**АНАЛИЗ ТЕХНОЛОГИИ ПОСТРОЕНИЯ СИСТЕМЫ ОБРАБОТКИ ГЕОДАННЫХ НА ПРИМЕРЕ WEB-СЕРВИСА «COORDINATE».**

1. Добрый день уважаемые члены государственной комиссии, уважаемый председатель! В дипломной работе передо мной стояла задача провести анализ технологии построения системы обработки геоданных, используемой в Web-сервисе мониторинга автотранспорта “Coordinate” и модернизировать систему, в соответствии с результатами анализа.

2. Система “Coordinate” была разработана мною в рамках ВКР бакалавра и развернута в общем доступе с марта 2015г. Система предназначена для мониторинга перемещения личного транспорта пользователей с предоставлением статистической информацией. Для управления системой предусмотрен функционал, позволяющий администрировать компоненты системы через WEB – интерфейс.

3.В ходе эксплуатации был зафиксирован ряд проблем и ошибок, негативно сказывающихся на оптимальную и безотказную работу с системой. Таким образом, основной целью данной работы стал анализ применённых при построении системы и ее компонент решений, а также их оптимизация с целью повышения надежности, отказоустойчивости и быстродействия системы в целом.

4. Прежде чем перейти к анализу системы, необходимо рассказать об архитектуре сервиса. Сервис представляет собой полностековое WEB-приложение, состоящее из нескольких основных компонент. Основная часть бизнес-логики сервиса содержится в серверной части системы, представляющее собой MVC-приложение, написанное на языке ruby с использование фреймворка Ruby on Rails. Данные хранятся в БД, под управлением PostgreSQL. Логика получения необходимых геоданных сосредоточена в мобильном приложении-трекере, представляющее собой приложение для мобильных устройств, под управление ОС Android. Пользовательский интерфейс системы реализован в виде Web-сайта. Для этих двух компонент системы, в серверной части были реализованы программные интерфейсы, через которые и происходит коммуникация с серверной частью. Все каналы связи защищены SSL-шифрованием, для предотвращения перехвата персональных данных.

5. В рамках анализа системы был проведен ряд мероприятий, включающий в себя разборы журналов, измерения времени исполнения, сбор статистики, а также проведен опрос, среди потенциальных пользователей системы, с целью изучения их потребностей и их удовлетворения в ходе модернизации системы. По результатам анализа был выделен ряд проблем в каждом из компоненте системы, исправление которой необходимо для повышения оптимальной и безотказной работы системы в целом.

6. Одним из ключевым этапом анализа и модернизации системы является анализ алгоритмической базы. На слайде представлены основные алгоритмы, совокупность которых образуют основу бизнес-логики системы. Алгоритмы фильтрации и синхронизации реализованы в мобильном приложении-трекере и представляют собой логику коммуникаций с сервером. Алгоритм распределения геоданных выполняется на сервере, при получении новых геоданных и сохранении их в систему. Первоначальная реализация этих алгоритмов была чересчур громоздкой и медленной, кроме того, особенности реализации алгоритмов в мобильном приложении-трекере были чересчур затратной по ресурсам, в частности, по электропотребелению.

7. На данном слайде представлен принцип работы алгоритма фильтрации геоданных, используемый для отсеивания геоданных, не удовлетворяющим определенным критериям, с целью уменьшения объема трафика между мобильным приложением-трекером и сервером, а также для уменьшения объема хранимых данных в БД. На первом изображении представлен трек, состоящий из геолокаций, сохраненных без использования фильтра. Основными параметрами для фильтрации геоданных служат accuracy (точность) и скорость. Получив новые геоданные, происходит проверка на допустимое отдаление от последней сохраненной координаты, основываясь на параметр accuracy. Затем, происходит отсеивание координат, имеющих нулевую скорость. Результатом выполнения становится географический центр полученного полигона. Такая предобработка позволила не только снизить разброс и сгладить линию трека, но и уменьшить количество “мусорных” данных.

8. На следующем слайде представлен алгоритм передачи геоданных, сохранённых в локальное хранилище. Данные попадают в локальное хранилище в случае ошибок при передачи или сохранении геоданных. Первоначальная реализация была реализована таким образом, что при получении новых геоданных, открывалось новое соединение и данные отправлялись на сервер. Такой подход не только был затратен по энергопотреблению, так еще и не гарантировал полную целостность передачи данных, хоть и вероятность сбоя была мала. В случае появления ошибок передачи, или потери соединения, данные сохранялись в локальное хранилище.

Концептуально, модернизация алгоритма была изменена полностью. Во-первых, был реализован пул соединений, что означает что в любой момент времени число открытых соединение не превышает константное число. В случае, если все соединения заняты, передача данных откладывается, до освобождения какого-либо соединения. Кроме того, при передачи данных на сервер добавлена проверка наличия текущего Интернет-соединения, если оно отсутствует, данные сразу направляются в локальное хранилище, без попытки передачи и выделения на это ресурсов. Так же, в новой версии алгоритма, все содержимое локального хранилище передается в одном запросе. Такой подход ненамного усложнил логику серверной компоненты, но заметно увеличил время автономной работы мобильного приложения-трекера.

9. Следующим проанализированным алгоритмом служит алгоритм распределения геоданных по маршрутам. Для сохранении геоданных в БД, необходимо определить, к какому именно маршруту относится сохраняемые геоданные. На слайде представлены основные варианты распределения геоданных по трекам, обрабатываемы алгоритмом. Основным критерием для поиска потенциальных треков служит временной диапазон в размере 15 минут. <АЛГОРИТМ>. По результатам эксплуатации и анализа, в работе данного алгоритма был выявлен ряд проблем и ошибок, а также недостаточное скорость выполнения и избыточное обращение к БД. Основная особенность первоначального алгоритма заключалась в том, что поиск потенциальных треков происходил, на основе параметров треков (время начала, время конца), что заметно влияло на сложность алгоритма и являлось источником множества ошибок.

10. На слайде представлена оптимизация алгоритма, лишённая основных недостатков предыдущей версии. Основное отличие заключается в том, что теперь выборка потенциальных треков по критериям происходит не по параметрам трека (дата начало/конца), а по геолокациям, расположенным в конкретном временном диапазоне (15 минут). <АЛГОРИТМ>. В результате, заметно уменьшилось количество обращений к БД, снизилась сложность алгоритма, увеличилась скорость выполнения, а главное, алгоритм стал более устойчивый к коллизия, а код его реализации более поддерживаемым.

11. Анализ БД основывался на обработке журналов сервера и системной статистики, ведомой СУБД PostgreSQL. За время эксплуатации был обнаружен высокий рост данных, выражаемый в экспоненциальной зависимости от числа пользователей системы, что привело к увеличению потребления ресурсов сервера. Благодаря изменению формата хранения некоторых таблиц и реорганизации межтабличных отношений, удалось незначительно снизить рост данных и уменьшить общий объем БД.

Еще одним слабым местом системы были медленные запросы к БД, блокирующие серверные процессы. Такими запросами оказывались либо запросы, сгенерированные системой ORM, либо запросы, выполняющие статистические расчеты над геоданными. План построения таких запросов был тщательно проанализирован и в таблицы были добавлены необходимые индексы для ускорения поиска данных, а наиболее медленные запросы были переработаны в более оптимальную форму.

12. Пользовательский интерфейс системы подразумевает собой адаптивный Web-сайт с простым и понятным интерфейсом и поддержкой русской и английской локализации. В ходе анализа системы были проведены замеры построения различных страниц WEB-сайта, с целью сбора информации о времени, занимаемой для полной загрузки интерфейса. По результатам анализа, было принято решение использовать частичное (порционное) построение страницы, а данные загружать асинхронными запросами и по запросу пользователю в тех местах, где это возможно. Кроме того особое внимание было уделено анализу безопасности приложения.

13. На следующем слайде представлены основные сервисы, предоставляющие картографические решения, в виде карточных тайлов, рассматриваемые мною при анализе WEB-сайта сервиса. Слева представлен свободный и полностью бесплатный проект OSM, использующий в качестве API свободную JS библиотеку Leaflet. OSM не владеет широким функционалом, что прибавило бы сложностей при реализации задачи. Справа представлены карты от Google maps — коммерческий проект, бесплатный для некоммерческого использования и техподдержкой. Этот сервис имеет свой API для интерактивных карт, что повлияло бы на расширяемость проекта и привело бы к потери универсальности. Кроме того, сервис накладывает ограничения на количество показов, при свободном использовании.

Mapbox, размещенный в центре, объединяет достоинства данных сервисом, и лишен практически всех недостатков. Это коммерческий проект, бесплатный для некоммерческого использования, с крупным сообществом и широким функционалом, что во много упростило использование библиотеки в целом. Кроме того, тайлы от MapBox основаны на OSM, а значит переход на OSM не потребует изменений в архитектуре сервиса.

14. При модернизации сервиса был использован широкий набор библиотек и технологий, часть которых представлена на слайде. В разработке активно использовалась система контроля версий Git и редактор кода Sublime Text. В рамках модернизации системы, все используемые библиотеки были обновлены до последних версий.

15. В заключении, хотелось бы предъявить результаты модернизации системы, выполненной в ходе анализа в рамках выпускной работы. Оптимизация запросов и и изменения формата хранения данных увеличили производительность работы с БД почти на 20%. Оптимизация алгоритмов, используемых для сбора и обработки геоданных позволило увеличить производительность системы и сократить объем передаваемого трафика на 40%. Реализация порционной отрисовки Web-сайта, в совокупности с асинхронной загрузкой позволило ускорить время построения пользовательского интерфейса на 20%.

Таким образом, можно говорить о том, что модернизация системы повысила надежность и удобство пользования системой для конечных пользователей и удовлетворяет их потребностям, выявленных в рамках проведенного опроса.