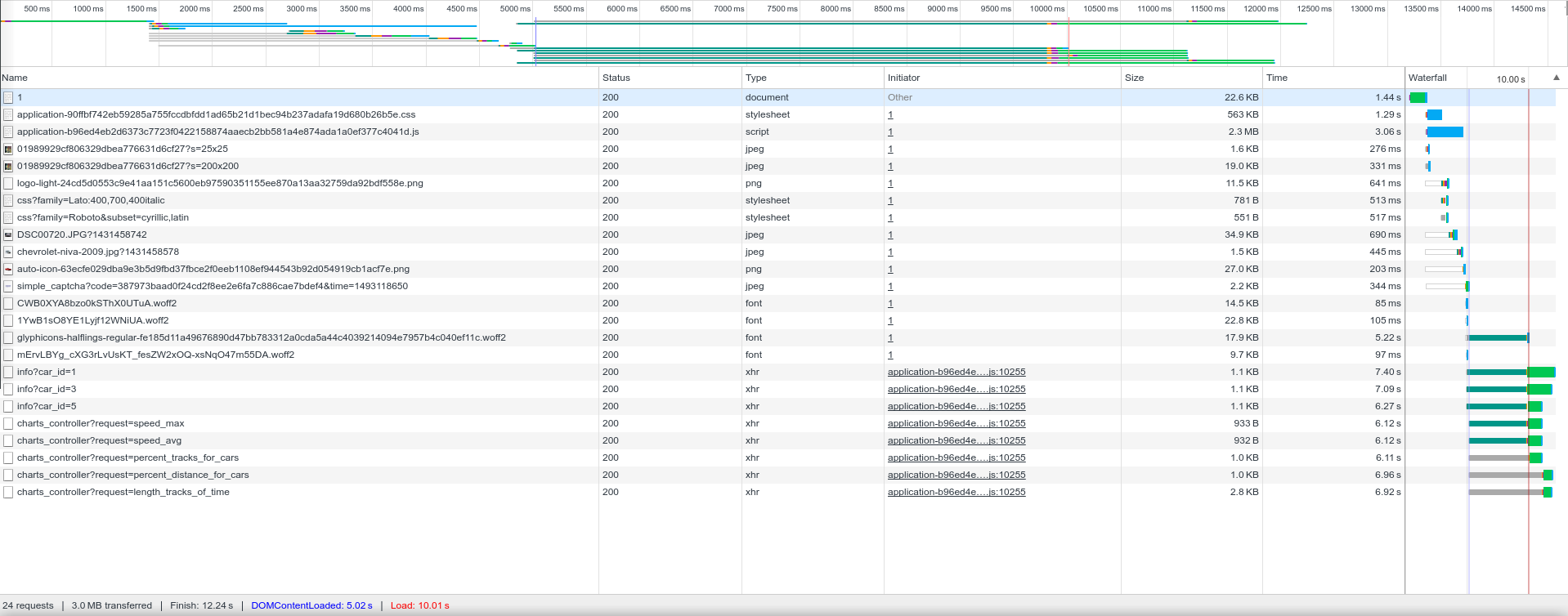
**Г.2. Алгоритм поиска целевого трека после оптимизации.**

Г.3. Алгоритм объединение треков.

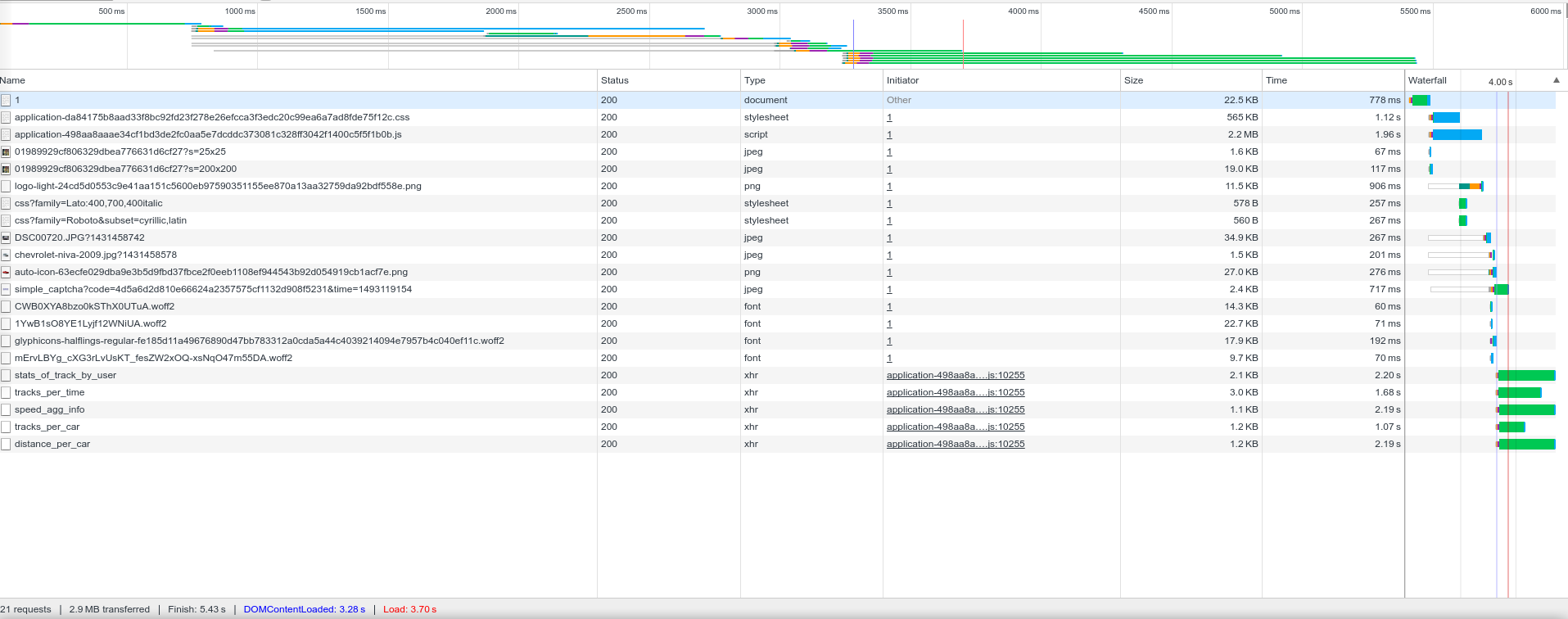


Приложение Д

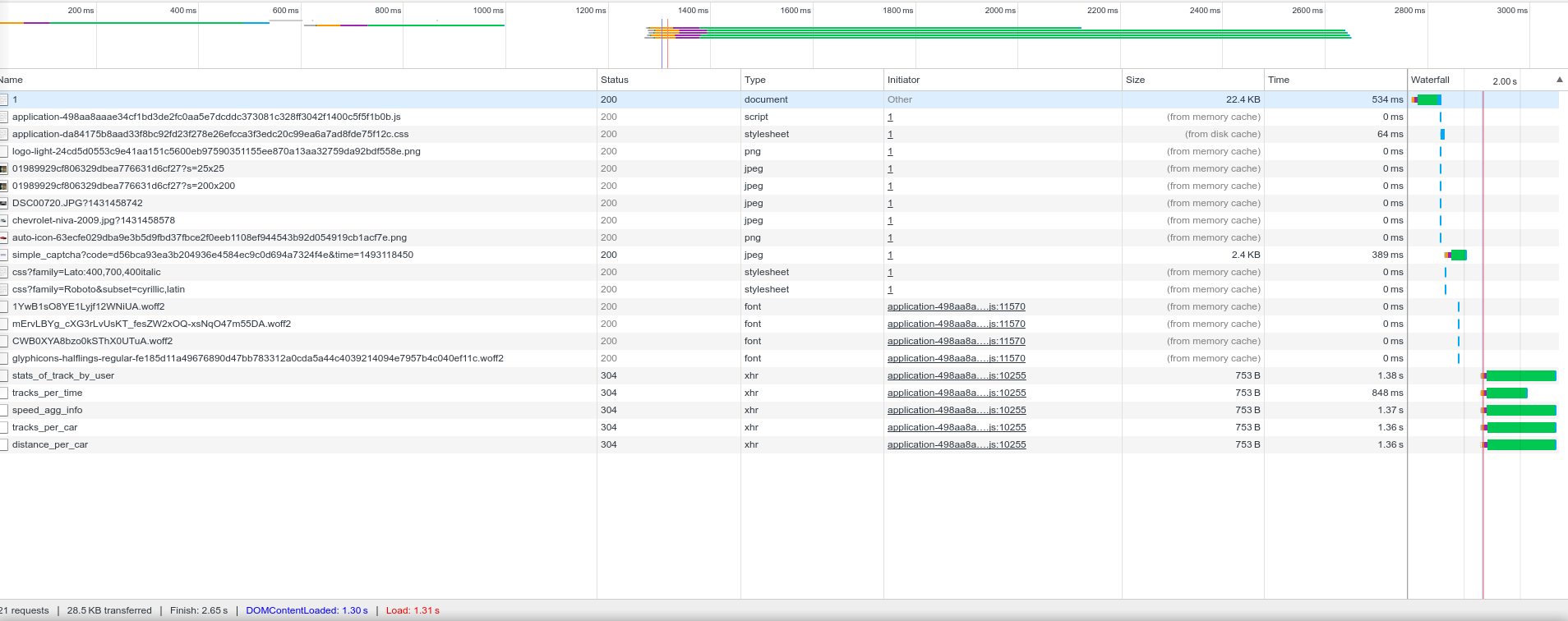
Д.1. Загрузка страницы ”Профиль пользователя” до оптимизации серверной части системы.



Д.2. Загрузка страницы ”Профиль пользователя” после оптимизации серверной части системы.

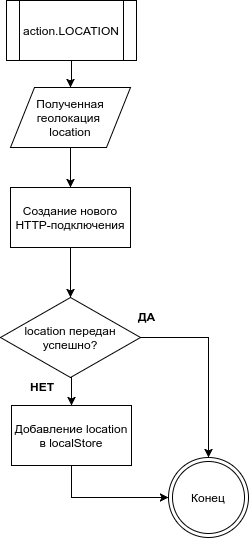


Д.3. Загрузка страницы ”Профиль пользователя” после оптимизации серверной части системы с включённым кешированием статических частей.

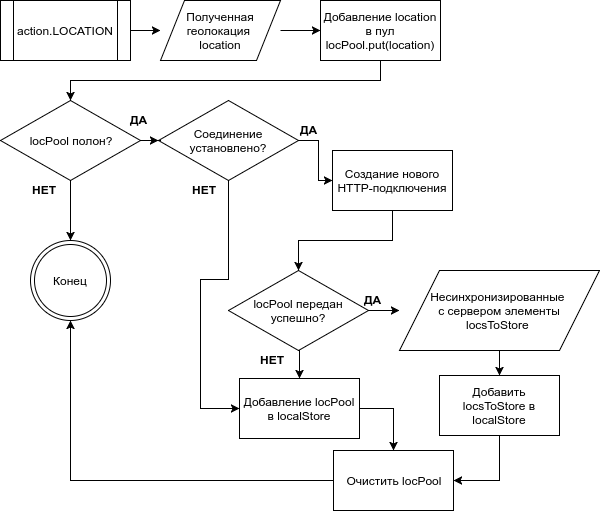


Приложение Е

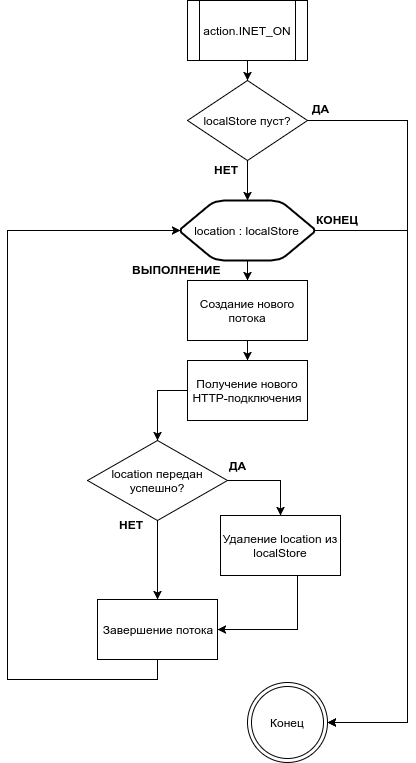
**Е.1. Алгоритм передачи геоданных до оптимизации.**



**Е.2. Алгоритм передачи геоданных после оптимизации.**



**Е.3. Алгоритм синхронизации геоданных до оптимизации.**

****

**Е.4. Алгоритм синхронизации геоданных после оптимизации.**

