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](#_Toc479256741)

[2. Постановка задач 4](#_Toc479256742)

[3. Описание системы 5](#_Toc479256743)

[3.1 Описание объектов исследования. 5](#_Toc479256744)

[3.2 Анализ востребованности системы. 6](#_Toc479256745)

[3.3 Основные технологии построения систем мониторинга 8](#_Toc479256746)

[3.4 Описание существующей системы “Coordinate" 9](#_Toc479256747)

[4. Исследование системы “Coordinate” 12](#_Toc479256748)

[4.1 Методология исследования 12](#_Toc479256749)

[4.2 Анализ решения хранения геоданных в системе “Coordinate” 13](#_Toc479256750)

[4.3 Анализ решения обработки геоданных в системе “Coordinate” 26](#_Toc479256751)

[4.4 Анализ решения отображения геоданных в системе “Coordinate” 28](#_Toc479256752)

[4.5 Анализ решения сбора геоданных в системе “Coordinate” 28](#_Toc479256753)

[4.6 Сравнение системы “Coordinate” с другими системами мониторинга 28](#_Toc479256754)

[5. Экономический анализ системы 29](#_Toc479256755)

[5.1 Расчет экономических затрат 29](#_Toc479256756)

[5.2 Расчет экономической рентабельности 29](#_Toc479256757)

[6. Заключительная часть 29](#_Toc479256758)

[6.1 Результаты исследования 29](#_Toc479256759)

[6.2 Заключение 29](#_Toc479256760)

[7. Список сокращений и условных обозначений 30](#_Toc479256761)

[8. Список литературы 30](#_Toc479256762)

[Приложение А 31](#_Toc479256763)

# 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.

## 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 3.1 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 4.1. Измерение свойств объекта А

Измерение модели объекта

*F (А′)*

Измерение объекта

*F (А)*

Сравнительный анализ *C(F(A), F(А′))*

Рисунок 4.2. 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 4.3. Сравнение синтеза свойств объекта и модели

## Анализ решения 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 4.4. 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 4.1. 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 4.2. 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 4.3. Результаты расчета зависимости роста записей

Рисунок 4.6. Гистограмма, изображающая зависимость роста данных

Рисунок 4.7. 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”

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | В байтах | В удобночитаемом виде | Байт на строку |
| Размер данных | 35790848 | 34 MB | 305 |
| Карта видимости | 0 | 0 bytes | 0 |
| Свободное пространство | 32768 | 32 kB | 0 |
| Общий размер, включая TOAST | 35831808 | 34 MB | 306 |
| Размер индексов | 12222464 | 12 MB | 104 |
| Общий размер, включая TOAST и индексы | 48054272 | 46 MB | 410 |
| Кол-во используемых строк в текстовом представлении | 35190836 | 34 MB | 300 |
| Число строк | 117048 |  | |
| Число действующих кортежей | 117048 |
| Число недействующих кортежей | 1245 |

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 4.5. 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.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 4.6 Информация о таблице “locations”

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.

## Анализ решения 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 4.7. Перечень контроллеров WEB-сервиса “Coordinate”

Исходя из анализа журналов, можно говорить, что наиболее нагруженным является метод ″Api::V1::GeodataController#recive″. Алгоритм его работы приведен в приложении В.

## Анализ решения отображения геоданных в системе “Coordinate”

Тут о том, как данные отображаются, как по ним вычисляется статистика и т.п.

## Анализ решения сбора геоданных в системе “Coordinate”

Тут о том, как данные отображаются, как получаются, формат передачи данных на сервер и т.п. Расписать про JSON API

## Сравнение системы “Coordinate” с другими системами мониторинга

Тут таблица, что лучше, что хуже.

# 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

Подводим результаты аналитического исследования.

## Заключение

Делаем вывод: насколько оправдано использованные в проекте решения, необходимость модернизации, и т.п.

# Список сокращений и условных обозначений

# 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**

Приложение В

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