Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

Факультет программной инженерии и компьютерной техники
(название факультета)

Кафедра информатики и прикладной математики
(название кафедры)

Направление подготовки (специальность) 09.04.01

О Т Ч Ё Т

о производственной (преддипломной) **практике**
(название практики)

Тема задания «Разработка архитектуры клиентской части веб-приложения для работы с устройствами Emlid Reach и Emlid Reach RS»

Студент Кузнецов Андрей Андреевич **Группа №** P4215
(Фамилия, Имя, Отчество)

Руководитель практики от организации Фёдоров Е.М., ИП Николаев Д.А., руководитель отдела разработки программного обеспечения
(Фамилия И.О., должность и место работы)

Руководитель практики от университета Исаев И.В., ассистент
(Фамилия И.О., должность)

Практика пройдена с оценкой _____

Подписи членов комиссии

_____ _____
(подпись) (Фамилия И.О.)

_____ _____
(подпись) (Фамилия И.О.)

_____ _____
(подпись) (Фамилия И.О.)

Дата «____» _____ 20____ г.

Санкт-Петербург
2018 г.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

С 30 апреля по 3 июня 2018 года обучающийся проходил производственную практику в ИП Николаев Денис Александрович. На практику было дано задание по разработке архитектуры веб-приложения для управления ГНСС-приёмниками, работающими под управлением программного обеспечения, основанного на программном комплексе высокоточного позиционирования RTKLIB.

В процессе прохождения практики были изучены следующие электронные источники и литература:

- документация программного комплекса RTKLIB;
- документация устройств Emlid Reach и Emlid Reach RS;
- техническое задание на разработку программного модуля.

2 ХОД РАБОТЫ

2.1 Этап 1 – Платформа для разработки

Платформой для разработки являлись устройства Emlid Reach и Emlid Reach RS. Данные ГНСС-приёмники работают под управлением программного обеспечения, основанного на программном комплексе высокоточного позиционирования RTKLIB. Работа пользователя с данными продуктами осуществляется через веб-приложение, доступ к которому можно получить с помощью любого устройства, на котором установлен веб-браузер.

Reach и Reach RS созданы на базе одного и того же вычислительного модуля и используют одинаковые приёмники u-blox, имеется ряд существенных различий в аппаратном обеспечении данных устройств. Различия Reach и Reach RS, которые необходимо учесть при создании архитектуры веб-приложения, указаны в таблице 2.1.

Таблица 2.1 – Различия Reach и Reach RS

Техническая особенность	**Reach**	**Reach RS**
Встроенная батарея	нет	да
Встроенная антенна	нет	да
Встроенное радио	нет	да
Физическая кнопка на корпусе	нет	да
Возможность управления фотокамерой	да	нет

2.2 Этап 2 – Постановка задачи работы

Основной задачей производственной практики являлось создание архитектуры веб-приложения, необходимого для осуществления взаимодействия пользователя с вышеупомянутыми ГНСС-приёмниками.

2.3 Этап 3 – Проектирование приложения

2.3.1 Основные задачи и требования

Для создания общей архитектуры приложения были сформулированы наиболее важные идеи, задачи и требования, предъявляемые к разрабатываемому продукту.

Важно отметить, что в следующем далее списке присутствуют пункты, которые касаются лишь разработчиков программных компонентов, отвечающих за взаимодействие с RTKLIB, системными утилитами и различными аппаратными компонентами Reach и Reach RS. Данные модули и детали их разработки выходят за рамки рассматриваемой работы, но являются важными для понимания общей структуры приложения.

Разрабатываемое приложение должно:

– **Осуществлять запуск приложений RTKLIB в управляемых контейнерах**. Надёжным и простым способом организации взаимодействия с приложениями RTKLIB является их запуск в управляемых контейнерах. При дан-

ном подходе необходимые утилиты будут работать так же, как если бы пользователь запускал их в терминале.

Используя подобный подход, становится возможно организовать взаимодействие веб-приложения с необходимыми программами таким образом, что:

1) не требуется вмешательство в исходный код RTKLIB;

2) облегчается поддержка совместимости с новыми версиями программного комплекса;

3) разрабатываемый продукт остаётся независим от изменений в кодовой базе RTKLIB.

– **Иметь клиентскую часть, представленную в виде одностраничного приложения**. Как показал проведённый обзор, данный подход в настоящее время является одним из самых популярных решений для создания веб-интерфейсов, предназначенных для управления такими устройствами, как Reach или Reach RS.

– **Поддерживать передачу данных по протоколу WebSocket**. Для создания отзывчивого интерфейса веб-приложения и обеспечения лучшего пользовательского опыта бо̲льшую часть взаимодействий клиентской части с сервером следует организовать с помощью асинхронных запросов и сообщений.

Наиболее удачным решение для разрабатываемого приложения является протокол WebSocket. Создавая на клиентской части веб-приложения слушателей определённых событий, становится возможно организовать отображение различной информации в режиме реального времени без постоянных опросов сервера.

2.3.2 Основные модули

Перечислим основные модули приложения (рис. 2.1):

а) Серверная часть

1) WebSocket сервер

2) Модуль взаимодействия с RTKLIB

3) Демоны и сервисы для взаимодействия с аппаратными компонентами устройства

б) Клиентская часть

1) WebSocket клиент

2) JavaScript-приложение

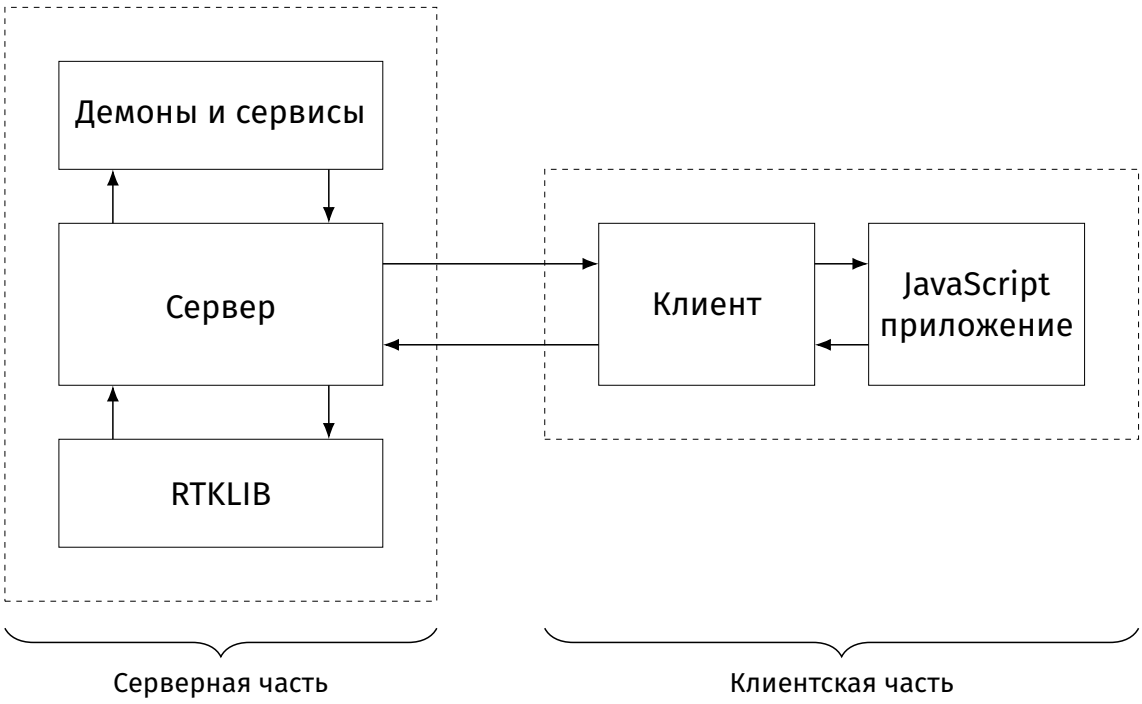

Рисунок 2.1 – Общая архитектура приложения

2.3.3 Выбор инструментов разработки

2.3.3.1 Серверная часть приложения

В рамках разработки описанного приложения выбор инструментов для написания серверной части приложения зависит не только от возможностей веб-фреймворков, доступных для того или иного языка программирования, но и от задач, описанных в пункте 2.3.1. Также, важно помнить, что разрабатываемое приложение предназначено для запуска на устройствах с ограни-

ченными вычислительными ресурсами.

Разработчиками серверной части приложения был проведён обзор высокоуровневых языков программирования, часто используемых для разработки веб-приложений. Основными из рассмотренных вариантов были языки Java, Python и Ruby.

На основании различных оценок, детали формирования которых выходят за рамки рассматриваемой работы, основным языком для серверной части приложения был выбран Python.

Одним из основных инструментов при написании серверной части приложения был выбран веб-фреймворк Flask. Запуск приложений RTKLIB в управляемых контейнерах осуществляется с помощью модуля rexpect.

2.3.3.2 Клиентская часть приложения

Основу клиентской части приложения составляют: язык гипертекстовой разметки HTML, CSS-стили и скрипты на языке JavaScript.

Для наиболее быстрой и удобной разработки одностраничного приложения было решено использовать специализированную библиотеку или фреймворк. На момент проектирования приложения одними из самых популярных инструментов для создания одностраничных приложений являлись React, Angular 2 и Vue.js.

Для анализа и сравнения вышеперечисленных библиотек и фреймворков были изучены:

- официальные документации;
- исходные коды простейших приложений и их компонентов;
- результаты синтетических тестов.

По результатам изучения возможных решений для разработки приложения был выбран фреймворк Vue.js. Данный выбор также был обусловлен наличием у автора данной работы опыта разработки с использованием фреймворка AngularJS, который является одним из источников вдохновения для со-

здателей Vue.js.

Также, вместе с Vue.js было решено использовать библиотеку-расширение для данного фреймворка – Vuex. Данная библиотека позволяет создать централизованное хранилище данных, доступ к которому будут иметь все компоненты приложения. Одной из важнейших особенностей работы с таким хранилищем является способность компонентов Vue.js реактивно обновлять своё состояние при изменении тех или иных данных в Vuex.

2.4 Детализированная архитектура приложения

С учётом инструментов, выбранных в предыдущем подразделе, уточним архитектуру приложения, представленную ранее (рис. 2.1). На рисунке 2.2 изображена детализированная архитектура разрабатываемого приложения.

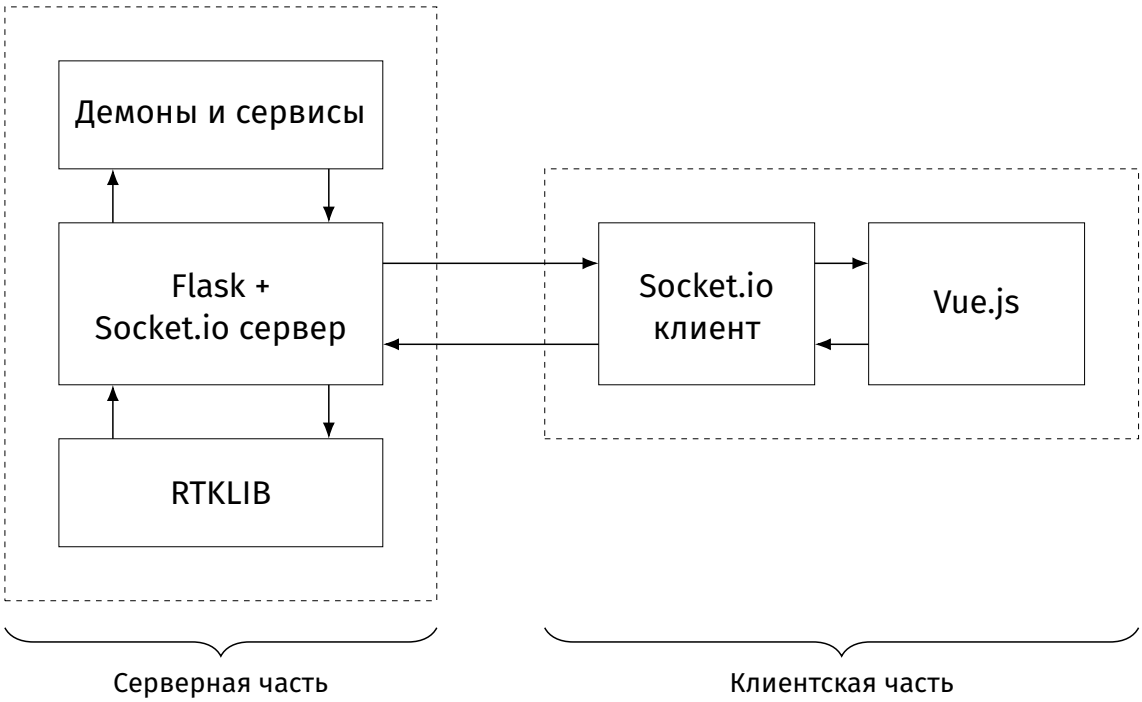

Рисунок 2.2 – Детализированная архитектура приложения

2.4.1 Подробная архитектура клиентской части приложения

Клиентская часть разрабатываемого приложения представляет из себя совокупность отдельных программных модулей и глобального хранилища, изменения данных в котором вызывают реактивное обновление модулей, имеющих соответствующие слушатели событий.

Модули приложения можно условно разделить на две группы:

– модули, являющиеся *моделями представления* для соответствующих экранов приложения;

– модули общего назначения, предназначенные для обработки событий, приходящих от сервера, или содержащие различные утилиты.

На рисунке 2.3 изображены все основные компоненты приложения и их связи с глобальным хранилищем. К модулям общего назначения относятся только модуль обработки событий и модуль работы с картами, остальные компоненты написаны с помощью Vue.js и связаны с представлениями приложения.

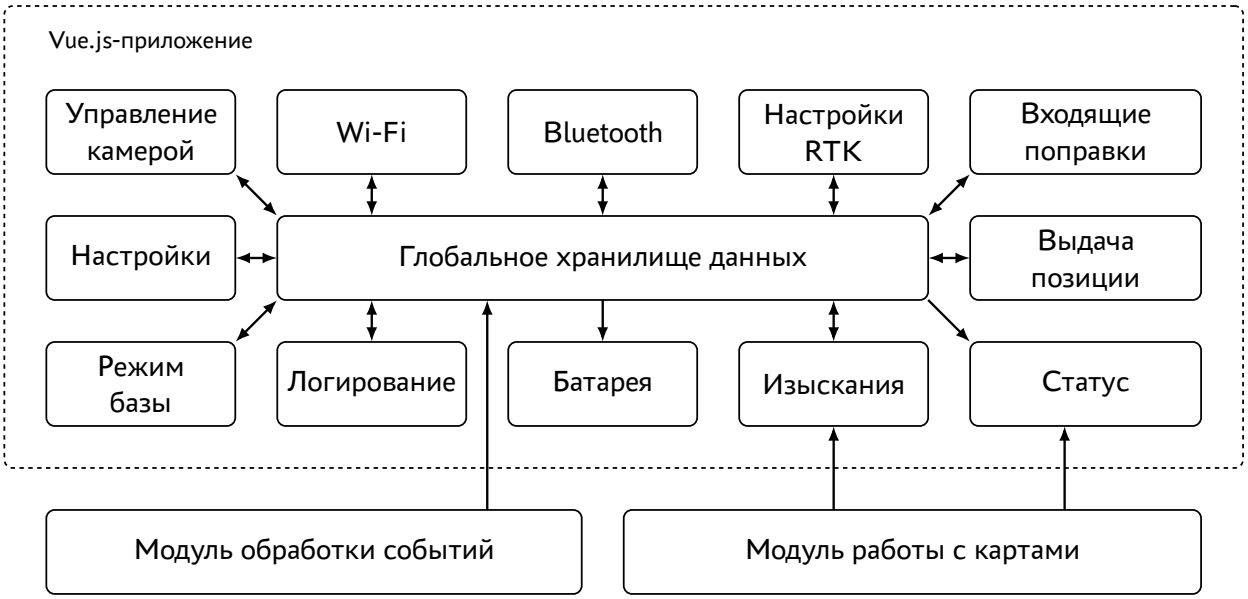

Рисунок 2.3 – Архитектура клиентской части приложения

3 РЕЗУЛЬТАТЫ РАБОТЫ

– Проведён обзор платформы для разработки приложения. По результатам обзора используемых устройств были выявлены основные технические особенности, которые необходимо было учесть при разработке клиентского приложения.

– На основании ключевых требований к итоговому приложению и с учётом особенностей программного обеспечения используемых устройств, была создана архитектура проекта, а также были выбраны средства для его реализации.

– Была произведена детализация архитектуры клиентской части приложения, в результате чего были выделены её ключевые модули.