TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Системная инженерия программного обеспечения
Отделение информационных технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Разработка фреймворка для создания оптимизированной клиентской части веб-сайтов компаний

УДК 004.415..2:004.455.1:004.744

Студент

Группа	ФИО	Подпись	Дата
8ИМ91	Ткачев Михаил		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН	Верховская Марина Витальевна	к.э.н		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД	Федоренко Ольга Юрьевна	д.м.н		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н		

Томск – 2021 г.

Планируемые результаты обучения

Код результатов	**Результаты обучения (выпускник должен быть готов)**	**Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)**
Р1	Воспринимать и самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте.	Требования ФГОС ВО (3++) (ОПК-1,2; ПК-1; УК-1,4,6), критерий 5 АИОР (п. 1.1), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р2	Владеть и применять методы и средства получения, хранения, переработки и представления информации посредством современных компьютерных технологий, в том числе в глобальных компьютерных сетях.	Требования ФГОС ВО (3++) (ОПК-1,2,6,7; ПК-1,2,3,5,10; УК-1), критерий 5 АИОР (п. 1.1, 1.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р3	Демонстрировать способность выстраивать логику рассуждений и высказываний, основанных на интерпретации данных, интегрированных из разных областей науки и техники, выносить суждения на основании неполных данных, анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями.	Требования ФГОС ВО (3++) (ОПК-1,3,6; ПК-5,6; УК-1,6), критерий 5 АИОР (п. 1.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р4	Анализировать и оценивать уровни своих компетенций в сочетании со способностью и готовностью к саморегулированию дальнейшего образования и профессиональной мобильности. Демонстрировать способность к самостоятельному обучению новым методам исследования, способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения.	Требования ФГОС ВО (3++) (ОПК-1,4,6; УК-6), критерий 5 АИОР (п. 1.6, п. 2.2,2.6.), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

Код результатов	Результаты обучения (выпускник должен быть готов)	Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)
Р5	Владеть современными коммуникативными технологиями, в том числе на иностранном языке для академического и профессионального взаимодействия. Владеть, по крайней мере, одним из иностранных языков на уровне социального и профессионального общения, применять специальную лексику и профессиональную терминологию языка.	Требования ФГОС ВО (3++) (ОПК-1,3; УК-3,4,5; ПК-7,8,9). Запросы студентов, отечественных и зарубежных работодателей.
Р6	Использовать на практике умения и навыки в организации исследовательских, проектных работ и профессиональной эксплуатации современных программных и информационных систем, в управлении коллективом. Способность организовывать и эффектно руководить работой команды проекта при разработке программных и информационных систем.	Требования ФГОС ВО (3++) (УК-2,3,5; ПК-5,6,7,8,11; ОПК1,8), критерий 5 АИОР (п. 2.1, п. 2.3, п. 1.5), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р7	Разрабатывать стратегии проектирования, критерии эффективности и ограничения применимости новых методов и средств проектирования и разработки программных систем.	Требования ФГОС ВО (3++) (УК-1,3; ПК-1,3,10; ОПК2,4,6,7), критерий 5 АИОР (п. 2.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р8	Планировать и проводить теоретические и экспериментальные (численные) исследования в области создания программных систем. Оценивать и выбирать вариант архитектуры программной/информационной системы.	Требования ФГОС ВО (3++) (ОПК-1,4,6,7; ПК-1,3,10; УК1,3), критерий 5 АИОР (п. 1.4), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р9	Владеть методами и средствами инженерии требований к системам, управления качеством программного обеспечения и системной интеграции/модернизации программного обеспечения.	Требования ФГОС ВО (3++) (УК-1; ОПК-4,5,7; ПК-1,2,4,8,11). Запросы студентов, отечественных и зарубежных работодателей.

Код результатов	Результаты обучения (выпускник должен быть готов)	Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)
Р10	Владеть современными инструментальными средствами программирования и технологиями управления данными. Использовать их при разработке требований, при проектировании и создании программного обеспечения, информационных систем/автоматизированных систем управления производством.	Требования ФГОС ВО (3++) (ПК-1,2,4,5,7,9,11; ОПК-2,5,7; УК-2). Запросы студентов, отечественных и зарубежных работодателей.
Р11	Осуществлять проектирование и разработку веб и мультимедийных приложений в среде корпоративных и глобальных информационно-телекоммуникационных систем.	Требования ФГОС ВО (3++) (ПК-1,2,3,5,6,9,11; ОПК2,4,5,7; УК-2,3,5). Запросы студентов, отечественных и зарубежных работодателей.
Р12	Осуществлять управление процессами внедрения/сопровождения (модернизации, интеграции) программных и информационных систем на основе принципов и методов системной инженерии.	Требования ФГОС ВО (3++) (ОПК-4,6,8; ПК-1,4,5,6,8,9,11; УК-2,3,4), критерий 5 АИОР (п. 2.6), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Системная инженерия программного обеспечения
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Савельев А. О.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Магистерской диссертации

Студенту:

Группа	ФИО
8ИМ91	Ткачев Михаил

Тема работы:

Разработка фреймворка для создания оптимизированной клиентской части веб-сайтов компаний	
Утверждена приказом директора (дата, номер)	№40-4/с от 09.02.2021

Срок сдачи студентом выполненной работы:	08.06.2021

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Требуется спроектировать и разработать клиентский веб фреймворк, опираясь на объектно-ориентированный подход. Затем провести тестирование производительности фреймворка и провести анализ результатов тестирования.
Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Анализ научной и технической документации, описывающей предметную область 2. Аналитический обзор актуальных веб технологий 3. Проектирование и разработка фреймворка 4. Проектирование и разработка демонстрационного приложения 5. Исследование производительности фреймворка

	6. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение. **7.** Социальная ответственность.
Перечень графического материала	Алгоритм обновления пользовательского интерфейса Диаграмма классов демонстрационного приложения SWOT-анализ

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Верховская М. В.
Социальная ответственность	Федоренко О. Ю.
Раздел на иностранном языке	Сидоренко Т.В.

Названия разделов, которые должны быть написаны на русском и иностранном языках:

Обзор актуальных фреймворков для клиентской веб разработки

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.03.2021

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Ткачев Михаил		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Системная инженерия программного обеспечения
Отделение информационных технологий

Форма представления работы:

Магистерской диссертации

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	08.06.2021

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
09.02.2021	Раздел 1. Анализ предметной области	15
10.03.2021 г	Раздел 2. Проектирование и разработка фреймворка	35
30.04.2021 г	Раздел 3. Разработка демонстрационного приложения	20
01.05.2021 г	Раздел 4. Исследование производительности демонстрационного веб приложения	10
10.05.2021 г	Раздел 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	10
20.05.2021 г	Раздел 6. Социальная ответственность	10

СОСТАВИЛ:
Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

СОГЛАСОВАНО:
Руководитель ООП

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ИМ91	Ткачев Михаил

Школа	ИШИТР	Отделение школы (НОЦ)	ОИТ
Уровень образования	Магистратура	Направление/специальность	Информационные системы и технологии

Тема ВКР:

Разработка фреймворка для создания оптимизированной клиентской части веб-сайтов компаний	
Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Использовать действующие ценники и договорные цены на потребленные материальные и информационные ресурсы, а также указанную в МУ величину тарифа на эл. энергию
2. Нормы и нормативы расходования ресурсов	-
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	Действующие ставки единого социального налога и НДС, ставка дисконтирования = 0,1 (см. МУ)
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого и инновационного потенциала НТИ	Дать характеристику существующих и потенциальных потребителей (покупателей) результатов ВКР, ожидаемых масштабов их использования
2. Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок	Построение плана-графика выполнения ВКР, составление соответствующей сметы затрат, расчет цены результата ВКР.
3. Определение ресурсной, финансовой, экономической эффективности	Оценка экономической эффективности использования результатов ВКР, характеристика других видов эффекта
Перечень графического материала	
1. *Матрица SWOT* 2. *График проведения и бюджет НТИ* 3. *Оценка ресурсной, финансовой и экономической эффективности НТИ*	

Дата выдачи задания для раздела по линейному графику	01.03.2021

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН	Верховская Марина Витальевна	к.э.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Ткачев Михаил		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8ИМ91	Ткачев Михаил

Школа	ИШИТР	Отделение школы (НОЦ)	ОИТ
Уровень образования	Магистратура	**Направление/специальность**	Информационные системы и технологии

Тема ВКР:

Разработка фреймворка для создания оптимизированной клиентской части веб-сайтов компаний	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объект исследования: процесс создания клиентской части веб-сайтов. Область применения: фреймворк используется для разработки веб-сайтов компаний. Все работы проводились в офисном помещении с помощью персонального компьютера. Площадь помещения – 25,2 $м^2$. Помещение освещалось посредством трех светильников с люминесцентными лампами.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности: 1.1 специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; 1.2 организационные мероприятия при компоновке рабочей зоны.	• ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования. • ГОСТ 22269-76. Система "Человек-машина". Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования. • ГОСТ Р 50923-96. Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения. • Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ. (ред. от 24.04.2020). • ГОСТ 12.0.003-2015 ССБТ. Опасные и вредные производственные факторы. Классификация • СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95 • СанПиН 1.2.3685-21 Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания • СанПиН 2.2.4.548–96 Физические факторы производственной среды.

	Гигиенические требования к микроклимату производственных помещений • СанПиН 2.2.4/ 2.1.8.562-96 Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки. Санитарные нормы • ГОСТ 12.1.038-82 ССБТ Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов • ГОСТ 12.1.004-91 ССБТ Пожарная безопасность. Общие требования. • ГОСТ 17.4.3.04-85 ССОП Охрана природы, почвы. Общие требования к контролю и охране от загрязнения
2. Производственная безопасность: 2.1 Анализ выявленных вредных и опасных факторов 2.2 Обоснование мероприятий по снижению воздействия	Вредные факторы: • повышенный уровень шума; • повышенная или пониженная температура воздуха рабочей зоны; • Неоптимальные показатели микроклимата; • Электромагнитное излучение; • Недостаточная или избыточная освещенность; • Психофизиологические факторы (монотонность труда, умственное и эмоциональное напряжение, перенапряжение зрительных анализаторов); • Опасные факторы: • Поражение электрическим током; • Короткое замыкание; • Статическое электричество.
3. Экологическая безопасность:	Воздействие на окружающую среду вредными и опасными отходами. Воздействие на литосферу происходит при утилизации: • компьютера и периферийных устройств (принтеры, МФУ, веб-камеры, наушники, колонки, телефоны); • люминесцентных ламп; • макулатуры.
4. Безопасность в чрезвычайных ситуациях:	Возможные ЧС: пожары и возгорания в здании, грозы, ураганы, оползни.

Дата выдачи задания для раздела по линейному графику	01.03.2021

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД	Федоренко Ольга Юрьевна	д.м.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Ткачев Михаил		

РЕФЕРАТ

Выпускная квалификационная работа содержит 107 страниц, 16 рисунков, 17 таблиц, 45 источников, 4 приложения.

Ключевые слова: фреймворк, клиентская часть веб-сайтов, JavaScript, объектно-ориентированное программирование, Webpack, HTML, DOM, Virtual DOM, одностраничные приложения, прогрессивные приложения, TypeScript, Angular, React, Vue, Node.js.

Объектом исследования данной работы является фреймворк для создания клиентской части веб-сайтов компаний.

Цель работы является создание клиентского веб-фреймворка, который обеспечит базовый необходимый функционал для синхронизации пользовательского интерфейса и состояния веб-приложения.

В процессе исследования проводились литературный анализ базовых инструментов, актуальных фреймворков современных подходов для разработки клиентской части веб-приложений. Также были сформированы функциональные и бизнес-требования к фреймворку.

В результате работы был спроектирован и разработан фреймворк для клиентской части веб-приложения Фреймворк предоставляет функционал для компонентной реализации веб-приложения с применением парадигмы объектно-ориентированного программирования и представлении пользовательского интерфейса в виде JavaScript объекта.

Область применения: фреймворк используется для создания оптимизированной и интерактивной клиентской части веб-сайтов компаний.
Преимуществами фреймворка является его безопасность и надежность, а также кастомизируемость индивидуальным разработчиком под свои нужды. Разработанный фреймворк позволяет абстрагироваться от задач обновления пользовательского интерфейса и особенностей клиентской разработки, что сделает клиентскую разработку доступнее для разработчиков со знаниями ООП языков, позволит оптимизировать бюджет компаний, не привлекая специалистов со специфичными знаниями

СПИСОК ТЕРМИНОВ, СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

DOM (англ. Document Object Model «объектная модель документа») — это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

API (англ. application programming interface) — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

JSON (англ. JavaScript Object Notation)— текстовый формат обмена данными, основанный на JavaScript.

Angular — открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

Vue — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов.

Node.js — программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Webpack — это сборщик модулей JavaScript с открытым исходным кодом.

Содержание

Введение

Основной причиной существования клиентских фреймворков в сфере веб-разработки является проблема синхронизации внутреннего состояния приложения с пользовательским интерфейсом [1]. Это происходит по причине того, что веб-приложения становятся на один уровень с нативными приложениями по степени интерактивности [2].

Регулярным алгоритмом обновления пользовательского интерфейса является цепочка действий, которая состоит из того, чтобы отреагировать на пользовательские действия, затем получить доступ к элементу объектной модели документа (DOM), который следует обновить и изменить его значение. Аналогичные действия требуются на любой пользовательский ввод [2]. Однако частое обращение к программному интерфейсу (API) DOM делают код менее читабельным и понятным, а также требует особой внимательности к оптимизации работы с данным API, так как обращение к DOM-дереву ресурсоемкий процесс. Поэтому в последнее время набрали популярность клиентские веб-фреймворки которые позволяют облегчить, оптимизировать и ускорить разработку веб-приложений [2].

Однако актуальные клиентские веб-фреймворки в лице Angular, React и Vue, которые занимают основную часть сферы, не лишены своих недостатков [1]. Наиболее важный из недостатков следует выделить то, что наиболее популярные решения, то есть фреймворки Angular и React, изначально созданы для собственных нужд компаний Google и Facebook, а не для сообщества веб-разработчиков, и их развитие направлено на покрытие необходимости компаний и полностью ими контролируется [3]. Открытость и доступность данных решений помогают данным компаниям за счет привлечения свободных разработчиков, создающих вспомогательные инструменты и развивать экосистему своего продукта. Однако у данных продуктов не стоит цель решения проблем сторонних разработчиков. Из чего вытекают риски изменения условий использования, несоответствия задач у компаний и сообщества, монополизация рынка ведущая, к извлечению выгоды и т.п [4]. Также следует отметить риски,

несущие в себе большой размер кодовой базы самих фреймворков в случае использовании их сторонними компаниями. В данном случае особо важным является вопрос надежности, безопасности и конфиденциальности. Риски в сфере безопасности связаны с большим числом зависимостей от сторонних проектов, как прямых, так и косвенных. Так согласно информации, изложенной в исследовании «State of Open Source Security— 2020», занимающейся сканированием исходного кода на уязвимости компании Snyk, более 75% уязвимостей безопасности для клиентских веб-приложений находятся в огромном числе непрямых зависимостей проекта [5]. Помимо этого, данный факт создает проблемы и в сфере надежности, так как надежность проекта начинает зависеть от огромного числа сторонних библиотек и действий большого числа разработчиков. Также, большую кодовую базу актуальных клиентских веб-фреймворков невозможно исследовать на уязвимости, несмотря на открытость исходного кода. Вместе с этим большие проекты невозможно тонко настроить под специфические нужды и невозможно самостоятельно поддерживать актуальность фреймворка. Большая кодовая база фреймворков не позволяет самому фреймворку быстро реагировать на изменение технологий и стандартов. Особо отметить следует сложность, а иногда невозможность обновления кодовой базы приложения при выходе новых версий фреймворков [6].

В следствии вышеуказанных проблем, а также отсутствия в данной сфере разработки отечественных аналогов целью данной работы является создание клиентского веб-фреймворка, который обеспечит базовый необходимый функционал для синхронизации пользовательского интерфейса и состояния веб-приложения. Разрабатываемый фреймворк создается без большого числа сторонних зависимостей, а также с применением объектно-ориентированного подхода (ООП) к разработке с использованием языка TypeScript. Данное решение позволит абстрагироваться от задач обновления пользовательского интерфейса и особенностей клиентской разработки, что сделает клиентскую разработку доступнее для разработчиков со знаниями ООП языков, позволит

оптимизировать бюджет компаний, не привлекая специалистов со специфичными знаниями. Для выполнения цели были поставлены задачи:

- Изучение архитектуры существующих фреймворков.
- Изучение проблем современных клиентских веб-фреймворков.
- Выбор методологий и паттернов проектирования для отдельных модулей.
- Создание фреймворка для веб-приложений.
- Создание демонстрационного приложения
- Исследование производительности созданного фреймворка

1. Анализ предметной области.

1.1 Базовые инструменты разработки клиентской части веб-приложений

Базовой частью веб-страницы является HTML. Язык гипертекстовой разметки HTML является стандартным языком разметки документов в браузере. Веб-браузеры получают HTML-документы с веб-сервера или из локального хранилища и преобразуют их в мультимедийные веб-страницы. HTML описывает структуру веб-страницы семантически и изначально включает подсказки для внешнего вида документа [2].

Элементы HTML — это строительные блоки HTML-страниц. С помощью конструкций HTML изображения и другие объекты, такие как интерактивные формы, могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания структурированных документов, определяя структурную семантику для текста, такого как заголовки, абзацы, списки, ссылки, цитаты и другие элементы [6]. HTML-элементы выделяются тегами, записанными с использованием угловых скобок. Такие теги, как <img /> и <input />, напрямую вводят контент на страницу. Другие теги, такие как <p>, окружают и предоставляют информацию о тексте документа и могут включать другие теги в качестве подэлементов. Браузеры не отображают HTML-теги, но используют их для интерпретации содержимого страницы [7].

HTML может встраивать программы, написанные на языке сценариев, таком как JavaScript, который влияет на поведение и содержимое веб-страниц. Включение CSS определяет внешний вид и компоновку контента. Консорциум World Wide Web (W3C), бывший разработчик HTML и нынешний разработчик стандартов CSS, с 1997 года поощряет использование CSS вместо явного презентационного HTML [8].

Каскадные таблицы стилей (CSS) – это язык таблиц стилей, используемый для описания представления документа, написанного на языке разметки, таком как HTML [7].

CSS разработан для разделения представления и содержимого, включая макет, цвета и шрифты. Данное разделение способно повысить доступность контента, внедрить гибкость и контроль свойств представления, позволяет ряду веб-страниц совместно использовать форматирование путем указания соответствующего CSS в отдельном файле .css, что снижает сложность и повторение структурного контента, а также позволяет создавать файл .css, который необходимо кэшировать, чтобы улучшить скорость загрузки страницы между страницами, которые совместно используют файл, и его форматирование [9].

JavaScript, часто сокращенно JS, является языком программирования, который соответствует спецификации ECMAScript. JavaScript – это высокоуровневый, интерпретируемый и мультипарадигмальный язык программирования. JavaScript является C-подобным объектно-ориентированным языком с динамической типизацией [10].

Наряду с HTML и CSS, JavaScript является одной из основных технологий всемирной паутины. JavaScript позволяет создавать интерактивные веб-страницы и является важной частью веб-приложений [11]. Подавляющее большинство веб-сайтов используют его для управления поведением страницы на стороне клиента, и все основные веб-браузеры имеют специальный механизм JavaScript для его выполнения [10].

Как язык с несколькими парадигмами, JavaScript поддерживает управляемые событиями, функциональные и императивные стили программирования. Он имеет интерфейсы прикладного программирования (API) для работы с текстом, датами, регулярными выражениями, стандартными структурами данных и объектной моделью документа (DOM) [11].

Изначально движки JavaScript использовались только в веб-браузерах, но теперь они встроены в некоторые серверы, обычно через Node.js. Они также встроены в различные приложения, созданные с помощью таких фреймворков, как Electron и Cordova [12].

На языке JavaScript написано множество приложений для различных сфер и потребностей. Для облегчения написания приложений на данном языке существуют разнообразные библиотеки и фреймворки. Фреймворки позволяют разработчику абстрагироваться от рутинных и долгих операций и сконцентрироваться на понимании и разработке логики приложения. Однако выбор фреймворка помимо очевидных преимуществ накладывает индивидуальные ограничения на разработку и функционал разрабатываемого приложения. Поэтому важным является выбор подходящего фреймворка для разрабатываемого приложения [13].

1.1.1Развитие современного JavaScript

Шестая версия ECMAScript была опубликована в июне 2015 года. Шестилетний перерыв после последнего ECMAScript и рост числа библиотек пользовательского интерфейса SPA привели к длительному пересмотру функциональности и стилей кодирования. Обновления коснулись объекта Number и глобальной области действия: isFinite и isNaN. Однако наиболее значительными изменениями являются новые ключевые слова для создания переменной, стрелочные функции, задания аргументов функций по умолчанию, деструктуризация, классы, модули и многие другие [12].

ES6 (ECMAScript 2015) являлся крупным и фундаментальным обновлением языка, который заложил основы современного JavaScript. После столь крупного обновления технический комитет Ecma решил иначе регулировать спецификацию языка, выпуская ежегодные менее масштабные версии ECMAScript. В результате чего стандарт ES7 (ECMAScript 2016) не привнес множество обновлений ограничиваясь двумя новыми функциями такими как новый метод поиска по массивам, поддерживающий значение массива NaN и новый оператор возведения в степень [13].

Следующее обновления стандарта ES8 (ECMAScript 2017) вышло в январе 2017 года. Ключевыми изменениями подверглись асинхронные операции и введена концепция «Shared memory and atomics». Введение конструкции

Async/Await для асинхронных функций позволило избежать неточностей вызова колбеков, а также улучшило внешний вид и читабельность кода. Ключевое слово «async» дает понять интерпретатору JavaScript что объявлена асинхронная функция. А ключевое слово «await» в свою очередь возвращает промис и ожидает его разрешение перед продолжением работы программы. Также это сделало возможным параллельный вызов функций с последующим применением конструкции Promise.all() [7]. Концепция «Shared memory and atomics» в спецификации описана следующим образом: «Общая память предоставляется в форме нового типа SharedArrayBuffer; новый глобальный объект Atomics обеспечивает атомарные операции над общими ячейками памяти, включая операции, которые можно использовать для создания блокирующих примитивов синхронизации». Это означает что за счет общей памяти можно разрешить нескольким потокам читать и записывать одни и те же данные с помощью нового конструктора SharedArrayBuffer. А объект Atomics гарантирует, что ничто из того, что пишется или читается, не будет прервано в середине процесса. Таким образом, операции завершаются до того, как начнется следующая. Данная концепция является заделом на бедующее внедрения возможности ручного управления памятью для создания высокопроизводительных параллельных программ [14].

Стандарт ES9 (ECMAScript 2018) вышел в июне 2018 года. Он привнес дополнение в раздел регулярных выражений в виде нового флага dotAll позволяющий установить соответствие одиночного символа для символов перевода строки, использование именованных групп внутри регулярных выражений, использование управляющих последовательностей Unicode и ретроспективная проверка существования некой строки перед другой строкой. Также были добавлены операторы Rest/Spread для работы со свойствами объектов, метод finally() для промисов и асинхронные итерации с новым оператором для циклов for-await-of [15].

Очередное обновления стандарта языка ES10 (ECMAScript 2019) произошло в июне 2019 года. Новшествами стали несколько методов массивов,

объектов и строк. Аргумент в блоке Catch при обработке ошибок стал опциональным [16].

Последней на сегодняшний день версией языка JavaScript является ES11(ECMAScript 2020) представленный в июне 2020 года. Обновление привнесло приватные переменные в классах, которые выдают ошибку на прямое обращение к ним извне класса. Введен новый тип данных BigInt расширяющий диапазон используемых численных значений. Добавлено новое свойство промисов, которое выполняет все промисы независимо от отклонения каких-либо промисов. С помощью Promise.allSettled можно создать новый промис, который возвращается только тогда, когда все переданные промисы выполнены. Следовательно, это дает доступ к матрице с данными каждого промиса. Добавлен динамический импорт модулей позволяющий асинхронно загружать модули приложения и возвращать промисы после загрузки [16].

1.1.2Основы работы браузера

Когда сервер отправляет данные в пакетах TCP, клиент пользователя подтверждает доставку, возвращая подтверждения. Соединение имеет ограниченную мощность в зависимости от условий аппаратного обеспечения и сети. Если сервер слишком быстро отправляет слишком много пакетов, они будут сброшены. Значение, не будет подтверждено. Сервер регистрирует это как отсутствующие подтверждения [4].

После того, как браузер получает первые данные, он может начать разбирать полученную информацию. Разбор полученных данных — это шаг, который браузер производит, чтобы преобразовать данные, которые он получает по сети, в DOM и CSSOM, которые используются, чтобы отобразить страницу на экран. DOM является внутренним представлением разметки для браузера. Им можно манипулировать через различные API в JavaScript [13].

Для оптимизации веб-ресурса важно учитывать правило первых 14 Кб, так как это тот размер первого пакета, определенный стандартом TCP. Следовательно, необходимо чтобы первый ответ сервера содержал все

необходимые данные для первоначального рендеринга. Но прежде, чем все отображается на экран, HTML, CSS и JavaScript должны быть проанализированы [14].

На первом этапе обрабатывается разметка HTML и строится дерево DOM. Разбор HTML включает в себя токенизацию и строительство дерева DOM. HTML-токены включают начальные и конечные теги, а также имена и значения атрибутов. Если документ хорошо сформирован, разборка документа происходит проще и быстрее [12]. Парсеры анализируют токенизированный ввод в документ, создавая дерево документа. Дерево DOM описывает содержание документа. Элемент <html> - первый тег и корневой узел дерева документа. Дерево DOM отражает отношения и иерархии между различными тегами. Теги, вложенные в другие теги, это дочерние узлы. Чем больше число узлов DOM, тем дольше требуется, чтобы построить дерево DOM [10].

Когда парсер находит неблокирующие ресурсы, такие как изображение, браузер запрашивает эти ресурсы и продолжают разбор документа. Разборка может продолжаться, до момента встречи тега <script> - особенно тех, которые используются без атрибута асинхронности, они останавливают разбор документа HTML. Хотя сканер Browser Preload ускоряет этот процесс, чрезмерные количество скриптов все еще могут грузится достаточно долго [11].

Когда браузер создает дерево DOM, этот процесс занимает основной поток обработки информации. В это время, сканер предварительного будет анализировать доступный контент и запрашивать приоритетные ресурсы, такие как CSS, JavaScript и веб-шрифты. Благодаря сканеру Preload не нужно ждать, пока парсер найдет ссылку на внешний ресурс, чтобы запросить его [14].

Оптимизации Preload Scanner обеспечивает снижение блокировки анализа документа. Чтобы скрипт не блокировал процесс, следует добавить атрибут async или атрибут отложенности. Ожидании скачивания CSS не блокирует разбор HTML или загрузку, но он блокирует выполнение JavaScript, поскольку JavaScript часто используется для влияния на CSS свойства элементов [16].

Второй шаг в критическом пути рендеринга – обработка CSS и построение дерева CSSOM. Объектная модель CSS похожа на DOM. DOM и CSSOM - оба деревья. Это независимые структуры данных. Браузер преобразует правила CSS в карту стилей, которые он может понять и обработать. Браузер проходит через каждое правило, установленное в CSS, создавая дерево узлов с родителями и дочерними отношениями на основе селекторов CSS. Как и в случае с HTML, браузер должен преобразовать полученные правила CSS в то, с чем он может работать. Следовательно, он повторяет схожий процесс, что и для HTML, но теперь для CSS. Дерево CSSOM включает в себя стили из таблицы стилей пользователя. Браузер начинается с наибольшего общего правила, применимого к узлу, и рекурсивно уточняет вычисленные стили, применяя более конкретные правила. Другими словами, он каскадит значения свойств. Создание CSSOM очень, очень быстро и не отображается уникальным цветом в текущих инструментах разработчиков [17].

Когда CSS проанализирован и создан CSSOM, другие данные, включая файлы JavaScript, загружаются (благодаря сканеру Preload). JavaScript интерпретируется, компилируется, анализируется и выполняется. Сценарии преобразовываются в абстрактные синтаксические деревья. Некоторые браузеры принимают абстрактное синтаксическое дерево и передают его в интерпретатор, которые выводят Bytecode, который выполняется в главном потоке [14].

Этапы рендеринга включают стили, макеты, отрисовку и, в некоторых случаях, композицию. CSSOM и DOM деревьев, созданные на ранних этапах, объединяют в общее дерево визуализации, которое затем используется для отображения каждого видимого элемента, который затем выводится на экран.

Третий шаг в критическом пути рендеринга сочетается в дереве DOM и CSSOM в дереве визуализации. Созданное дерево визуализации начинается с корня дерева DOM, проходящая каждый видимый узел. Каждый видимый узел имеет свои правила CSSOM, применяемые к нему. Дерево Render хранит все видимые узлы с контентом и определенными стилями - сопоставляя все соответствующие стили к каждому видимому узлу в дереве DOM [12].

Четвертым шагом в критическом пути является запуск дерева визуализации, чтобы вычислить геометрию каждого узла. Макетирование — это процесс, с помощью которого определяются ширина, высота и расположение всех узлов в дереве визуализации, а также определяется размеры и положения каждого объекта на странице. Как только дерево визуализации построено, начнется макетирование. Представление дерева, идентифицированное, какие узлы отображаются (даже если невидимы) вместе с их вычисленными стилями, но не размерами или местоположением каждого узла. Чтобы определить точный размер и местонахождение каждого объекта, браузер начинает визуализировать объекты начиная с корня дерева визуализации [13].

Принимая размер просмотра в качестве основы, макетирование обычно начинается с тега body, анализируя размеры всех потомков данного тега, причем свойства модели каждого элемента, затем предоставляется пространство для заполнения пространства элементами. Начальный размер и положение узлов, называется макетом. Последующие пересчеты размера узлов и местоположения называются отраслями [10].

На последнем шагу в критическом пути рендеринга обрисовываются отдельные узлы DOM на экране, первая отрисовка называется первой значимой отрисовкой. В этой фазе растеризации браузер преобразует все рассчитанное на этапе макетирования к фактическим пикселям на экране. Изображение включает в себя изображения каждой визуальной части элемента на экране, включая текст, цвета, границы, тени и элементы, такие как кнопки и изображения. Браузер должен сделать это очень быстро [11].

Когда разделы документа нарисованы на разных слоях, перекрывая друг друга, композиция необходима для того, чтобы они были отрисованы на экране в правильном порядке, и содержимое отображалось правильно. Когда изображение приходит с сервера, процесс рендеринга воспроизводится с шага макетирования [17].

После того, как главный поток отобразит страницу начинается загрузка и отработка отложенных сценариев поэтому в это время основной поток может

быть занят, и недоступен для прокрутки, касания и других взаимодействий. Время интерактивности (TTI) — это измерение того, сколько времени потребовалось от этого первого запроса до интерактивности страницы. Если основной поток занимается разбором, компиляцией и выполнением JavaScript, он не доступен и, следовательно, не сможет своевременно реагировать на взаимодействия пользователя в обычном порядке (менее 50 мс) [15].

1.2 Обзор актуальных фреймворков для разработки клиентской части веб приложений

В сфере программирования не существует четкого определения, каким критериям должно соответствовать программное обеспечение чтобы достоверно считаться фреймворком. Иногда фреймворки путают с библиотеками [14].

Библиотека обычно состоит из предварительно написанного кода, классов, процедур, скриптов, конфигураций и многого другого. В основном библиотеку можно с легкостью интегрировать в существующий проект для сокращения времени разработки. Это связано с тем, что многие вопросы, касающиеся основных алгоритмов и функций, уже были решены другими опытными программистами. Библиотека экономит время для разработчиков, позволяет сосредоточиться на проблемах, связанных с логикой. Но использование сторонней библиотеки также может представлять потенциальный риск для приложения [15].

По сравнению с библиотеками, фреймворки предлагают полный стек полезных функций и берут на себя ответственность за архитектурные решения в процессе разработки. Фреймворки могут включать в себя стратегии для маршрутизации URL-адресов в приложении, управления состоянием, объединения и других. Кроме того, фреймворки обеспечивают улучшения рабочего процесса, которые включают передовые практики для основных аспектов разработки, таких как общая структура приложения или создание шаблонов [14].

1.2.1Обзор фреймворка Angular

Angular — это полнофункциональный фреймворк охватывающий большую часть нужд разработчиков. С другой стороны, Angular может показаться сложным при изучении и использовании фреймворка, поскольку при этом требуется освоить большой набор взаимодействующих элементов (службы, внедрение зависимостей и т. Д.) [7].

Angular содержит все, что может понадобиться для разработки крупномасштабных интерфейсов. Для сравнения, React полагается на плагины, разработанные сообществом (например, для поддержки маршрутизаторов). В конечном итоге разработчик, находится внутри машины кодового мышления, которая хочет, чтобы вы соответствовали ее идеалам и соглашениям. Более того, крупномасштабные проекты с большим количеством членов команды могут выиграть от более жесткого и четко определенного архитектурного стиля, присутствующего в Angular [11].

Angular адоптировал классы ECMAScript. Эти классы используют встроенное состояние как состояние компонента. Они украшены аннотациями, определяющими их метаданные. Представления в Angular похожи на представления во Vue в том, что они представляют собой простые HTML-шаблоны с дополнительной привязкой данных и поддержкой логики с помощью встроенных директив. Компоненты можно рассматривать как небольшие части интерфейса, которые не зависят друг от друга. В то время как AngularJS был построен в основном на архитектуре Model-View-Controller (MVC), начиная с версии 2 Angular считается компонентным, что очень похоже на MVC, но обеспечивает более высокую возможность повторного использования компонентов в приложении. Это позволяет создавать пользовательские интерфейсы с множеством интерактивных частей и в то же время упрощает процесс разработки [14]. Основные преимущества данной архитектуры — это возможность повторного использования. Компоненты аналогичной природы хорошо инкапсулированы, другими словами, самодостаточны. Разработчики могут повторно использовать их в разных частях приложения. Это особенно

полезно в корпоративных приложениях, где разные системы сходятся воедино, но могут иметь много похожих элементов, таких как поля поиска, средства выбора даты, списки сортировки и т. Д. Инкапсуляция также гарантирует, что новые разработчики, которые недавно присоединились к проекту, могут лучше читать код и в конечном итоге быстрее выходить на плато производительности. В добавок независимый характер компонентов упрощает юнит-тесты, процедуры обеспечения качества, направленные на проверку производительности мельчайших частей приложения, юнитов. Компоненты, которые легко отделяются друг от друга, могут быть легко заменены лучшими реализациями самих себя [15].

Встроенный TypeScript: улучшает инструментарий, и делает код более чистым и более масштабируемым Angular написан с использованием языка TypeScript, который по сути является подмножеством JavaScript. Он полностью компилируется в JavaScript, но помогает выявлять и устранять типичные ошибки при кодировании. В то время как небольшие проекты JavaScript не требуют такого подхода, приложения корпоративного масштаба требуют от разработчиков делать свой код более чистым и чаще проверять его качество. Внесение политики Angular, основанной на TypeScript, в раздел преимуществ, является спорным моментом для многих инженеров [16]. Жалобы, связанные с TypeScript, то и дело появляются среди сообщества разработчиков. Разработчикам приходится учить еще один язык. Однако TypeScript существует не просто так, и остается возможность использовать JavaScript, если хотите. Виктор Савкин, бывший разработчик из команды Google Angular, объясняет, что переход от JavaScript к TypeScript оправдан инструментами для крупных проектов корпоративного масштаба. TypeScript имеет улучшенные службы навигации, автозаполнения и рефакторинга. В настоящее время TypeScript считается базовым языком для Angular, и для TypeScript также создается документация [17].

Как упомянуто ранее RxJS — это библиотека, обычно используемая с Angular для обработки асинхронных вызовов данных. Thinkster предлагает

рассматривать RxJS для кода JavaScript, как сборочную линию Генри Форда по производству автомобилей. Это позволяет обрабатывать события независимо параллельно и продолжать выполнение, не дожидаясь наступления какого-либо события и не оставляя веб-страницу без ответа. В принципе, это работает как сборочная линия, где исполнение разбито на отдельные и взаимозаменяемые части, а не привязано к одному человеку. Очевидно, что асинхронное программирование существовало и до RxJS, но эта библиотека упростила многие вещи. Хотя многие инженеры жалуются на кривую обучения RxJS. А также на сообщения об ошибках, которые слишком загадочны, чтобы их можно было понять без дополнительных исследований, сопровождаемых манипуляциями методом проб и ошибок. Библиотека работает с Observables, своего рода схемами, которые описывают, как объединяются потоки данных и как приложение реагирует на переменные в этих потоках [18].

Философия, не зависящая от платформы Angular была разработана с учетом подхода, ориентированного на мобильные устройства. Идея состоит в том, чтобы поделиться базой кода и, в конечном итоге, набором инженерных навыков в веб-приложениях с приложениями для iOS и Android. Чтобы реализовать это амбициозное позиционирование, в 2015 году разработчики Angular сотрудничали с командой, стоящей за фреймворком NativeScript (который ориентирован на создание близких к родным мобильным приложениям). Не только сам код, но и концепции Angular, такие как внедрение зависимостей, привязка данных, службы и маршрутизация, аналогичны как для NativeScript, так и для Angular. Однако этот агностицизм распространяется не на повторное использование кода, а на тот же набор инженерных навыков. Другими словами, разработчики должны использовать компоненты пользовательского интерфейса NativeScript для создания мобильных интерфейсов, но они будут работать в знакомых средах JavaScript и Angular, и кривая обучения работе с мобильными устройствами не будет такой крутой [7].

Основной фактором, который обеспечивает производительность приложений на Angular происходит за счет внедрения иерархических

зависимостей и поддержки Angular Universal. Внедрение иерархической зависимости. Angular использует улучшенную иерархическую инъекцию зависимостей по сравнению с AngularJS. Этот метод отделяет фактические компоненты от их зависимостей, выполняя их параллельно друг другу. Angular строит отдельное дерево инжекторов зависимостей, которое можно изменять без перенастройки компонентов [13]. Итак, классы не имеют зависимостей сами по себе, но потребляют их из внешнего источника. Каждому дереву компонентов назначено дерево инжекторов, которые содержат информацию о зависимостях. Такой подход обеспечивает высокую производительность приложений Angular. Как утверждает команда Angular, Angular 2 был в 5 раз быстрее, чем Angular 1.x. Angular Universal — это сервис, который позволяет отображать вид приложений на сервере, а не в клиентских браузерах. Google предоставляет набор инструментов для предварительной визуализации вашего приложения или его повторной визуализации для каждого запроса пользователя. В настоящее время набор инструментов адаптирован к серверным фреймворкам Node.JS и поддерживает ASP.NET Core. Google утверждает, что они собираются добавить поддержку PHP, Python и Java. Средство визуализации — это механизм, который переводит шаблоны и компоненты в JavaScript и HTML, которые браузеры могут понимать и отображать. Ivy — это третья итерация рендерера Angular после исходного компилятора и рендерера. Помимо других обновлений, Ivy применяет технику встряхивания дерева, что означает, что он удаляет неиспользуемые фрагменты кода, делая приложения меньше и позволяет быстрее его загружать. Он обратно совместим: после обновления Angular существующие приложения будут отображаться с помощью Ivy без дополнительных хлопот [17].

Некоторые инженеры-программисты считают сам факт поддержки Angular со стороны Google главным преимуществом этой технологии. Хотя это может показаться оправданным, самого Google недостаточно. Хорошим признаком является то, что Google объявил о долгосрочной поддержке (LTS) для этой технологии. Игорь Минар и Стивен Фуин, инженеры Angular, подтвердили эту приверженность в Keynote ng-conf 2017. По сути, это означает, что Google

планирует придерживаться экосистемы Angular и развивать ее, пытаясь удерживать лидирующие позиции среди инструментов фронтенд-инжиниринга. Поскольку Angular существует уже довольно давно, его завалили пакетами, плагинами, надстройками и инструментами разработки. Вы можете изучить часть работы сообщества, просмотрев список ресурсов Angular. К ним относятся IDE, инструменты, среды пользовательского интерфейса, Angular Universal для рендеринга на стороне сервера, о котором мы упоминали выше, инструменты аналитики, средства для ASP.NET, библиотеки данных и т. Д. Если средний инженер заблудился, всегда найдется инструмент, который поможет решить возникающую проблему [14].

Независимо от заявлений LTS, именно сообщество вокруг любой технологии делает ее сильной на рынке. И история сообщества Angular довольно противоречива. Согласно опросу разработчиков StackOverflow 2018 года, Angular (как Angular 1.x, так и Angular) является второй наиболее часто используемой технологией в категории Frameworks, Libraries и Other Technologies, что неплохо. Но реальность такова, что разработчики, которые обычно используют Angular, скорее всего, откажутся от работы согласно тому же опросу в категории «Самые любимые, опасные и разыскиваемые фреймворки, библиотеки и инструменты». В настоящее время TensorFlow является самой популярной технологией, набравшей 73,5% голосов, за ней следует React с 69,7%. А 45,4 процента респондентов считают Angular самой опасной технологией. Он ниже, чем у Hadoop (46,1), Xamarin (51) и Cordova (59,6), но все же достаточно высок. С другой стороны, результат ниже, чем в предыдущем году (48,3), и, вероятно, люди, которые придерживаются AngularJS в устаревших продуктах, также проголосовали. Итак, некоторая часть этой негативной реакции может быть адресована AngularJS. В основном это связано с тем, что пользователи AngularJS 1.x, скорее всего, откажутся от работы и не подумают о переходе на современный Angular. Так как невозможно напрямую обновиться с AngularJS 1.x [15].

Компании, использующие Angular: Microsoft, Autodesk, MacDonald's, UPS, Cisco Solution Partner Program, AT&T, Apple, Adobe, , Clarity Design System, Upwork, Udemy, YouTube, Paypal, Nike, Google, Telegram, AWS.

1.2.2Обзор фреймворка React

Сильная сторона React состоит в том, что он органически вырос в результате использования в мире и продолжает развиваться в ответ на интенсивное использование. Он претерпел значительный рост, но его корни и постоянные преимущества заключаются в том, что он является фреймворком, используемым Facebook для собственных приложений [12].

Можно увидеть приверженность Facebook к продвижению инноваций в этой структуре с помощью перспективных функций, таких как Concurrent Model. React также активно разработал так называемые «чистые» или функциональные компоненты и хуки, чтобы расширить их возможности. Эти компоненты позволяют избежать некоторых накладных расходов, связанных с компонентами на основе классов. Vue имеет некоторую поддержку функциональных компонентов, и их можно создавать в Angular, но React - явный лидер в этой области [13].

Angular согласованно обрабатывает большие базы кода и может предложить преимущества по сравнению с React в области крупномасштабных проектов. В React можно определять элегантные крупномасштабные приложения, но сам фреймворк не будет делать столько, сколько Angular, чтобы обеспечить соблюдение этого определения [17].

React использует неизменяемые состояния объекта, доступные только через setState (), для представления состояния компонента. Это отличается от Vue и Angular, которые используют более встроенный подход JavaScript к состоянию данных [20].

React использует JSX для своих шаблонов представлений. JSX - интересный подход, поскольку он похож на HTML со сверхспособностями JavaScript (или JavaScript со сверхспособностями HTML). JSX может немного

оттолкнуть при первом изучении фреймворка. В долгосрочной перспективе он работает довольно хорошо, и его несложно изучить. Централизованное управление данными React по умолчанию осуществляется через Redux [21].

Преимущества React [13]:

- Экономическая эффективность - поскольку нет необходимости начинать тратить большие деньги с самого начала и благодаря сокращенному времени вывода на рынок.
- Отличный пользовательский опыт - никто не любит стоять в очереди и ждать, пока веб-сайт полностью загрузится. У React есть функции, которые делают приложения React невероятно быстрыми: Concurrent Mode, React Fiber, Suspense, virtual DOM и многие другие.
- Популярность React не является преходящим сезонным трендом. Это стабильная и зрелая технология.
- Повышение производительности и скорости - когда дело доходит до таких технологий, как библиотеки или фреймворки, их производительность и скорость являются ключом к успеху. Так как если React может справиться с Facebook, наиболее часто используемым приложением, он может справиться с чем угодно. И причина того, что React такой быстрый, - это виртуальный DOM. Вместо того, чтобы каждый раз генерировать все файлы HTML, он ищет различия между старым DOM и текущим файлом и обновляет его соответствующим образом [14].
- Оптимизация для SEO. В то время как поисковым системам трудно работать с приложениями с тяжелым JavaScript, для приложений React все по-другому. Это потому, что они могут работать на сервере, в то время как виртуальная модель DOM заботится о рендеринге и отрисовке в браузер.
- Сокращенное время выхода на рынок (TTM) - время - деньги, особенно для стартапов и других компаний, стремящихся создавать цифровые продукты. Поскольку React ускоряет весь процесс разработки, можно намного быстрее протестировать MVP на рынке и внести соответствующие изменения, не тратя на это больших денег [15].

- Обратная совместимость - дело в том, что некоторые фреймворки требуют переписывания кода после обновления фреймворка до последней версии. Но, не в случае с React.js. Общедоступный API всегда остается практически неизменным, поэтому Facebook и другим компаниям проще обновлять код, используя его старые части.

- Время разработки - React — это скорость, и это означает не только скорость загрузки страницы, но и скорость разработки. Многие вещи ускоряют процесс разработки: 1. готовые решения 2. созданные компоненты можно использовать повторно 3. возможность настроить веб-приложение с помощью «Create React App», который является стартовым комплектом, предоставляемым Facebook [16].

- Легко масштабируется - благодаря модульности приложения React легко масштабируются. Можно начать с малого и оттуда развиваться. Например, вы можете начать с создания простого MVP, а затем сделать его одностраничным приложением таким же всеобъемлющим, как Facebook.

- Полезные инструменты - разработчики React имеют в своем распоряжении набор удобных инструментов под названием React Developer Tools. Это расширение для Google Chrome и Mozilla Firefox, которое позволяет проверять иерархию компонентов в виртуальной DOM. Это также позволяет редактировать отдельные компоненты.

- Огромное сообщество - Сообщество React с момента своего основания в 2013 году быстро росло и, продолжает расти. Велика вероятность, что проблема, которую пытается решить разработчик, уже решена или он легко найдет того, кто ему поможет. Это потому, что есть много разработчиков, которые делают React лучше и лучше [12].

- Многоразовые компоненты, также известные как модульность - здесь очень помогает компонентный подход, потому что приложения React состоят из отдельных компонентов, которые можно разделить на подкомпоненты. Каждый компонент и подкомпоненты отвечают за одну небольшую часть всего приложения и могут быть повторно использованы в

любое время и в любом месте. Благодаря этому можно создать сложное приложение из простых блоков.

- Удобство тестирования и отладки - ReactJS использует собственные инструменты для тестирования и отладки компонентов перед их фактическим использованием [13].

Как и у любой другой технологии, у React есть свои недостатки. Так React предоставляет только часть представления модели MVC. Из-за этого разработчикам придется полагаться и на другие технологии. Однако некоторые разработчики воспринимают это как преимущество, поскольку позволяет обеспечить полную независимость. Из-за этого каждый проект может выглядеть по-разному. Еще React - все еще довольно новая технология, и она стремительно развивается. Поэтому некоторым может быть трудно поддерживать этот темп из-за появления новых функций и удаления старых. А некоторым просто не нравятся постоянные изменения [15].

В умелых технических руках React - мощный инструмент для создания надежных, производительных и масштабируемых цифровых продуктов. Владельцы бизнеса особенно оценят его рентабельность, сокращение времени вывода на рынок и оптимизацию SEO. С другой стороны, разработчикам понравится компонентный подход и сокращенное время разработки [17].

Компании, использующие React: Facebook, Instagram, Netflix, Yahoo, Whatsapp, Dropbox, Airbnb, Atlassian, Microsoft, и многие другие.

1.2.3Обзор фреймворка Vue

В некотором смысле Vue находится где-то между Angular и React, компромиссом между нисходящим дизайном Angular и органическим ростом React. Несмотря на то, что Vue является новым фреймворком и не имеет поддержки со стороны крупной корпорации, он идет в ногу с разработками и предоставляет полностью жизнеспособную структуру. Также существует ряд качественных плагинов и комплектов для Vue (например, Quasar и Vuetify)[22].

Vue имеет репутацию самого простого в освоении. Вероятно, это происходит из его модели данных JSON и определений представления (по сравнению с JSX React). Шаблоны Vue также могут включать встроенные функции JavaScript, в отличие от JSX. Vue предлагает Vuex в качестве встроенного централизованного решения для управления состоянием [23].

Преимущества Vue [23]:

- Данный фреймворк обладает самой плавной кривой обучения. Также этому способствует обширная и качественная документация.
- Универсальность позволяет Vue масштабироваться между библиотекой и полнофункциональным фреймворком, что упрощает разработчикам создание фантастических приложений.
- Vue упрощает разработку, так как готовый к развертыванию проект весит 20 КБ после min + gzip. Это приводит к ускорению выполнения, а также стимулирует разработку и позволяет разработчикам отделять виртуальную модель DOM от шаблона и от компилятора. Более того, когда минимальный размер проекта, не нужно прилагать дополнительные усилия для чрезмерной оптимизации [15].
- Будучи доступным, универсальным и производительным, Vue помогает разработчикам создавать приложения, которые можно сопровождать и тестировать. Его можно использовать для создания как SPA (одностраничных приложений), так и очень сложных веб-приложений, поскольку разработчики могут свободно интегрировать более мелкие части в существующую инфраструктуру, не затрагивая всю систему [16].
- В отличие от других фреймворков, Vue.js можно использовать как библиотеку и как полноценный фреймворк. Его можно модифицировать, изменять и масштабировать в соответствии с требованиями. В то же время, если необходимо выполнить какую-либо интеграцию или разработать веб-приложение, оно может использовать REST API, будь то разработка WordPress или WooCommerce [17].

К негативным сторонам Vue относится то, что Vue быстро развивается. То, что работает сегодня во Vue, завтра может устареть. Это наиболее значительный недостаток Vue, поскольку разработчики находят его довольно сложным, и им приходится довольно часто изучать фреймворк. Также следует отметить небольшое сообщество. В результате он не так популярен по сравнению с другими фреймворками, такими как React и Angular. Кроме того, поскольку фреймворк был создан китайской компанией, большая часть кода была написана на китайском языке, что создает некоторые проблемы для англоговорящих пользователей. Большинство членов сообщества не говорят по-английски, что тоже в небольшом количестве, что может не поддержать [22].

Компании, использующие Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab и Laracasts, Adobe, Behance, Codeship, Reuters.

1.3 Современные подходы фронтенд разработки

Большинство предприятий приняли Интернет в качестве своей бизнес-платформы, и эта тенденция усиливается. Таким образом, существует потребность в разработке быстрых, надежных и устойчивых ресурсов, реализующих актуальные и перспективные технологии [6].

1.3.1 Одностраничные приложения

Разработчики гнались за мечтой о предоставлении веб-приложений с внешним видом и ощущениями, присущими нативным настольным приложениям, примерно с тех пор, как они их писали. Различные решения для более естественного взаимодействия, такие как IFrames, Javaapplets, Adobe Flash и Microsoft Silverlight, были опробованы с разной степенью успеха. Несмотря на то, что технологии различаются, все они имеют по крайней мере одну общую цель: привнести мощь настольного приложения в тонкую кроссплатформенную среду веб-браузера. Одностраничное (веб-приложение) или SPA разделяет эту цель, но без подключаемого модуля браузера или нового языка для изучения [17].

Действие происходило в начале 2000-х. Совершенно новый взгляд на дизайн веб-страниц возник, когда движение AJAX начало набирать обороты. Все началось с интересного, но непонятного элемента управления ActiveX в браузере Microsoft Internet Explorer, который использовался для асинхронной отправки и получения данных. Эти скромные начинания в конечном итоге привели к революции, когда функциональность элемента управления была официально принята основными поставщиками браузеров как API XMLHttpRequest (XHR). Разработчики, которые начали объединять этот API с JavaScript, HTML и CSS, получили замечательные результаты. Сочетание этих методов стало известно как AJAX, или асинхронный JavaScript и XML. Ненавязчивые запросы данных AJAX в сочетании с возможностями JavaScript для динамического обновления объектной модели документа (DOM) и использованием CSS для изменения стиля страницы на лету вывели AJAX на передний план современной веб-разработки. Продолжая это успешное движение, концепция SPA выводит веб-разработку на совершенно новый уровень, распространив технику манипуляции на уровне страниц в AJAX на все приложение. Кроме того, шаблоны и методы, обычно используемые при создании SPA, могут привести к общей эффективности в дизайне приложений, сопровождении кода и времени разработки [18].

Как и в случае с большинством новых решений, дизайн одностраничных приложений включает в себя множество подходов. Различные мнения современных экспертов, а также множество конкурирующих библиотек и фреймворков могут сделать поиск правильного решения для вашего проекта SPA сложной задачей [19].

Одностраничное приложение (SPA) — это веб-приложение, в котором маршрутизацию обрабатывает не серверная часть, а клиентский JavaScript. SPA быстро и легко использовать, потому что пользователю не нужно ждать, перезагрузки страниц при посещении разных страниц одного ресурса и не требуется загружать разные повторяющиеся элементы страницы такие как шапка, сайдбар или подвал сайта. Приложение загружает состояние на лету. Это приводит к улучшению взаимодействия с пользователем, поскольку приложение

обладает высокой производительностью [18]. После начальной загрузки страницы все инструменты, необходимые для создания и отображения представлений, загружаются и готовы к использованию. Если необходимо новое представление, оно создается локально в браузере и динамически прикрепляется к DOM через JavaScript. Никаких обновлений браузера не требуется. Поскольку наша логика представления в SPA в основном является клиентской, задача объединения HTML и данных переносится с сервера в браузер. Как и на стороне сервера, исходный HTML-код содержит заполнители, в которые должны быть вставлены данные (и, возможно, другие инструкции по рендерингу) [19].

Веб-браузер по-прежнему остается отличным способом распространения программного обеспечения из-за его повсеместного распространения и стандартизированной среды. У конечных пользователей уже есть веб-браузер. Он также отлично подходит для обновлений программного обеспечения, поскольку обновления происходят на сервере, а пользователям не нужно беспокоиться о процессе установки. К сожалению, перезагрузки страниц, дублирование контента при каждом запросе и большие объемы транзакций уменьшают преимущества контента, доставляемого браузером. Однако взаимодействие с клиентами через SPA позволяет предоставить конечным пользователям наилучшие возможности [20].

1.3.2 Прогрессивные веб приложения

Прогрессивное веб-приложение (PWA) — это новое поколение веб-приложений. Оно предлагает похожий пользовательский опыт, как и при использовании нативных приложений при работе с веб-сайтом, который использует функции PWA. В частности, PWA обеспечивает возможность просмотра веб-страниц без интернета после предварительной разовой загрузки, а также интерактивные пользовательские сервисы, полностью используя кэш, push-уведомления и сервисную рабочую силу. Совокупность этих новых функций HTML5 стирает границу между нативными и веб-приложениями, особенно на мобильных устройствах, способствуя использованию веб-

приложений с короткими задержками поставки данных и более короткому пути публикации и доступа пользователя [21].

Однако это не новая технология или фреймворк, а лучшие практики, которые были приняты в Интернете, чтобы дать пользователю ощущение нативного приложения. Такие компоненты, как Service worker, AppShell, манифест веб-приложения и push-уведомления, работая вместе придают PWA ощущение оригинальности. Его можно установить на главный экран пользователя с ресурса веб-сайта одним касанием. Пользователь может наслаждаться полноэкранным режимом, не испытывая хлопот, связанных с процессом загрузки. Оно загружается быстрее даже в нестабильной сети и работает в автономном режиме [22]. PWA имеет возможность отправлять соответствующие push-уведомления, тем самым увеличивая вовлеченность пользователей. Популярность PWA растет очень быстрыми темпами в сфере электронной коммерции, бизнеса, новостных онлайн-порталов и других областях. Ключевым преимуществом данной технологии является то, PWA работает для всех пользователей во всех браузерах и имеет возможность своевременного обновления, предоставляя пользователям всегда актуальный функционал и релевантную информацию. Новый опубликованный контент приходит пользователю в виде обновление, как только пользователь подключается к Интернету. Оболочка и содержимое приложения после кэширования всегда загружаются из локального хранилища [23]. Также помимо системы обновлений имеется Push-уведомления, которые работают за счетинтерфейсов Notifications Push API и Notifications API. Данные интерфейсы повышают вероятность повторного посещения PWA пользователем. Пользователь может получать сообщения, отправленные с сервера, которые могут отображаться как уведомления. Уведомления прогрессивных веб-приложений полностью естественны и похожи на уведомления собственных приложений [22].

Помимо этого, в плане доступности PWA стоит отметить сохранившуюся структуру веб-сайтов, которая подходит для всех форм,

факторов и размеров устройств: мобильных, настольных и планшетных. Адаптивная функция достигается за счет дизайна, концепции гибкой сетки и медиа-запросов CSS 3. Доступность данных приложений также обуславливается манифестом веб-приложения W3C за счет чего поисковые системы идентифицируют PWA как приложение. Это увеличивает вероятность отображения в поисковых системах по сравнению с нативным приложениями, доступными в соответствующих магазинах. Внедрение сервис-воркерев позволяет PWA работать в автономном режиме и обеспечивать хорошую производительность даже в локальной сети. Приложение рассматривает потерю подключения не как ошибку, а как случай, который можно запланировать и обработать без промедления [23].

Важным моментом является обязательное применение HTTPS-соединения и SSL-сертификата для обслуживания страницы, что позволяет предотвратить атаки типа «злоумышленник в середине», взлома пароля и манипуляций с контентом [22].

PWA являются продуктом Интернета, то есть быстрее и дешевле для разработки, снижая затраты на привлечение пользователей и всегда имеют актуальную версию функционала и контента. Тем не менее, PWA сталкивается с некоторыми проблемами [21].

Из проблем технологии стоит отметить проблемы с кроссбраузерностью, вследствие чего некоторые браузеры имеют ограниченную функциональность, поскольку они не могут получить доступ ко всему аппаратному обеспечению для конкретного устройства. Также отсутствует поддержка входа в систему между приложениями [24].

1.3.3 Теневая и виртуальная объектная модель документа

DOM (объектная модель документа) – это фундаментальная концепция в клиентской разработке, и наверняка каждый, кто пытался изучить веб программирование, слышал о ней не раз. Манипуляции с DOM не так просты и удобны, и, что самое важное, они создают множество проблем с

производительностью. В настоящее время существуют две основные концепции DOM, Shadow DOM и Virtual DOM которые появились с прогрессивными веб-фреймворками, такими как Angular, React.js и Vue.js [28].

DOM – это API для документов HTML или XML, и он создает логическую структуру, к которой можно получить доступ и которой можно управлять. Другими словами, Javascript может получать доступ и вносить изменения в объектную модель документа. Причина реализации объектной модели документа заключалась в том, чтобы предоставить стандартный интерфейс программирования, который можно было бы использовать с любым языком программирования в различных средах. Под модификацией DOM мы можем понимать добавление, удаление или изменение элементов веб-сайта, назначение им различного поведения и т. Д. Каждый браузер имеет свой глобальный объект, называемый window. Внутри window есть разные свойства и методы. Одно из свойств объекта window — это документ, в котором мы можем найти множество свойств и методов, которые можно использовать для доступа к элементам DOM для взаимодействия с ними [29].

Shadow DOM– это инструмент, используемый для создания приложений и веб-сайтов на основе компонентов. Shadow DOM состоит из небольших частей и не представляет всю объектную модель документа. Мы можем рассматривать его как поддерево или как отдельную DOM для элемента. Теневой DOM можно представить как кирпичи, из которых создается DOM. Основное различие между DOM и Shadow DOM заключается в том, как он создается и как ведет себя. Обычно узлы DOM, которые мы создаем, размещаются внутри других элементов, как в дереве, которое мы видели ранее. В случае Shadow DOM мы создаем дерево с областью видимости, которое связано с элементом, но отделено от дочерних элементов. Оно называется теневым деревом, а элемент, к которому оно прикреплено, называется теневым хостом. И здесь мы подходим к большому преимуществу Shadow DOM: все, что мы добавим в Shadow DOM, является локальным, даже стили [29].

Прежде всего, он изолирует DOM, поэтому DOM компонента является отдельным элементом, который не отображается в глобальной DOM. Еще одна проблема, с которой он помогает, определение объема CSS, что означает, что стили, созданные внутри одного элемента Shadow DOM, изолированы и остаются в рамках этого Shadow DOM. Это значительно упрощает стилизацию, поскольку нам не нужно сильно беспокоиться об именовании пространства, и мы можем использовать простые селекторы и имена классов. Кроме того, мы можем думать о приложении, как о том, что оно построено из блоков (на самом деле оно основано на компонентах), а не как об одном массивном глобальном объекте. Shadow DOM может повлиять на производительность приложения. В случае с Shadow DOM браузер знает, какую часть нужно обновить [30].

Виртуальный DOM — это концепция DOM, используемая React.js и Vue.js. В концепции виртуального DOM копия DOM сохраняется в памяти, и, хотя любые изменения выполняются в DOM, она сравнивается, чтобы найти различия. Затем браузер знает, какие элементы были изменены, и может обновлять только эту часть приложения, чтобы избежать повторного рендеринга всей DOM. Это сделано для повышения производительности библиотек пользовательского интерфейса. Как мы знаем, из предыдущего абзаца в DOM каждый элемент перерисовывается, независимо от того, был он изменен или нет. В концепции Virtual DOM можно применить более одного изменения одновременно, чтобы избежать повторного рендеринга для каждого отдельного изменения элемента. Самая большая проблема, которую решает Virtual DOM, — это повышение производительности при манипулировании DOM [30].

Единственное, что их объединяет, это то, что они помогают с проблемами производительности. Обе создают отдельный экземпляр объектной модели документа, помимо этого, обе концепции различны. Virtual DOM создает копию всего объекта DOM, а Shadow DOM создает небольшие части объекта DOM, которые имеют свою собственную изолированную область действия для элемента, который они представляют [28].

1.4 Формирование требований к разрабатываемому фреймворку по результатам аналитического обзора

В результате аналитического обзора предметной области данной работы сформированы несколько групп требований к разрабатываемому фреймворку в таблице 1.

Таблица 1 – Группы требований

Код	**Группа требований**
B	Бизнес-требования
F	Общие функциональные требования

В таблицах 2-3 представлена спецификация вышеописанных групп требований.

Таблица 2 – Бизнес-требования

Код	**Требование**
B.1	Требования к безопасности
B.1.1	Количество строк кода фреймворка должно быть не большим для его проверки на уязвимости
B.2	Требования к надежности
B.2.1	Возможность самостоятельной поддержки кода фреймворка
B.3	Возможность работы с фреймворком специалистов любого ООП языка
B.4	Возможность тонкой настройки фреймворка под нужды компании
B.5	Возможность совместимости фреймворка со сторонними библиотеками
B.6	Возможность подконтрольного обновления кодовой базы фреймворка
B.7	Возможность оперативно реагировать на развитие технологий и стандартов
B.8	Отечественная разработка

Таблица 3 – Общие функциональные требования

Код	**Требование**
F.1	В фреймворке должно быть оптимизированно взаимодействие с DOM API
F.2	В фреймворке должна присутствовать организация работы с пользовательским интерфейсом в виде изменения JavaScript-объекта
F.3	В фреймворке должно быть оптимизированно использование обработчика событий
F.4	В фреймворке должна присутствовать организация подконтрольного обновления пользовательского интерфейса
F.5	В фреймворке должно быть оптимизированно разделение логики приложения и ее сгруппированное размещение
F.6	В фреймворке должно быть реализовано создание одностраничного роутинга
F.7	В фреймворке должно быть реализовано разделение приложения на компоненты
F.8	Возможность организации возможности взаимодействия компонентов
F.9	В фреймворке должно быть реализовано создание класса страницы для создания и группировки компонентов, а также управления событиями жизненного цикла
F.10	В фреймворке должно быть разработаны стратегии разбиения проекта на бандлы, и точки входа
F.11	В фреймворке должна быть разработана стратегии кеширования
F.12	В фреймворке должны быть разработаны вспомогательные утилиты для преобразования верстки в одностраничное приложение
F.13	В фреймворке должен быть разработан интерфейс настроек и вспомогательной нотации верстки для утилит
F.14	В фреймворке должны быть разработаны утилиты для создания отельной утилиты и скриптов

Опираясь на сформированные бизнес-требования и функциональные требования, необходимо спроектировать и разработать фреймворк, удовлетворяющий этим требованиям.

2. Проектирование и разработка фреймворка

Разрабатываемый фреймворк представляет собой совокупность классов, дающие возможность создавать одностраничные прогрессивные веб приложения. Главными элементами являются классы: компонент, страница, роутер, управление событиями, а также вспомогательные утилиты на платформе NodeJS.

Основными особенностями фреймворка является использование представления пользовательского интерфейса в виде JavaScript объекта в совокупности с виртуальным DOM деревом, которые обеспечивают основательный и гибкий подход к созданию приложений. Для обеспечения надежности и повышения качества кода по умолчанию используется TypeScript, а также отсутствуют сторонние зависимости в работе написанных на данном фреймворке приложений. Однако существуют зависимости в момент разработки такие как: WebPack, TypeScript, Babel, NodeJS. Они позволяют сделать разработку удобнее и быстрее и не оказывают влияние на конечный продукт.

2.1 Проектирование и разработка класса компонента.

Компоненты являются строительными блоками клиентских веб приложений, позволяя разделить приложение на логические составляющие. Обычно они состоят из части, отвечающей за пользовательский интерфейс и соответствующей ему программной логики. Данный подход позволяет с легкостью переиспользовать уже созданные компоненты, а также облегчает поддержку существующей кодовой базы. Angular, React и Vue также реализуют компонентный подход в создании приложений.

Подход к созданию и использованию компонентах во всех трех фреймворках схожий. Компоненту назначается HTML тег с помощью, которого его можно будет вызвать в требуемом коде. Инициализация и вызов компонента происходит за счет размещение HTML тега с соответствующим названием в требуемом месте. Пример инициализации аналогичных компонентов в проекте,

реализованном с применением различных фреймворков представлен на рисунке 1.

```
<div>
    <app-settings-modal></app-settings-modal>
    <h4>
        <span>Your Resources</span>
        <span>{{ resourcesCount }}</span>
    </h4>
    <app-resource-search (onSearch)="handleSearch($event)">
    </app-resource-search>
    <app-resource-list
        (onResourceClick)="handleResourceSelect($event)"
        [activeId]="activeResource?._id"
        [resources]="resources"
    ></app-resource-list>
</div>
```

а

```
<div>
    <SettingsModal />
    <h4>
        <span>Your Resources</span>
        <span>{resources.length}</span>
    </h4>
    <ResourceSearch onSearch={searchResources} />
    {loading ? ('Loading Resources') : (
        <ResourceList
            activeId={activeResource?._id}
            onItemClick={setSetlectedResource}
            resources={resources}
        />
    )}
</div>
```

б

Рисунок 1 – Инициализация аналогичных компонентов в проекте, реализованном с применением различных фреймворков: а. Angular; б. React

Передача необходимых параметров происходит в атрибутах тега при вызове экземпляра компонента. Однако данный подход не разделяет логически понятие страницы и элемента страницы, а также в случае Angular не позволяет полноценно работать с классами [13].

В разрабатываемом фреймворке компонент состоит из экземпляра виртуального представления DOM дерева, который реализует интерфейс VNode, представленный на рисунке 2, а также класса, который отвечает за поведение данного компонента.

interface VNode
tag:string attributes:{[key:string]:string} $element:HTMLElement Children:VNode[] | string

Рисунок 2 – Интерфейс экземпляра элемента виртуального DOM дерева

Структура класса Component представлена на рисунке 3. Пользовательские компоненты наследуются от данного класса.

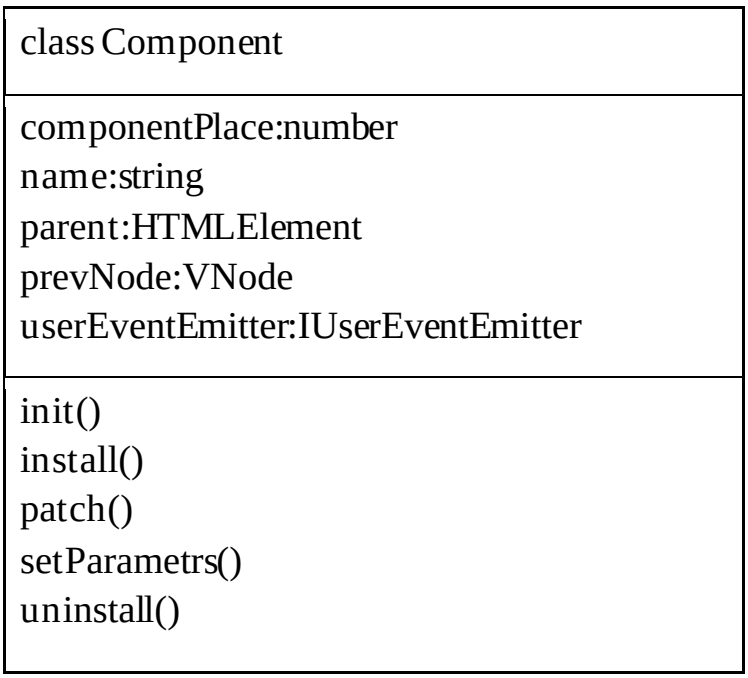


Рисунок 3 – Структура класса Component

Обязательными методами для реализации в пользовательских компонентах является метод install который отвечает за первичную отрисовку компонента. Алгоритм работы данного метода представлен на рисунке 4. Так сначала создается HTML элемент, соответствующий тегу, и устанавливаются его атрибуты. Затем генерируется уникальный id для элемента и устанавливается атрибутом и значением соответствующему полю виртуального представления. Затем проверяется тип дочернего элемента, если это текст, то текст добавляется к элементу, иначе это массив дочерних элементов, которые рекурсивно устанавливаются в родительский элемент. Затем происходит установка всего созданного из виртуального представления HTML элемента в родительский для компонента элемент и состояние клонируется в поле для исходного виртуального представления страницы.

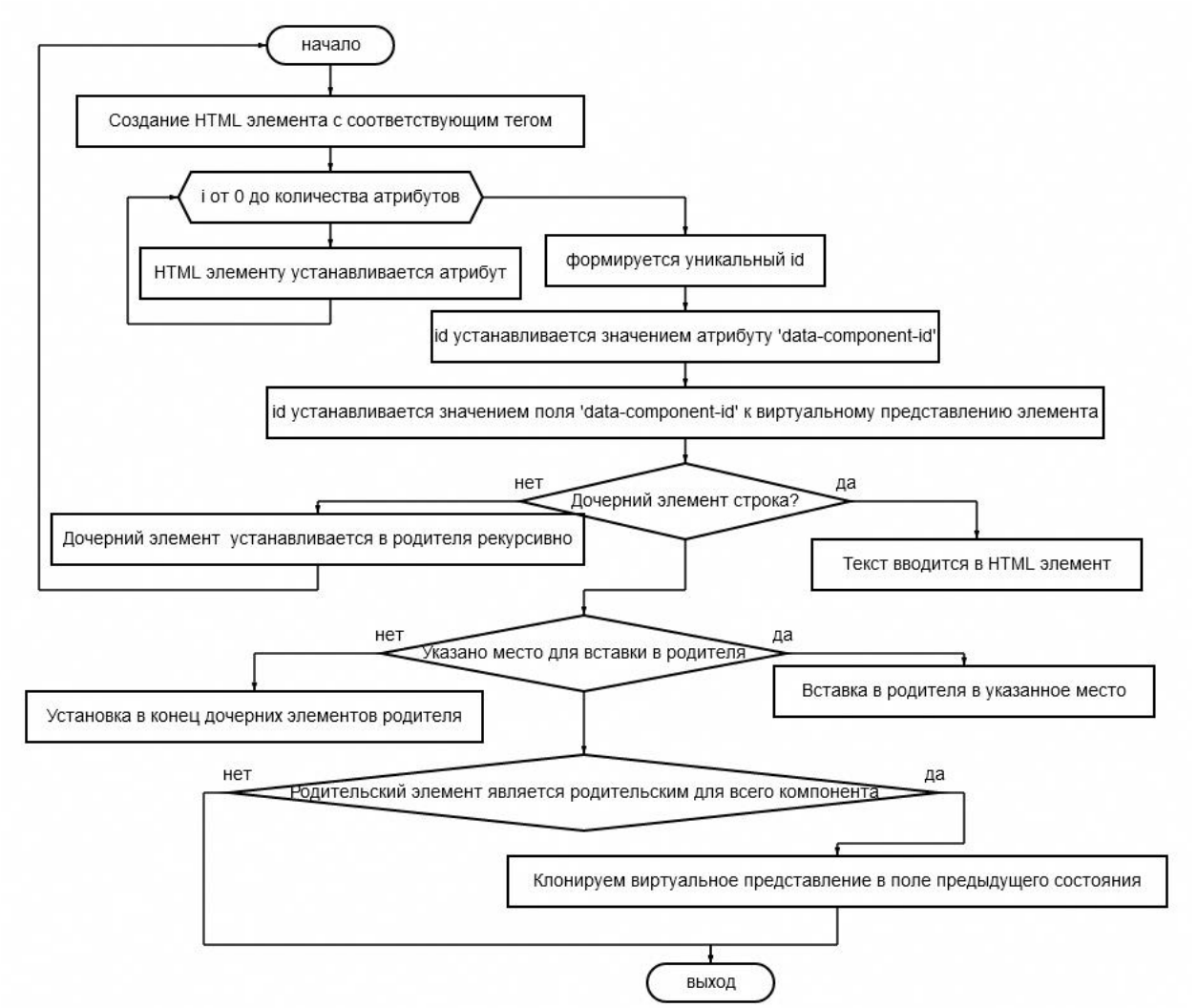


Рисунок 4 – Алгоритм установки виртуального представления на страницу

В классе Component имеется поле хранящее предыдущее виртуальное состояние компонента, оно создается после его установки и перезаписывается после каждого обновления. Пользовательский компонент может взаимодействовать с полями parent для добавления обработчиков событий к компоненту и с userEventEmitter для того, чтобы создавать события. В конструкторе класса пользовательского компонента следует клонировать исходное состояние пользовательского интерфейса для того, чтобы избежать мутаций исходного состояния при его переиспользовании. Для того чтобы синхронизировать пользовательский интерфейс с состоянием приложение следует изменить соответствующий элемент виртуального представление и передать его в вызов функции patch. Алгоритм работы функции представлен в Приложении Б. Аргументами данной функции являются текущее виртуальное

представление, соответствующий HTML элемент, сформированный при установке и новое виртуальное представление. По умолчанию в функцию необходимо передавать только новое виртуальное представление, остальные параметры предопределены. В начале выполнения функции проверяется наличие id у элемента, если его нет, то генерируется новый id. Если id у исходного состояния и нового отличаются, то удаляется старое состояние методом uninstall и устанавливается новое. Если id равны, то происходит проверка и обновление атрибутов если это необходимо. Следом проверяется являются дочерние элементы у нового состояния строками. Если дочерние элементы строчные, то устанавливается новое значение если необходимо. Если в исходном состоянии тип дочерних элементов был строкой, а в новом состоянии не строка то устанавливаются новые дочерние элементы. И в последнем случае если в исходном и новом состоянии были массивы дочерних элементов, то происходит их сравнение, перестановка, удаление и изменение если требуется. После всех операций в исходное состояние клонируется новое установленное состояние. Таким образом при изменении виртуального представления на странице происходят детальные изменения без полной перерисовки страницы.

Использование JavaScript объекта позволяет сосредоточить всю логику внутри класса делая код и поведение страницы понятнее, чем код, в котором логика распределена по двум различным файлам. В итоге пользователю для создания собственного компонента необходимо:

1. Наследоваться от класса Component
2. Клонировать исходное состояние в конструкторе
3. При инициализации компонента после его установки добавить необходимые обработчики событий на весь компонент и определять необходимый целевой элемент за счет data атрибутов HTML элементов.
4. При необходимости обновить пользовательский интерфейс вызвать функцию patch передов в нее аргументов измененное виртуальное состояние компонента.

2.2 Проектирование класса UserEventEmitter

Класс UserEventEmitter позволяет компонентам создавать события и подписываться на созданные другими компонентами события. Данный класс представляет собой реализацию паттерна наблюдатель. Наблюдатель— это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах [31].

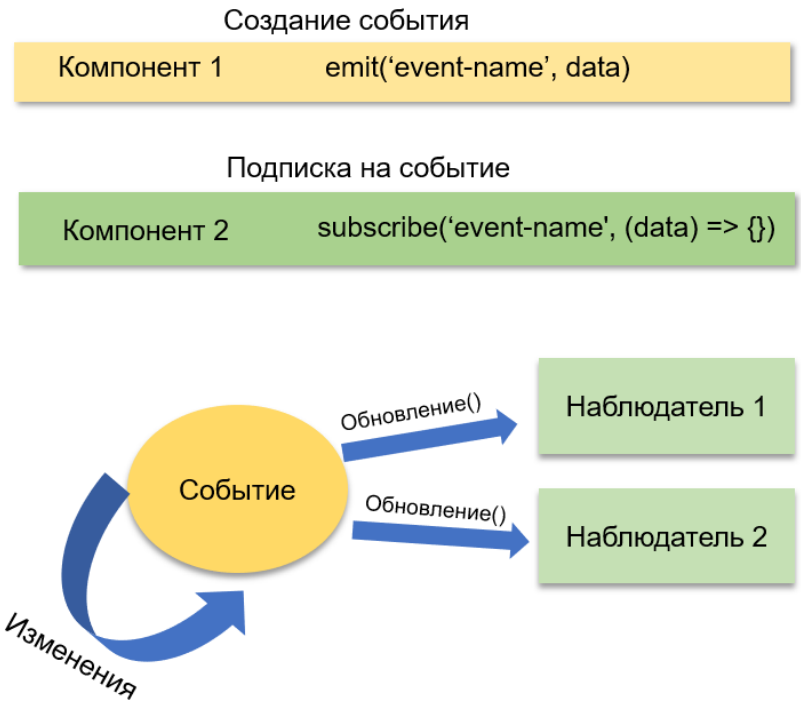


Рисунок 5 – Схема подписки на события используя класс UserEventEmitter

На рисунке 5 представлена схема работы подписки на события. При инициализации данного класса создается пустой объект, содержащий слушателей. Класс имеет два метода для испускания события и подписки на событие. При испускании события происходит вызов функций классов слушателей, подписанных на событие.

2.3 Проектирование и разработка класса страницы

Понятие страницы отсутствует у фреймворков Angular, React, Vue. Вместо них данную функцию выполняют компоненты. В данном фреймворке класс Page необходим для создания, группировки и инициализации

компонентов. Структура класса Page представлена на рисунке 6. Пользовательские страницы наследуются от данного класса.

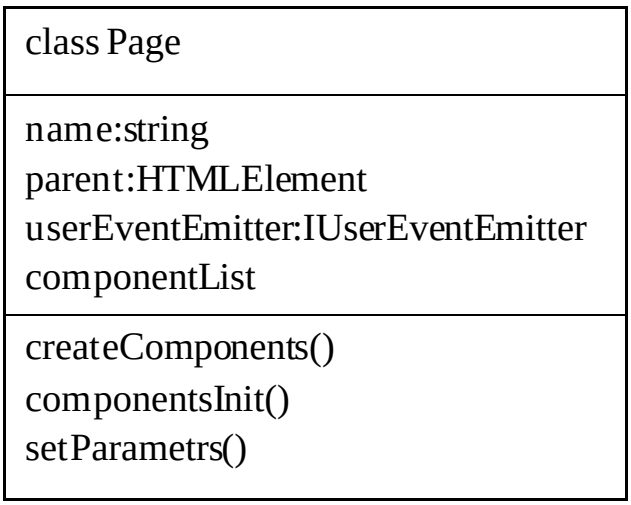


Рисунок 6 – Структура класса Page

В классе Router, который будет представлен далее установлена следующая последовательность при инициализации страницы:

1. Вызывается функция setParametrs для передачи на страницу родительского элемента, а также userEventEmitter.
2. Следом происходит вызов crateComponents для того, чтобы все указанные пользователем компоненты были созданы
3. Затем происходит инициализация компонентов, установка им переданных параметров на первом шаге, а затем их отрисовка на странице.

При наличии сохранённого состояния компонента, например в локальном хранилище браузера, при соответствующей настройке StateManager будет проверен на наличие состояние и если оно будет найдено, то будет загружено начальным значением в компонент за место значения по умолчанию.

2.4 Проектирование класса Router

При создании приложения создается экземпляр класса роутер, в который параметрами передается компонент навигации, а также массив страниц с соответствующими им адресами. Структура класса Router представлена на рисунке 7.

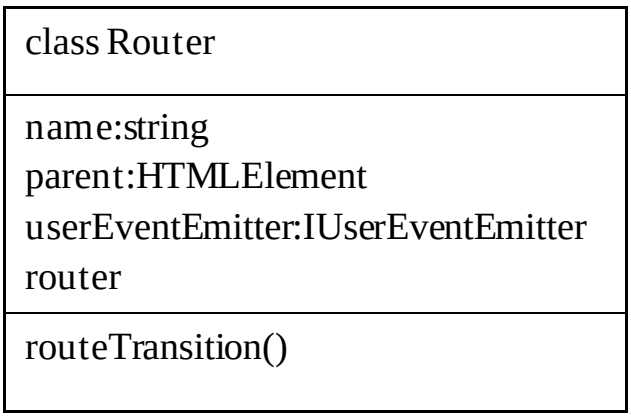


Рисунок 7 – Структура класса Router

В классе Router в конструкторе создается общий userEventEmitter для того, чтобы независимые компоненты могли взаимодействовать за счет отправки подписки на события. Также в данном классе создается общий для всех страниц родительский элемент, куда будут загружаться страницы при переходе по внутренним ссылкам приложения. Также в конструкторе на весь документ прикрепляется обработчик события окончания первоначальной загрузки HTML документа. Затем добавлен следующий вложенный обработчик события кликов. Данный обработчик проверяет наличие соответствующего дата атрибута у ссылки и при его наличии происходит переход на соответствующую страницу, запустив указанные в классе Page необходимые методы. Данный цикл повторяется при каждом нажатии на ссылки с соответствующими дата атрибутами.

2.5 Настройка прогрессивного веб-приложения и стратегии кеширования

Настройка прогрессивного веб-приложения или PWA начинается с создания и подключение манифеста. Манифест содержит базовые данные о имени, цветовой схеме, пространственной ориентации и иконках приложения. Следующим этапом следует создать и подключить Service Worker, файл, определяющий логику и стратегию кэширования данных. После его подключения следует определить события и добавить к ним соответствующие слушатели с необходимой логикой. В частности, в текущем приложении слушаются события установки, активации и получения данных.

Во время установки происходит кеширование указанных файлов и именование версии кэша. Во время активации проверяется по имени, соответствует ли имя кэша текущему имени и при несоответствии данные обновляются, а неактуальный кэш очищается. При получении данных в случае наличия интернета приоритет отдается новым данным, а при его отсутствии загружаются сохраненные данные.

2.6 Разбиение проекта на бандлы и оптимизация загрузки

Разбиение проекта на бандлы помогает решить важные задачи связанные с оптимизацией загрузки из различных точек входа приложения, а также позволяет сформировать грамотную стратегию кеширования файлов. В плане оптимизации разбиение на бандлы помогает подгружать только необходимые на данный момент данные, связанные с текущей точкой входа в приложение. После загрузки основных данных в данном случае рекомендуется лениво, не блокируя поток подгружать данные для остальных станиц.

В вопросе кеширования следует рассмотреть частоту изменения кода. К примеру, используемые библиотеки обновляются не часто поэтому их стоит отделить от данных, связанных с самим приложением. Также следует отмечать у всех файлов в бандле имя при помощи хэша, что позволит определять и безошибочно заменять старые файлы на актуальные.

2.7 Вспомогательные утилиты

Важным элементом фреймворка являются инструменты, с которыми он поставляется. Так в большинстве фреймворках вспомогательные инструменты написаны на языках не знакомых рядовым веб разработчикам, что делает невозможным для них редактировать и настраивать инструменты под свои нужды. Поэтому последовательным решением было создание утилит на платформе NodeJS, которые позволяют преобразовывать верстку сайтов в объекты с интерфейсом VNode. Также были созданные скрипты и команды позволяющие задать начальные правила и разобрать верстку на навигацию,

страницы и компоненты в результате чего из верстки можно сразу на выходе получить одностраничное приложение. Для создание начального проекта необходимо объявить файл настроек с примером содержимого представленным на рисунке 8.

Рисунок 8 – Пример файла конфигурации

Затем требуется в исходной верстке макета обозначить размещение компонентов с помощью дополнительной нотации:

<!--$#Component Name-->

<!--/$#Component Name-->

Затем утилиты смогут сформировать заданную файловую структуру и разбить верстку на компоненты и создать стартовые страницы с подключенными на ними компонентами, что на выходе генерирует стартовое одностраничное приложение, в которое требуется дописать интерактивность и логику.

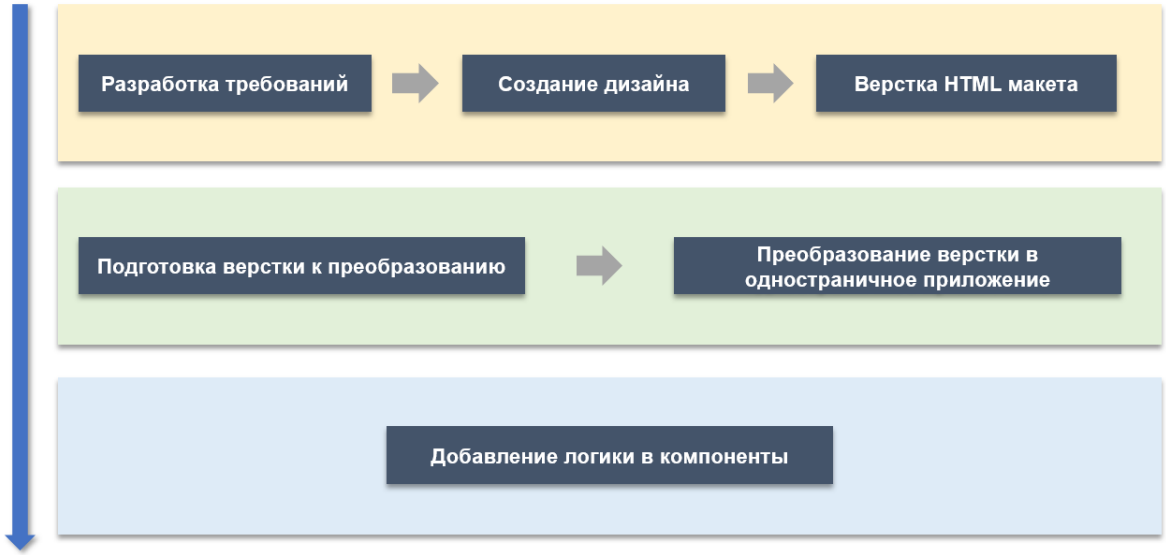


Рисунок 9 – Базовый алгоритм разработки приложения

Базовый алгоритм разработки приложения на разрабатываемом фреймворке представлен на рисунке 9.

3. Разработка демонстрационного приложения

В качестве демонстрационного приложения был выбран конфигуратор для выбора комплектующих персонального компьютера, который также позволяет сохранять, редактировать и удалять созданные сборки. Приложение состоит из 3 страниц: главная, конфигуратор и страница о компании. Как видно на рисунке 10 на главной странице расположен баннер с кнопкой для перехода в конфигуратор. А навигация на странице выполнена в виде меню, которое можно вызвать по нажатию на соответствующую кнопку.

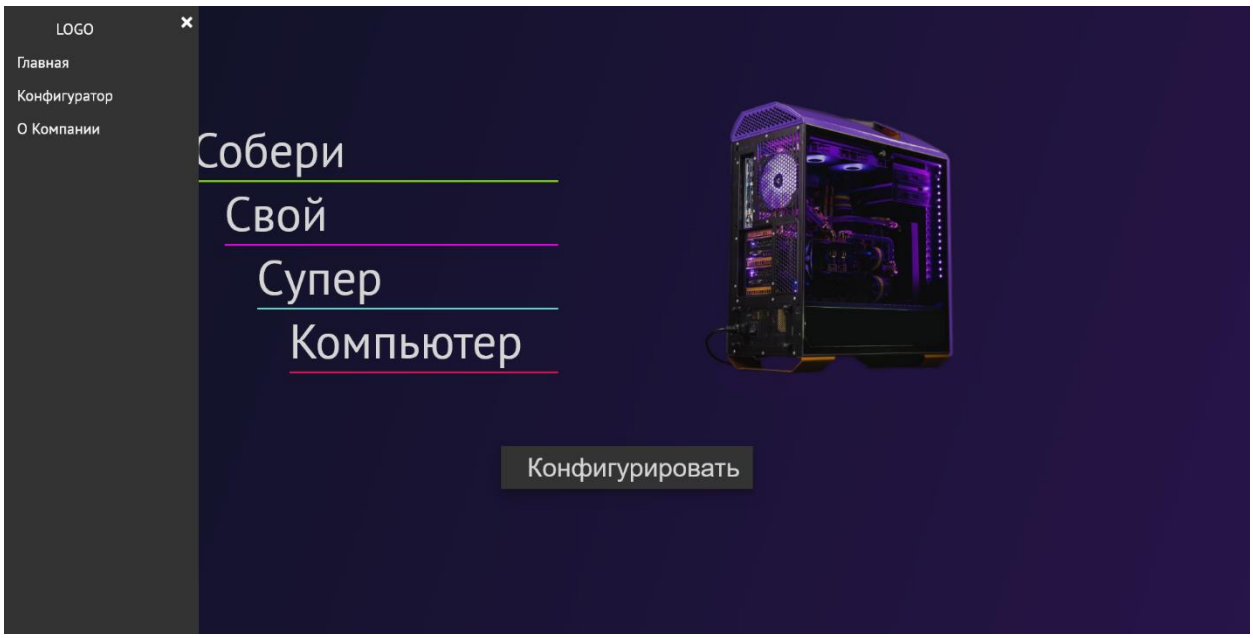


Рисунок 10 – Скриншот главной страницы и навигации демонстрационного приложения

На рисунке 11 представлен конфигуратор демонстрационного приложения. Особенностью приложения является возможность выбрать необходимые для работы программы, а также за счет интерполяции системных требований необходимых программ оценить системные требования на перспективу нескольких лет вперед.

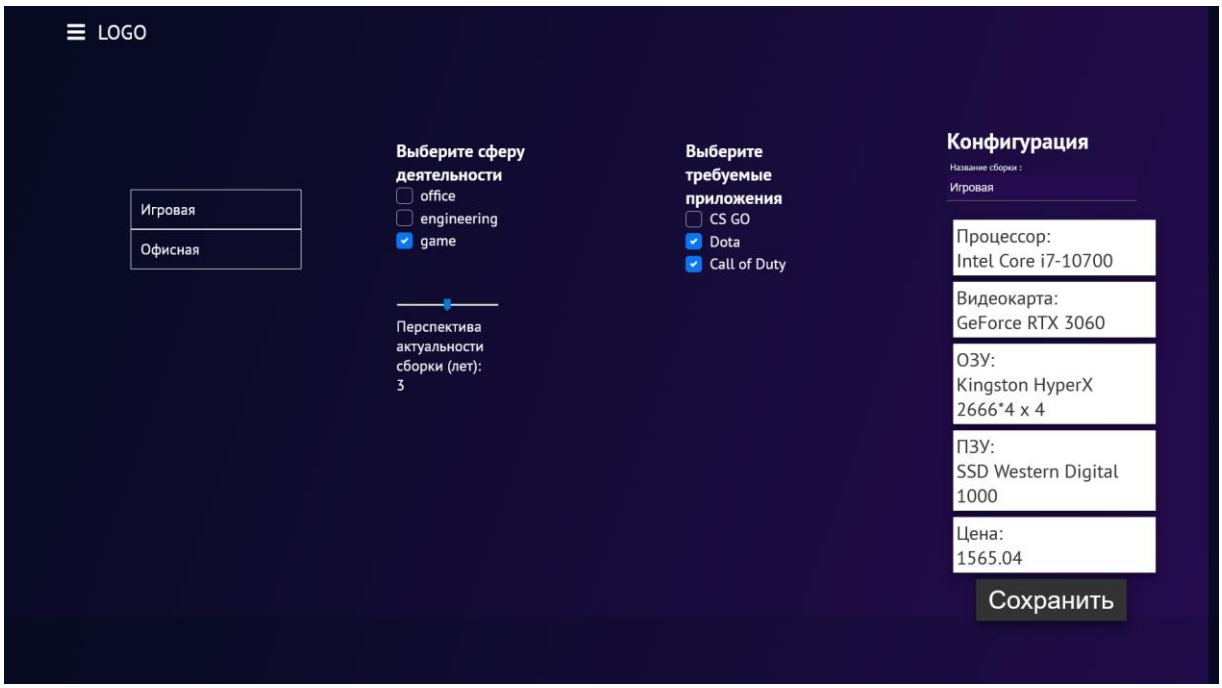


Рисунок 11 – Скриншот конфигуратора демонстрационного приложения

Диаграмма классов представлена в приложении В. Все три класса для страниц реализуют стандартные методы, описанные в родительском классе Page. Класс AboutPage инициализирует один компонент с названием AboutComponent, который содержит и отображает на страницу информацию о компании. Класс HomePage содержит компонент BannerComponent, содержащий и отображающий баннер с кнопкой перехода в конфигуратор. Класс ConfiguratorPage содержит следующие компоненты:

- ConfigurationsListComponent;
- ActivityFieldsComponent;
- ActualityDurationComponent;
- ProgramListComponent;
- ConfigurationComponent.

Данная страница содержит основную логику демонстрационного веб приложения. Так ConfigurationsListComponent при инициализации отображает список сохраненных конфигурация. А при нажатии на одну из них вызывается метод onChangeSelectedConfig который создает и испускает событие chooseConfig для подписки другими компонентами, остальные компоненты

реагируют на данное события вызывая собственный метод onChooseConfig. При изменении данных в ConfigurationComponent испускается событие updateConfigList в ответ на это происходит вызов одноименного метода и отображается актуальная информация.

Компонент ActivityFieldsComponent после инициализации отображает типы доступных для выбора программ. При пользовательском вводе генерируется и испускается событие событие fieldsOfActivity. Ответом на данное событие является вызов метода updateFields у компонента ProgramListComponent для отображения программ выбранного типа. После выбора пользователем требуемых программ компонент испускает событие fieldsOfPrograms в ответ на которое запускается метод showConfiguration в компоненте ConfigurationComponent. Данный метод опираясь на данные, присланные в событии, производит подбор соответствующих комплектующих методами chooseCPU, chooseGPU, chooseRAM, chooseMemory в совокупности с вводом пользователем данных о времени актуальности сборки, за счет интерполяции рассчитываются оценочные требования программ в бедующем и затем сборка отображается на странице, доступная для сохранения, или удаления если у она была ранее сохранена, о чем может свидетельствовать наличие параметра id у поля configuration.

Данное приложение доступно для установки как PWA приложение, что позволяет обеспечить пользовательский опыт нативного приложения.

4. Исследование производительности демонстрационного веб приложения

В качестве метрик были выбраны несколько критериев, таких как производительность демонстрационного приложения, сравнительный размер фреймворка, количество строк самого фреймворка. Количество строк рассчитывалось утилитой cloc [9].

Важным и непростым критерием является производительность веб приложения, которая представляет собой совокупность прогрессивных веб метрик, которые можно исследовать инструментом Google lighthouse. Данные метрики представлены на рисунке 12.

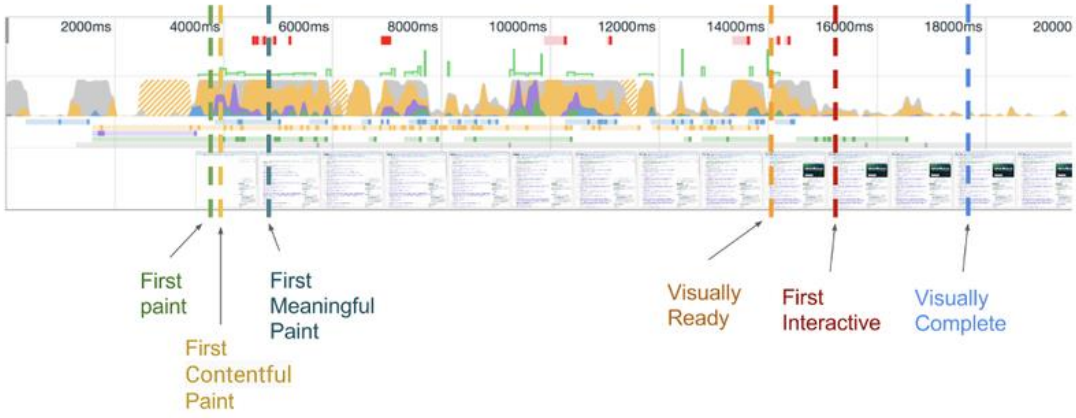


Рисунок 12 – Прогресивные веб метрики

Прогрессивные веб-метрики:

- First Contentful Paint(FCP) - время, за которое пользователь увидит какой-то контент, что-то отличное от просто пустой страницы. В результате пользователь понимает, грузится ли вообще страница после того, как он ввел URL и нажал на Enter. FCP вызывается в момент отрисовки текста (причем текст, ожидающий загрузки шрифта, тут не учитывается), изображения или какого-то иного контента. Время между FP и FCP варьируется от миллисекунд до секунд.
- Speed Index - Индекс скорости измеряет, насколько быстро контент визуально отображается во время загрузки страницы. Lighthouse сначала снимает видео загрузки страницы в браузере и вычисляет визуальный переход

между кадрами. Затем Lighthouse использует модуль Speedline Node.js для расчета индекса скорости.

- Largest Contentfull paint – LCP измеряет, когда на экран выводится самый большой элемент содержимого в области просмотра. Это примерно то время, когда основное содержимое страницы видно пользователям. См. Определение самой большой Contentful Paint для получения дополнительных сведений о том, как определяется LCP.
- Time To Interactive - TTI измеряет, сколько времени требуется странице, чтобы стать полностью интерактивной. Страница считается полностью интерактивной, если:
  - На странице отображается полезный контент, который измеряется с помощью First Contentful Paint,
  - Обработчики событий регистрируются для наиболее видимых элементов страницы,
  - Страница реагирует на действия пользователя в течение 50 миллисекунд. Visually Complete – событие вычисляется посредством снятия скриншотов страницы и сравнения их с помощью алгоритма Speed Index. Speed Index сам по себе вычисляет медиану между результатами Visually Complete. Чем меньше значение, тем лучше. 100% результат Visually Complete — финальная точка.
- Total blocking time -TBT измеряет общее время, в течение которого страница блокируется от ответа на действия пользователя, такие как щелчки мыши, касания экрана или нажатия клавиатуры. Сумма рассчитывается путем сложения блокирующей части всех длинных задач между First Contentful Paint и Time to Interactive. Любая задача, которая выполняется более 50 мс, - это долгая задача. Время после 50 мс - это блокирующая часть. Например, если Lighthouse обнаруживает задачу длительностью 70 мс, блокирующая часть будет составлять 20 мс.

На рисунке 13 показан результат исследования производительности веб приложения, в случае, когда точкой входа была главная страница. Результат

исследования производительности в случае, когда точкой входа являлась страница конфигуратора представлен на рисунке 14.

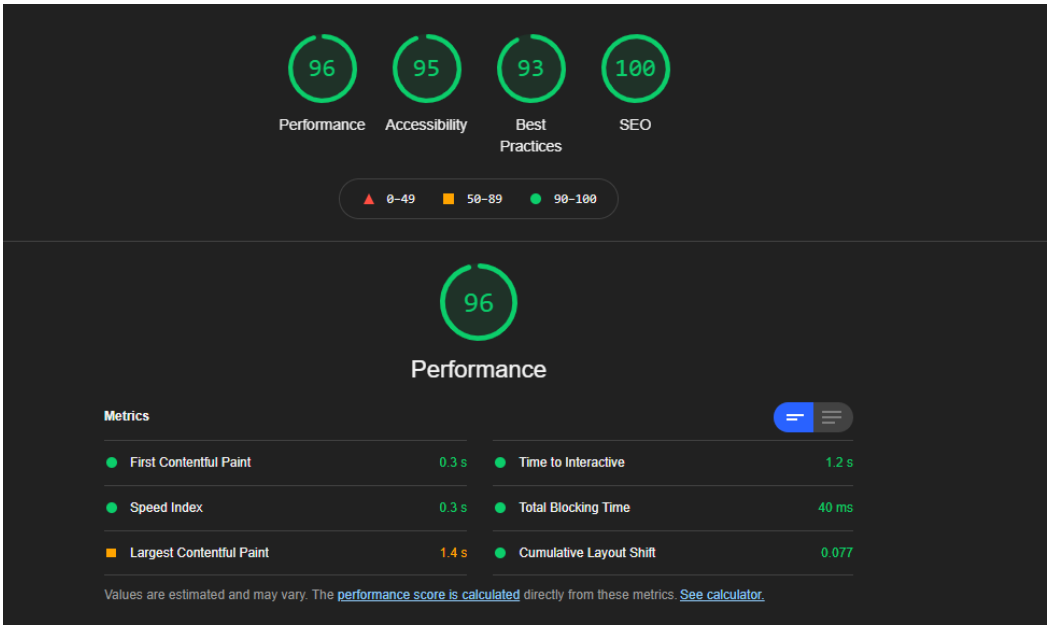


Рисунок 13 – Результаты исследования производительности главной страницы

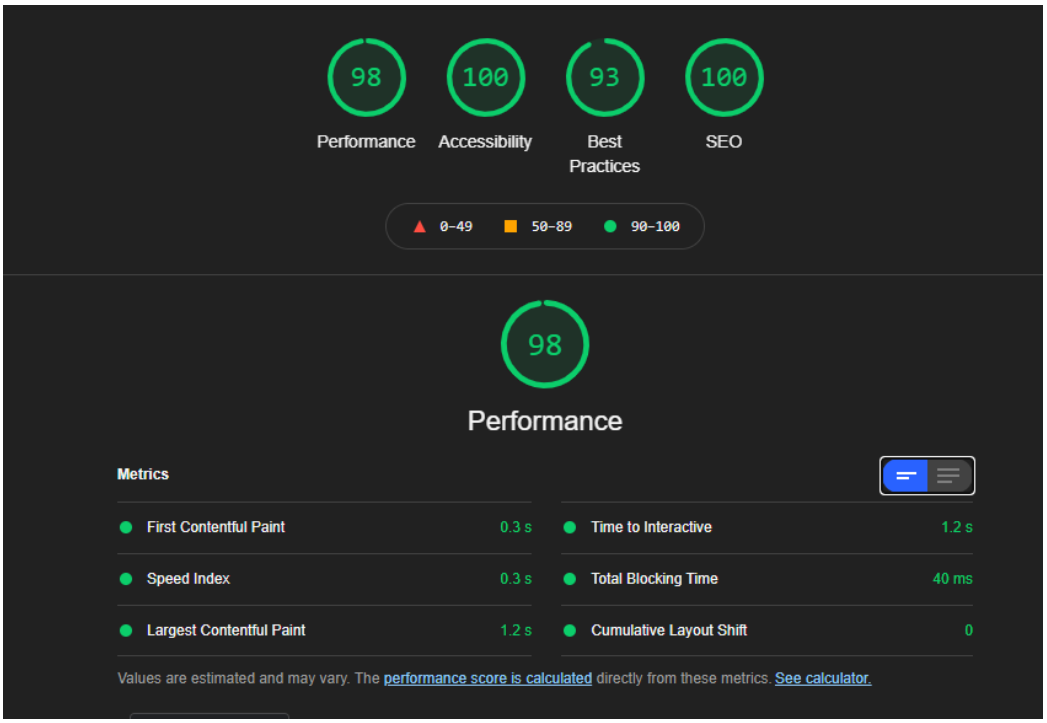


Рисунок 14 – Результаты исследования производительности страницы конфигуратора

Как видно из результатов тестирования производительности с использованием инструмента Google Lighthouse загрузка страниц происходит оптимизированно и с высокой скоростью. Обе страницы находятся в зеленой зоне результатов инструмента и оцениваются поисковиком Google как отличные.

5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

5.1 Потенциальные потребители результатов исследования

В данной работе представлен обзор современных технологий клиентской веб разработки, а также проектирование и разработка фреймворка позволяющего упростить и оптимизировать создание современных веб сайтов.

Продукт данной работы предназначен для того, чтобы специалисты смежных областей с клиенткой веб разработкой, такие как веб дизайнеры, верстальщики, бекенд разработчики и т.д. имели возможность создавать типовые сайты для компаний, не учитывая особенности реализаций современных клиентских веб технологий.

Целью раздела «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» является проектирование и создание конкурентоспособных разработок, технологий, отвечающих современным требованиям в области ресурсоэффективности и ресурсосбережения.

Достижение цели обеспечивается решением задач:

- оценка коммерческого потенциала и перспективности проведения научных исследований;
- планирование научно-исследовательских работ;
- определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования.

5.2 SWOT-анализ

SWOT – Strengths (сильные стороны), Weaknesses (слабые стороны), Opportunities (возможности) и Threats (угрозы) – представляет собой комплексный анализ научно-исследовательского проекта. SWOT-анализ применяют для исследования внешней и внутренней среды проекта. SWOT-анализ проекта приведен в приложении Г.

5.3 Планирование управления научно-техническим проектом

Для успешной реализации данного проекта были определены такие показатели как занятость каждого из участников, перечень всех проводимых работ, и сроки выполнения каждого из этапов. Для представления полученных показателей также введем следующие условные обозначения исполнителей проекта:

- НР –Савельев А.О. Доцент ОИТ, к.т.н, научный руководитель работы;
- ИМ –Ткачев М.С., инженер (магистрант)

Таблица 4 – Календарный план проекта

Код работы	Название	Длительность, дни	Дата начала работ	Дата окончания работ	Состав участников
1	Составление технического задания	2	09.02.2021	10.02.2021	НР
2	Литературный обзор предметной области	32	11.02.2021	14.03.2021	ИМ
3	Разработка требований к проекту	2	15.03.2021	16.03.2021	НР, ИМ
4	Проектирование проекта	17	17.03.2021	02.04.2021	ИМ
5	Разработка проекта	43	03.04.2021	16.05.2021	ИМ
6	Проведение тестирования сравнительной производительности	3	17.05.2021	19.05.2021	ИМ
7	Анализ результатов тестирования	5	20.05.2021	25.05.2021	ИМ
8	Подготовка отчетной документации	4	26.02.2021	31.05.2021	НР, ИМ

Таблица 5 – Календарный план-график проведения НИОКР по теме

№	Вид работ	Исп.	$T_{кi}$	Продолжительность выполнения работ											
				Фев.		март			апрель			май			
				2	3	1	2	3	1	2	3	1	2	3	
1	Составление технического задания	НР	2												
2	Литературный обзор предметной области	ИМ	32												
3	Разработка требований к проекту	НР, ИМ	2												
4	Проектирование проекта	ИМ	17												
5	Разработка проекта	ИМ	43												
6	Проведение тестирования сравнительной производительности	ИМ	3												
7	Анализ результатов тестирования	ИМ	5												
8	Подготовка отчетной документации	НР, ИМ	4												

▨ - руководитель ■ - инженер(магистрант)

5.4 Бюджет научно-технического исследования

5.4.1Материальные затраты

Таблица 6 – Матереные затраты НТИ

Наименования	Затраты, руб
Персональный компьютер	35 000
Расходные материалы (макулатура, периферийные устройства)	4000
Итого	**39 000**

5.4.2Основная заработная плата исполнителей темы

Инженер работает по 5-дневной рабочей неделе, а руководитель по 6-дневной.

Таблица 7 – Баланс рабочего времени

Показатели рабочего времени	Руководитель	Инженер
Календарное число дней	365	365
Количество нерабочих дней - выходные дни - праздничные дни	52 14	98 20
Потери рабочего времени - отпуск - невыходы по болезни	56 0	24 0
Действительный годовой фонд рабочего времени	243	223

Месячный должностной оклад работника:

$$З_м = З_{тс} \cdot k_p, \quad (1)$$

где $З_{тс}$ – заработная плата по тарифной ставке, руб.;

Месячный должностной оклад взят из документа «Оклады в ТПУ». Руководитель Савельев А. Е. – доцент ОИТ, ктн. $З_{ок}$ = 35111,55 руб. Инженер $З_{ок}$ = 27430,9 руб.

Расчёт основной заработной платы приведен в таблице 6.

Месячный должностной оклад определяется по формуле:

$$З_м = З_{ок} * К_р \quad (2)$$

где $З_{ок}$, руб. – должностной оклад

$k_р$ – районный коэффициент, равный 1,3 (для Томска).

$З_{дн}$ – среднедневная заработная плата работника, руб.

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн} = \frac{З_м \cdot М}{F_д} \quad (3)$$

где $З_м$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5-дневная неделя;

при отпуске в 56 раб. дней М=10,4 месяца, 6-дневная неделя;

Fд – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.

Основная заработная плата ($З_{осн}$) рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot T_р \quad (4)$$

где $З_{осн}$ – основная заработная плата одного работника;

$Т_р$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн.

Таблица 8 – Расчёт основной заработной платы

Исполнители	Оклад	$k_р$	$З_м$, руб	$З_{дн}$, руб.	$Т_р$, раб. дн.	$З_{осн}$, руб.
НР	35111,5	1,3	45644,9	1592,9	8	12743,2
ИМ	22695,0	1,3	29503,5	1337,8	106	141806,8

В настоящую статью включается основная заработная плата научных и инженерно-технических работников, рабочих макетных мастерских и опытных производств, непосредственно участвующих в выполнении работ по данной теме. Величина расходов по заработной плате определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и тарифных ставок.

5.4.3Дополнительная заработная плата исполнителей темы

Расчет дополнительной заработной платы ведется по следующей формуле:

$$З_{доп} = k_{доп} \cdot З_{осн} \quad (5)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,10 – 0,15).

Таблица 9 – Заработная плата исполнителей НТИ

	Руководитель	Инженер(магистрант)
Основная заработная плата, руб	12 743,2	141 806,8
Дополнительная зп, руб	1 529,2	17 016,8
Зп исполнителя, руб	14 272,4	158 823,6
Итого, руб	173 096,0	

5.4.4Отчисления во внебюджетные фонды

Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы:

$$З_{внеб} = k_{внеб} \cdot (З_{осн} + З_{доп}) \quad (6)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды, равный 30% (налоговая нагрузка) по тарифу на 2021 год (статья 425 НК РФ).

Таблица 10 – Отчисления во внебюджетные фонды

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**
НР	12 743,2	141806,8
ИМ	1 529,2	17 016,8
Коэффициент отчислений во внебюджетные фонды	0,3	
Итого 4 281,7 + 47 647,1 = 51 928,8		

5.4.5Накладные расходы

Расчет затрат на электроэнергию, потраченную в ходе выполнения проекта на работу используемого оборудования, рассчитываемые по формуле:

$$C_{эл.об} = P_{об} * t_{об} * C \quad (7),$$

где $P_{об}$ – мощность, потребляемая оборудованием, кВт; $t_{об}$ – время работы оборудования, час; С – тариф на 1кВт·час. Тариф на электроэнергию равен 6,59 руб за 1 кВт.ч. При этом время работы оборудования рассчитывается на основе данных таблицы 2 (Трд = 106 дней), продолжительность рабочего дня равна 8 часов (Трд = 106 *8 = 848 ч). В свое очередь мощность, потребляемая оборудованием, определяется по формуле: $P_{ОБ}$ для ПК – 0,5 кВт.

$$C_{эл.об} = 0{,}5 * 848 * 6{,}59 = 2794{,}2 \text{ руб}$$

5.4.6Формирование бюджета затрат научно-исследовательского проекта

Рассчитанная величина затрат научно-исследовательской работы (темы) является основой для формирования бюджета затрат проекта, который при формировании договора с заказчиком защищается научной организацией в качестве нижнего предела затрат на разработку научно-технической продукции.

Таблица 11 – Расчет бюджета затрат НТИ

Наименование статьи	Сумма, руб.	Примечание
1. Материальные затраты	39 000	Пункт 3.1
2. Затраты по основной и дополнительной заработной плате исполнителей темы	173 096,0	Пункт 3.3
3. Отчисления во внебюджетные фонды	51 928,8	Пункт 3.4
4. Накладные расходы	2794,2	Пункт 3.5
5. Бюджет затрат НТИ	**266 819,4**	Сумма ст. 1- 5

5.5 Определение ресурсной, финансовой, бюджетной, социальной и экономической эффективности исследования

Проектом данной работы является фронтенд фреймворк, позволяющий разработчику абстрагироваться от платформы разработки за счет использования технологии виртуального DOM дерева и представления страницы в виде JavaScript объекта. Данный подход позволяет разработчику изменять виртуальное представления страницы привычными для многих языков программирования методами, не учитывая особенности платформы. Также отсутствует необходимость разработчику иметь дело с двумя языками HTML и JavaScript одновременно, что упрощает и оптимизирует рабочий цикл разработки.

В настоящее время отсутствуют отечественные разработки в данной сфере, которые могли бы оказать конкуренцию лидерам сферы, которыми являются два фреймворка: React от компании Facebook и Angular 2 от компании Google.

Первым критерием сравнения текущего проекта с конкурентными решениями является снижение стоимости разработки. Так как конкурентные решения имеют собственную специфичную архитектуру и регулярно изменяются, то при поиске и найме сотрудников для разработки к ним нужно предъявлять повышенные требования, которые необходимо компенсировать зарплатой. Согласно данным сайта HH.ru количество вакансий для React вдвое превышает Angular, поэтому найти разработчика становится сложнее. В случае текущего проекта к кандидату предъявляются общие требования к навыкам программирования. Также наличие специфичной архитектуры влияет на критерий порога входа. Из-за масштабности Angular, его порог входа является достаточно высоким.

В плане критерия надежности следует понимать, что конкурентные решения являются рабочими разработками крупных частных компаний и созданы в первую очередь для нужд самих компаний. Пока нужды компаний схожи с общими нуждами, данные решения могут применятся повсеместно,

однако их развитие и направление диктуется компанией, а не сферой разработки, потребностями сообщества и стандартов. Поэтому данные решения в долгосрочной перспективе являются ненадежными по сравнению с проектом, цель которого удовлетворение нужд сообщества разработчиков.

В плане критерия безопасности все три представленных проекта представляют из себя проекты с открытым исходным кодом, поэтому считается что такие проекты могут быть исследованы самими пользователями на наличие уязвимостей, однако как показывает практика, несмотря на открытый исходный код проанализировать настолько крупные проекты на наличие уязвимостей становится невозможным. Так количество строк кода у Angular составляет 118523, у React – 67163 строки. Количество строк в текущем разрабатываемом проекте составляет 6730, что значительно меньше, чем у конкурентов, что позволяет разработчику проанализировать код на наличие уязвимостей.

Производительность платформ можно исследовать по критерию First Contentful Paint(FCP) – время, за которое пользователь увидит какой-то контент, что-то отличное от просто пустой страницы. Данный параметр был измерен в схожих условиях и составил 900мс для angular, 700мс для текущего проекта и React.

В итоге ключевыми критериями при оценке проектов являются снижение стоимости разработки, надежность, безопасность, производительность, а также удобство в эксплуатации и порог входа в технологию.

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный финансовый показатель разработки определяется как:

$$I_{\text{финр}}^{\text{исп.}i} = \frac{\Phi_{\text{p}i}}{\Phi_{\max}} \quad (8)$$

где $I_{\text{финр}}^{\text{исп.i}}$ – интегральный финансовый показатель разработки;

Φ_{pi} – стоимость i-го варианта исполнения;

$Ф_{max}$ – максимальная стоимость исполнения научно-исследовательского проекта (в т.ч. аналоги).

Таблица 12 – Сравнительная оценка характеристик вариантов исполнения проекта

Объект исследования / Критерии	Весовой коэффициент параметра	Текущий проект	Аналог 1	Аналог 2
1. Способствует снижению стоимости разработки	0,10	5	3	2
2. Удобство в эксплуатации	0,20	4	3	3
3. Порог входа	0.10	4	4	3
4. Безопасность	0,20	5	4	3
5. Надежность	0,20	5	4	4
6. Производительность	0,10	5	5	4
7. Применение в проектах	0.10	1	4	3
ИТОГО	1,00			

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно определить следующим образом:

$$I_{pi} = \sum a_i \cdot b_i \quad (9)$$

где I_{pi} – интегральный показатель ресурсоэффективности для i-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i^a, b_i^p – балльная оценка i-го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования.

Аналог 1 – React

Аналог 2 – Angular

$I_{р-исп1}$=5*0,10+4*0,20+4*0,10+5*0,20+5*0,20+5*0.10+1*0.10=4,30;

$I_{р-исп2}$=3*0,10+3*0,20+4*0,10+4*0,20+4*0,20+5*0.10+4*0.10=3,80;

$I_{р-исп3}$=2*0,10+3*0,20+3*0,10+3*0,20+4*0,20+4*0.10+3*0.10=3,20.

Интегральный финансовый показатель разработки можно оценить за счет анализа объёма проекта, приняв ценность и трудоемкость строк равными.

Таблица 13– Сравнительная эффективность разработки

№ п/п	Показатели	Исп.1	Исп.2	Исп.3
1	Интегральный финансовый показатель разработки	0,06	0,57	1
2	Интегральный показатель ресурсоэффективности разработки	4,30	3,80	3,20
3	Интегральный показатель эффективности	71	6,67	3,2
4	Сравнительная эффективность вариантов исполнения	1	10,74	22,19

Сравнение значений интегральных показателей эффективности позволяет судить о приемлемости существующего варианта решения поставленной в магистерской диссертации технической задачи с позиции финансовой и ресурсной эффективности.

6. Социальная ответственность

В данной работе представлен обзор современных технологий клиентской веб разработки, а также проектирование и разработка фреймворка позволяющего упростить и оптимизировать создание современных веб сайтов.

Продукт данной работы предназначен для того, чтобы специалисты смежных областей с клиенткой веб разработкой, такие как веб дизайнеры, верстальщики, бекенд разработчики и т.д. имели возможность создавать типовые сайты для компаний, не учитывая особенности реализаций современных клиентских веб технологий.

Разработка осуществлялась в офисном помещении за настольным персональным компьютером.

Так как основная работа при написании диссертации, разработка ПО, связана с компьютером, или персональной электронной вычислительной машиной (ПК), то в рамках текущего раздела целесообразным является рассмотрение следующих вопросов:

- Выявление и изучение вредных и опасных производственных факторов при работе с ПК;
- Определение способов снижения действия описанных факторов до безопасных пределов или по возможности до полного их исключения;
- Безопасность окружающей среды;
- Безопасность в чрезвычайных ситуациях (ЧС), которые могут возникнуть при эксплуатации ПК.

6.1 Правовые и организационные вопросы обеспечения безопасности

В Российской федерации трудовые отношения между работником, работодателем и государством регулируются Трудовым кодексом Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021) [32]. Данный нормативно-правовой акт регламентирует права и обязанности между сторонами трудовых взаимоотношений, нормы рабочего времени, порядок оплаты труда и компенсаций и т.д. Согласно трудовому кодексу, продолжительность рабочего

дня не должна превышать 24 часов в неделю для работников до 16 лет, 35 часов для работников в возрасте от 16 до 18 лет или являющихся инвалидами I или II групп. В остальных случаях рабочая неделя должна длиться не более 40 часов. Так количество рабочих часов в неделю не превышало 40 часов, работа велась по будним дням, в течение рабочего дня предоставлялся часовой перерыв, который к рабочему времени не относится [32].

Персональные данные работника – информация, необходимая работодателю в связи с трудовыми отношениями и касающаяся конкретного работника. (Статья 85 ТК РФ.) Обработка персональных данных работника – получение, хранение, комбинирование, передача или любое другое использование персональных данных работника. Данные о работнике, предоставленные работодателю, обрабатываются только с согласия работника. Запрещена дискриминация по любым признакам и принудительный труд. Также ответственные лица не имеют права получать и обрабатывать персональные данные работника о его расовой, национальной принадлежности, политических взглядах, религиозных или философских убеждениях, частной жизни. В случаях, непосредственно связанных с вопросами трудовых отношений, в соответствии со статьей 24 Конституции Российской Федерации ответственные лица вправе получать и обрабатывать данные о частной жизни работника только с его письменного согласия [1].

Деятельность при выполнении магистерской диссертации также связана с работой за компьютером (или ПК). Основным документом, регулирующим условия и организацию работы с ПК, является ГОСТ 22269-76. Система "Человек-машина". Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования и ГОСТ 12.2.032-78 ССБТ. «Рабочее место при выполнении работ сидя. Общие эргономические требования» [33], которые включают ряд требований к ПК и организации рабочего места, а также к факторам, оказывающим на пользователя ПК опасное и вредное влияние.» [34].

Основными эргономическими требованиями к помещению, в котором выполняется работа с использованием компьютерной техники являются [34]:

1. Помещение должно иметь естественное и искусственное освещение (согласно СП 52.13330.2016);

2. Рабочие места по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева;

3. Площадь на одно рабочее место пользователя персонального компьютера на базе плоских дискретных экранов (жидкокристаллические, плазменные) должна составлять 4,5 м^2;

4. При размещении рабочих мест с компьютерами расстояние между рабочими столами с мониторами должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов –не менее 1,2 м;

5. Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы (учитывая расположение оборудования, документации и пр.);

6. Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм, но не ближе 500 мм;

7. В помещениях с компьютерами ежедневно должна проводиться влажная уборка.

Рабочее помещение представлено комнатой площадью 25,2 м^2, где на одно рабочее место приходится площадь 5 м^2. В помещении имеется естественное и искусственное освещение. Окно с естественным освещением расположено справа от рабочего места. Фактические значения эргономических параметров рабочего стола соответствуют нормирующим требованиям. Так высота рабочего нерегулируемого по высоте составляла 700 мм, мониторы расположены от глаз на расстоянии 600 мм. Кроме того, при работе с ПК было использовано рекомендуемое СанПиНом время перерывов 10-15 минут после

каждых 45-60 минут работы с использованием комплекса профилактических мероприятий: упражнения для глаз и физкультурные минуты.

6.2 Производственная безопасность

Согласно ГОСТ 12.0.003-2015 «Опасные и вредные производственные факторы. Классификация» все производственные факторы классифицируются по группам элементов: физические, химические, биологические и психофизические. Для данной работы целесообразно рассмотреть физические и психофизические вредные и опасные факторы производства, характерные как для рабочей зоны инженера-программиста, как разработчика рассматриваемой в данной работе системы. Так как психофизические факторы, связанные с нормированием рабочего дня и отдыха, а также с предоставлением комфортного рабочего места, были описаны в пункте 2 данного раздела, то далее будут рассматриваться физические факторы [35].

Таблица 14 – Возможные опасные и вредные факторы

Факторы (ГОСТ 12.0.003-2015)	Этапы работ		Нормативные документы
	Разраб.	Экспл.	
1. Отклонение показателей микроклимата	+	+	1. СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений. 2. СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95* 3. ГОСТ 12.1.003-2014 ССБТ. Шум. Общие требования безопасности. 4. ГОСТ 12.1.006-84 ССБТ. Электромагнитные поля радиочастот. Общие требования безопасности. 5. ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов.
2. Недостаточная освещенность рабочей зоны	+	+	
3. Превышение уровня шума	+	+	
4. Повышенный уровень электромагнитных излучений	+	+	
5. Повышенное значение напряжение в электрической цепи, замыкание которой может произойти через тело человека	+	+	

В случае разработки ПО, где объектом является рабочее место, включая персональный компьютер и помещение, среди таких вредных воздействий можно указать: микроклимат помещения, неправильное освещение, шум, электромагнитное излучение, опасность поражения электрическим током

6.2.1Микроклимат рабочего места

В соответствии с СанПиН 2.2.4.548 – 96 «Гигиенические требования к микроклимату производственных помещений» параметрами, характеризующими микроклимат, являются: температура воздуха; температура поверхностей; относительная влажность воздуха; скорость движения воздуха [36].

Таблица 15 – Параметры микроклимата

Период года	Температура воздуха, °С		Относительная влажность воздуха, %	Скорость движения воздуха, м\с	
	Мин.	Макс.	Макс.	Мин.	Макс.
холодный	20-22	24-25	15-75	0,1	0,1
теплый	21-23	25-28	15-75	0,1	0,2

Для поддержания оптимальных значений микроклимата предусмотрены следующие средства: центральное отопление, вентиляция (искусственная и естественная), искусственное кондиционирование. Для отслеживания значений параметров используется термометр, совмещенный с гигрометром.

6.2.2Шум

Шум — это совокупность звуков различной интенсивности и частоты, беспорядочно сочетающихся и изменяющихся во времени. Звук — механическое колебание упругой среды (воздушной) с частотой от 16 до 20000 Гц. Звуковая волна несет с собой звуковое давление, измеряемое в Ньютонах на м (Н/м) и звуковую энергию, измеряемую в ваттах на м (Вт/м) [37].

Шумовая болезнь, как и вибрационная, — это сложный симптомокомплекс функциональных и органических изменений в организме и было бы неправильно отдавать первенство изменению функции органа слуха. Общее действие проявляется прежде всего при воздействии на ЦНС,

проявляющуюся в резком замедлении всех нервных реакций, сокращении времени активного внимания, снижении работоспособности и качества работы. Даже производственный травматизм на шумных предприятиях выше, чем на бесшумных [38].

Предельно допустимые уровни звука и эквивалентные уровни звука на рабочих местах для трудовой деятельности разных категорий тяжести и напряженности в дБА. Допустимый уровень шума ограничен ГОСТ 12.1.003-2014 ССБТ и СН 2.2.4/2.1.8.562–96. Уровень шума на рабочем месте программистов не должен превышать 50дБА, а в залах обработки информации на вычислительных машинах 65дБА.

Согласно П 4 ГОСТ 12.1.029 – 80. средства и методы коллективной защиты в зависимости от способа реализации подразделяются на [39]:

- Акустические (звукоизолирующие кожухи, кабины, акустические экраны, выгородки, звукопоглощающие облицовки, объемные поглотители звука и др.). Во всех случаях, когда на кожух могут передаваться вибрации от источника шума, стенки кожуха следует покрывать вибродемпфирующим материалом мастичного типа.
- Архитектурно-планировочные (создание шумозащищенных зон, рациональное размещение оборудования рабочих мест, рациональные акустические решения планировок зданий и генеральных планов объектов и др.).
- Организационно-технические (применение малошумных технологических процессов и машин, оснащение шумных машин средствами дистанционного управления и автоматического контроля, использование рациональных режимов труда и отдыха работников на шумных предприятиях и др.).

Средства индивидуальной защиты от шума подразделяют на:

- противошумные наушники, закрывающие ушную раковину;
- противошумные вкладки, перекрывающие наружный слуховой канал;
- противошумные шлемы и каски;
- противошумные костюмы.

Наушники могут быть независимыми, имеющие жесткое или мягкое оголовье или встроенными в головной убор.

В рассматриваемом рабочем помещение уровень шума является допустимым и не превышает значений, установленных нормами, и составляет не более 50 дБА. Кроме того, для уменьшения шума производятся: регулярное техническое обслуживание компьютеров и другой техники в помещении.

6.2.3Освещенность рабочей зоны

Нормы освещённости на рабочих местах производственных помещений при искусственном освещении регламентируются СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95. При разработке ПО подразумевает зрительный тип работы, в следствии чего важна организация правильного освещения. Пренебрежение данным фактором может привести к профессиональным болезням зрения. В рабочем помещении сочетаются естественное освещение (через окна) и искусственное освещение (использование ламп при недостатке естественного освещения) [39].

Разряд зрительных работ программиста относится к категории III г (высокой точности), параметры искусственного освещения указаны в таблице 3.

Таблица 16 – Нормативные значения освещенности

Характеристика зрительной работы	Наименьший или эквивалентный размер объекта различения, мм	Разряд зрительной работы	Подразряд зрительной работы	Контраст объекта с фоном	Характеристика фона	Искусственное освещение		
						Освещенность, лк		
						При системе комбинированного освещения		При системе общего освещения
						Всего	В том числе от общего	
Высокой точности	От 0,3 до 0,5	III	г	Средний и большой	Светлый и средний	400	200	200

Согласно СП 52.13330.2016 «Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95» уровень освещения на поверхности рабочего стола при работе с ПК должен быть в диапазоне от 300 до 500 лк.

Работа над текущей работой велась в помещении прямоугольной формы с параметрами: длина помещения А = 6 м, ширина помещения В = 4,2 м, высота помещения Н = 3 м, Площадь помещения S = 25,2 м^2. Коэффициент отражения стен Rc= 30 %, потолка Rn= 50 %. Коэффициент запаса k=1,5, коэффициент неравномерности Z= 1,1. Выбираем светильники типа ШОД, λ= 1,2. Приняв h_c= 0,5 м, определяем расчетную высоту: $h=H-h_c-h_{рп}=3-0,5-0,7=1,8$ м

Расстояние между светильниками:

$L=\lambda*h=1,2*1,8=2,16$ м

Расстояние от крайнего ряда светильников до стены: $L/3=0,72$ м

Определяем количество рядов светильников и количество светильников в ряду:

$$n_{\text{ряд}}=\frac{(B-\frac{2}{3}*L)}{L}+1=\frac{(4,2-\frac{2}{3}*2,16)}{2,16}+1=2,28\approx 2,$$

$$n_{\text{СВ}}=\frac{(A-\frac{2}{3}*L)}{l_{\text{СВ}}+0,5}=\frac{(6-\frac{2}{3}*2,16)}{1,2+0,5}=2,68\approx 3,$$

Размещаем светильники в два ряд. В ряду устанавливается 3 светильника типа ШОД 2-40 , расстояние в ряду между светильниками составило 0,4 м.

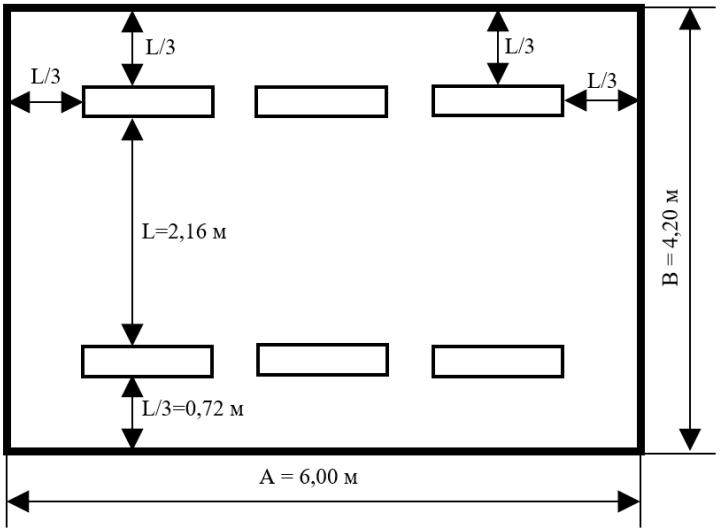


Рисунок 15 – План помещения и размещения светильников

Находим индекс помещения

$$i=\frac{S}{h*(A+B)}=\frac{25,2}{1,8*(6+4,2)}=1,37,$$

Определяем коэффициент использования светового потока: η= 0,38.

Определяем потребный световой поток ламп в каждом из рядов

$$\Phi = \frac{E_H * S * K_S * Z}{N_л * \eta} = \frac{300 * 25{,}2 * 1{,}5 * 1{,}1}{12 * 0{,}38} = 2735, лм$$

По таблице выбираем ближайшую стандартную лампу ЛТБ 2850 лм 40 Вт. Делаем проверку выполнения условия:

$$-10\% \leq \frac{\Phi_{л.станд} - \Phi_{л.расч}}{\Phi_{л.станд}} * 100\% \leq +20\%,$$

$$-10\% \leq -4{,}2\% \leq +20\%,$$

Определяем электрическую мощность осветительной установки P= 12*40 = 480 Вт.

6.2.4Электромагнитное излучение

Персональные компьютеры являются источниками электромагнитных волн, то есть распространяющихся в пространстве возмущений электромагнитного поля (ЭМП). Все электрические приборы излучают такие волны, однако наибольший вклад вносит экран монитора. При определённых уровнях такие поля оказывают вредное влияние на человека: нарушение функционального состояния нервной и сердечно-сосудистой систем, это проявляется в повышенной утомляемости, понижении качества выполнения рабочих операций, изменении кровяного давления и пульса [40]

Ввиду того, что используется жидкокристаллический монитор, то контроль мягкого рентгеновского излучения не осуществляется. Допустимые значения излучения показаны в таблице 4 с учётом ГОСТ 12.1.006-84 ССБТ. «Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля» [41].

Таблица 17 – Временные допустимые уровни ЭМП, создаваемых ПК

Наименование параметров		**ВДУ ЭМП**	**ВДУ ЭМП**
Напряженность электрического поля	в диапазоне частот 5 Гц – 2 кГц	25 В/м	27 В/м
	в диапазоне частот 2 кГц – 400 кГц	2,5 В/м	2,5 В/м
Электростатический потенциал экрана видеомонитора		500 В	490 В

Нормы допустимых уровней напряженности электромагнитных полей зависят от времени пребывания человека в контролируемой зоне. Присутствие

персонала на рабочем месте в течение 8 ч допускается при напряженности, не превышающей 5 кВ/м.

Основной способ снижения вредного воздействия – это увеличение расстояния от источника (не менее 50 см от пользователя). Защитой от воздействия электромагнитного поля токов промышленной частоты являются стационарные или переносные заземленные экранирующие устройства. На предприятии электромагнитное излучение не превышает 5 кВ/м, поэтому при работе за компьютером специальные экраны и другие средств индивидуальной защиты применены не были.

6.2.5Электробезопасность

Среди распространенных опасностей в рабочей зоне находится и поражение электрическим током. Опасность поражения определяется величиной тока проходящего через тело человека или напряжением прикосновения. При получении человеком разряда электрического тока могут быть получены электротравмы, электрические удары и даже летальный исход. ГОСТ 12.1.038-82 ССБТ определяет предельно допустимые значения напряжения прикосновения и тока на рабочем месте. Так для переменного тока с частотой 50Гц напряжение прикосновение составляет не более 2 В при токе 0,3 мА. А в случае постоянного тока не более 8 В и 1 мА соответственно [41].

Основным источником угрозы поражения электрическим током является персональный компьютер. Во избежание несчастных случаев сотрудники в обязательном порядке должны проходить соответствующий инструктаж.

Не следует работать на персональном компьютере при:

- повышенной влажности (относительная влажность воздуха более 75%);
- высокой температуре (более 35 °С);
- наличие токопроводящей пыли, токопроводящих полов и возможности одновременного соприкосновения к имеющим соединение с землёй металлическим элементам и металлическим корпусом электрооборудования.

Персональный компьютер питается от сети 220 В переменного тока с частотой 50 Гц. Это напряжение опасно для жизни, поэтому обязательны следующие меры предосторожности:

•перед началом работы нужно убедиться, что выключатели и розетка закреплены и не имеют оголённых токоведущих частей;

•при обнаружении неисправности оборудования и приборов необходимо, не делая никаких самостоятельных исправлений, сообщить человеку, ответственному за оборудование.

Чтобы избежать поражения электрическим током, необходимо защитить все токоведущие части от возможных прикосновений, а металлические корпуса должны быть заземлены.

Таким образом, все требования при работе с ПК были выполнены, так как все необходимые показатели норм находятся в допустимых пределах [42].

6.3 Экологическая безопасность

В текущей работе выявлен предполагаемый источник загрязнения окружающей среды, а именно воздействие на литосферу в результате образования отходов при поломке предметов вычислительной техники и оргтехники. Утилизация компьютерного оборудования является достаточно сложной.

Вышедшее из строя ПК и сопутствующая оргтехника относится к IV классу опасности и подлежит специальной утилизации. Непосредственная переработка большей части компонентов включает в себя их сортировку, последующую гомогенизацию и отправку для повторного использования, т.е. с предварительным помолом или переплавкой. При этом она должна соответствовать процедуре утилизации ГОСТ Р 53692-2009 Ресурсосбережение. Обращение с отходами. Этапы технологического цикла отходов [43].

Люминесцентные лампы представляют собой «чрезвычайно опасные» виды отходов. Содержание ртути в любых люминесцентных лампах составляет от трех до пяти миллиграмм ртути. С учетом этого необходимо обеспечивать

определенные условия хранения, их эксплуатации и утилизации. Согласно санитарным нормам хранить ртутьсодержащие отходы необходимо в специальных герметичных контейнерах, доступ посторонним лицам к таким контейнерам должен быть запрещен. Транспортировка ламп на полигоны складирования должна выполняться организациями, которые специализируются на утилизации опасных отходов. Категорически запрещено размещение таких отходов, как люминесцентные лампы на полигонах твердых бытовых отходов [43].

В ходе работы также создает бытовой мусор (канцелярские, пищевые отходы, искусственные источники освещения), который должен быть утилизирован в соответствии с определенным классом опасности или переработан, чтобы не оказывать негативное влияние на состояние литосферы

6.4 Безопасность в чрезвычайных ситуациях

Самым распространенным чрезвычайным обстоятельством в офисе является пожар. Такое рабочее место относится к категории «В» (пожароопасные), так как в данном помещении присутствует пыль, вещества и материалы, способные при взаимодействии с воздухом гореть [44].

Возникновение пожара может произойти по нескольким факторам:

- Возникновением короткого замыкания в электропроводке вследствие неисправности самой проводки или электросоединений и электрораспределительных щитов;
- Возгоранием устройств вычислительной аппаратуры вследствие нарушения изоляции или неисправности самой аппаратуры;
- Возгоранием мебели или пола по причине нарушения правил пожарной безопасности, а также неправильного использования дополнительных бытовых электроприборов и электроустановок; возгоранием устройств искусственного освещения.

Методы борьбы с пожарами предусматривают:

- Инструктажи, наличие планов эвакуаций, правильный монтаж и эксплуатация оборудования, правильное содержание зданий и территорий, обучение правилам техники безопасности, издание специальных инструкций и плакатов [45]
- Соблюдение противопожарных правил, исключение образования горючей среды, применение трудно сгораемых материалов
- Предусмотренные средства сигнализации, огнетушители, автоматические стационарные системы тушения пожаров, своевременная эвакуация.

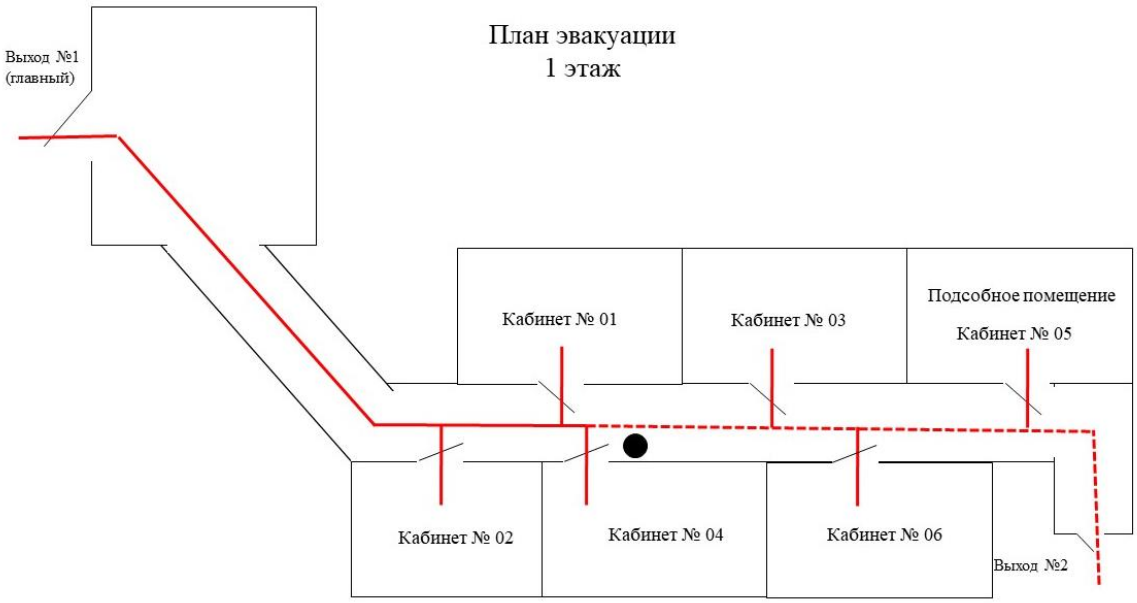


Рисунок 16 – Пути эвакуации

Для предотвращения пожара помещение с ПК должно быть оборудовано первичными средствами пожаротушения: углекислотным огнетушителем типа ОУ-2 или ОУ-5.

Пожар может нанести не только вред здоровью, но и материальный ущерб. Применимо к выполняемой работе в случае пожара могут быть уничтожены бумажные документы и\или электронные носители информации. Для защиты информации рекомендуется использовать облачные хранилища данных для данных и документов. Для исходных кодов программ рекомендуется использовать системы контроля версий [45].

6.5 Вывод по разделу

Проанализировав условия труда на рабочем месте, где велась работа над магистерской диссертацией, можно сделать вывод, что помещение удовлетворяет необходимым нормам и в случае соблюдения техники безопасности и правил пользования компьютером работа в данном помещении не приведет к ухудшению здоровья работника.

Само помещение и рабочее место в нем удовлетворяет всем нормативным требованиям. Во избежание негативного влияния на здоровье во время работы с ПК необходимо делать перерывы и проводить специализированные комплексы физических упражнений.

Действие вредных и опасных факторов сведено к минимуму соответствующими профилактическими работами, т.е. микроклимат, освещение и электро- и пожаробезопасность соответствуют требованиям, предъявленным в соответствующих нормативных документах.

Относительно рассмотренного вопроса об экологической безопасности можно сказать, что несмотря на небольшое число вредных и опасных отходов, стоит уделить внимание на их правильную утилизации без вреда для окружающей среды.

Заключение

В результате работы был спроектирован фреймворк для клиентской части веб-приложения. Фреймворк предоставляет функционал для компонентной реализации веб-приложения с применением парадигмы объектно-ориентированного программирования и представлении пользовательского интерфейса в виде JavaScript объекта. Также в функционал фреймворка обеспечивает основные необходимые возможности такие как синхронизация состояния приложения, подписки на события одного компонента другим, подконтрольное обновление пользовательского интерфейса, настроенная стратегия кеширования и разбиения проекта на точки входа для оптимизации загрузки. Компоненты могут быть переиспользованы по необходимости. Преимуществами фреймворка является его безопасность и надежность, отсутствие сторонних зависимостей, а также кастомизируемость индивидуальным разработчиком под свои нужды вследствие его относительно небольшого размера.

Для анализа производительности фреймворка было создано демонстрационное приложение, в ходе тестирования производительности которого анализировались такие метрики как время срабатывания браузерных событий в том числе загрузка, скриптинг, рендеринг, отрисовка и системные процессы, а также время, за которое пользователь увидит контент отличный от пустой страницы. Все вышеперечисленные параметры оказались в зеленой зоне инструмента Google lighthouse, что означает хорошую оптимизацию и безошибочную работу приложения.

Разработанный фреймворк позволяет абстрагироваться от задач обновления пользовательского интерфейса и особенностей клиентской разработки, что сделает клиентскую разработку доступнее для разработчиков со знаниями ООП языков, позволит оптимизировать бюджет компаний, не привлекая специалистов со специфичными знаниями.

Список публикаций студента

1. Ткачев М., Трофимова А.Е., Ротарь В.Г. ОБЗОР АКТУАЛЬНЫХ ФРЕЙМВОРКОВ ДЛЯ РАЗРАБОТКИ КЛИЕНТСКОЙ ЧАСТИ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ // Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 22–26 марта 2021 г. - Томск: ТПУ, 2021 - С. 458.
2. Трофимова А.Е., Ткачев М., Ротарь В.Г. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ СКЛАДСКИМИ ПРОЦЕССАМИ ИНТЕРНЕТ-МАГАЗИНОВ // Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 22–26 марта 2021 г. - Томск: ТПУ, 2021 - С. 458.

Литература

3. Першина Е. Л., Геращенко А. А. Сравнительный анализ фронтенд фреймворков //Образование. Транспорт. Инновации. Строительство. – 2020. – С. 717-720.

4. Пьюривал С. Основы разработки веб-приложений //СПб.: Питер. – 2015. – 272 с.

5. Бурханов К. С., Родионов В. В. Три кита реактивных фреймворков //информатика и вычислительная техника (ИВТ-2020). – 2020. – С. 48-51.

6. Fickers A., Balbi G. (ed.). History of the International Telecommunication Union: Transnational techno-diplomacy from the telegraph to the Internet. – Walter de Gruyter GmbH & Co KG, 2020. – Т. 1.

7. Mohanty M. S., nanda D. R. B. E-commerce: evolution, present status and future prospects.

8. Li X. et al. Measuring ease of use of mobile applications in e-commerce retailing from the perspective of consumer online shopping behaviour patterns //Journal of Retailing and Consumer Services. – 2020. – Т. 55. – С. 102093.

9. Акулов А. А., Бариев Р. Р. Исследование популярных javascript фреймворков для разработки клиентской части веб-приложения //Цифровизация экономики: направления, методы, инструменты. – 2019. – С. 358-362.

10. Газзаев Т. Р., Алексеев М. Д. Актуальность библиотеки jquery в 2019 году //проблемы научной мысли. – 2019. – С. 32.

11. Dao C. The Nature And Evolution Of JavaScript. – 2020.

12. Mitropoulos D. et al. Time present and time past: analyzing the evolution of JavaScript code in the wild //2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). – IEEE, 2019. – С. 126-137.

13. Chatzimparmpas A. et al. Analyzing the Evolution of Javascript Applications //ENASE. – 2019. – С. 359-366.

14. Варданян В. Г. Методы оптимизации программ на языке JavaScript, основанные на статистике выполнения программы //Труды Института системного программирования РАН. – 2016. – Т. 28. – №. 1.

15. Акулов А. А., Бариев Р. Р. Исследование популярных javascript фреймворков для разработки клиентской части веб-приложения //Цифровизация экономики: направления, методы, инструменты. – 2019. – С. 358-362.
16. Елисеева Е. С., Хаханова А. Д., Учанева А. А. Применение JAVASCRIPT-фреймворков при разработке интерактивных образовательных веб-приложений //Современное образование: традиции и инновации. – 2020. – №. 2. – С. 240-243.
17. Кургасов В. В., Лапшова А. Г. Javascript фреймворки //Центральный научный вестник. – 2018. – Т. 3. – №. 15-16. – С. 40-41.
18. Шкарбан Ф. В., Халилова З. Э., Абдуллаев А. Н. JavaScript-фреймворки и библиотеки //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. – 2017. – №. 3. – С. 48-52.
19. Сукиасян В. М., Придиус Е. С. Современные принципы и подходы к frontend архитектуре веб-приложений //Наука, техника и образование. – 2019. – №. 10 (63).
20. Будаев Е. С., Димова А. В. Обзор и сравнительный анализ библиотек и фреймворков JavaScript //Вестник современных исследований. – 2019. – №. 6.2. – С. 33-37.
21. Сулыз А. В., Панфилов А. Н. Технология разработки одностраничного веб-приложения на платформе angular 8 //Молодой исследователь Дона. – 2020. – №. 2 (23).
22. Оглукян А. К. Overview of the angular framework: pros and cons //теория и практика модернизации научной деятельности в условиях. – 2020. – С. 33.
23. Жуков С. В. Обзор и сравнение популярных фреймворков для frontend-разработки //Передовые инновационные разработки. Перспективы и опыт использования, проблемы внедрения в производство. – 2019. – С. 148-150.
24. Потовиченко М. А., Шатилов Ю. Ю. Разработка клиентской части одностраничного web-приложения с использованием библиотеки react //научное обозрение• технические науки scientific review• technical sciences www. science-education. ru 2020 г. – 2020. – С. 39.

25. Бондаренко Ю. В. Почему стоит использовать vue. Js //наука в современном обществе: закономерности и тенденции развития. – 2018. – С. 22.

26. Zakas N. C. Understanding ECMAScript 6: the definitive guide for JavaScript developers. – No Starch Press, 2016.

27. Vu T. Building a food application with full stack JavaScript. – 2020.

28. Persson M. JavaScript DOM Manipulation Performance: Comparing Vanilla JavaScript and Leading JavaScript Front-end Frameworks. – 2020.

29. Jagadale D. et al. Progressive web application for salons using artificial intelligence //International Engineering Journal For Research & Development. – 2020. – Т. 5. – №. 5. – С. 8-8.

30. Da Silva Moreira R. Development of a Progressive Web Application for stray pets reporting. – 2020.

31. Magomadov V. S. Exploring the role of progressive web applications in modern web development //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1679. – №. 2. – С. 022043.

32. Pham D. C. Progressive web applications as a solution for businesses: Case study: Twitter. – 2020.

33. Sarcar V. Observer Pattern //Design Patterns in C#. – Apress, Berkeley, CA, 2020. – С. 269-286

34. Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021).

35. ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования.

36. ГОСТ Р 50923-96. Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения.

37. ГОСТ 12.0.003-2015. Опасные и вредные производственные факторы. Классификация.

38. СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений.

39. ГОСТ 12.1.003-2014 ССБТ. Шум. Общие требования безопасности.
40. СН 2.2.4/2.1.8.562–96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки.
41. СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95*.
42. ГОСТ 12.1.006-84 ССБТ. Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля (с Изменением N 1).
43. ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов (с Изменением N1).
44. ГОСТ Р 12.1.019-2009. Электробезопасность. Общие требования и номенклатура видов защиты.
45. ГОСТ 30775-2001 Ресурсосбережение. Обращение с отходами. Классификация, идентификация и кодирование отходов.
46. ГОСТ Р 22.3.03-94. Безопасность в ЧС. Защита населения. Основные
47. ГОСТ 12.2.037-78. Техника пожарная. Требования безопасности

Приложение А. Раздел на иностранном языке

Overview of popular frontend frameworks

Студент

Группа	ФИО	Подпись	Дата
8ИМ91	Ткачев Михаил		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

Консультант-лингвист отделения иностранных языков ШБИП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИЯ	Сидоренко Татьяна Валерьевна	к.п.н.		

Introduction

In the field of programming, there is no clear definition of what criteria software must meet in order to be credibly considered a framework. Sometimes frameworks are confused with libraries.

A library usually consists of pre-written code, classes, procedures, scripts, configurations, and more. Basically, the library can be easily integrated into an existing project to reduce development time. This is due to the fact that many questions regarding the basic algorithms and functions have already been solved by other experienced programmers. The library saves time for developers by allowing them to focus on logic-related problems. But using a third-party library can also pose a potential risk to the application.

Compared to libraries, frameworks offer a full stack of useful functions and take responsibility for architectural decisions during the development process. Frameworks can include strategies for routing URLs in an application, managing state, bundling, and others. In addition, frameworks provide workflow enhancements that include best practices for core development aspects such as general application structure or templating.

Starting a new project, a programmer is faced with a choice: which JavaScript framework to choose for the site - Vue.js, React or Angular? There are differences between them, and quite significant. However, each of them is suitable for solving problems. Therefore, the question of work efficiency remains open. According to Stack Overflow Trends, React is at the top, overtaking Angular. When comparing Angular vs React to the popularity of Vue in 2020, Stack Overflow Trends shows that React gets the highest percentage, followed by Angular. But Vue's popularity has grown steadily over the past few years.

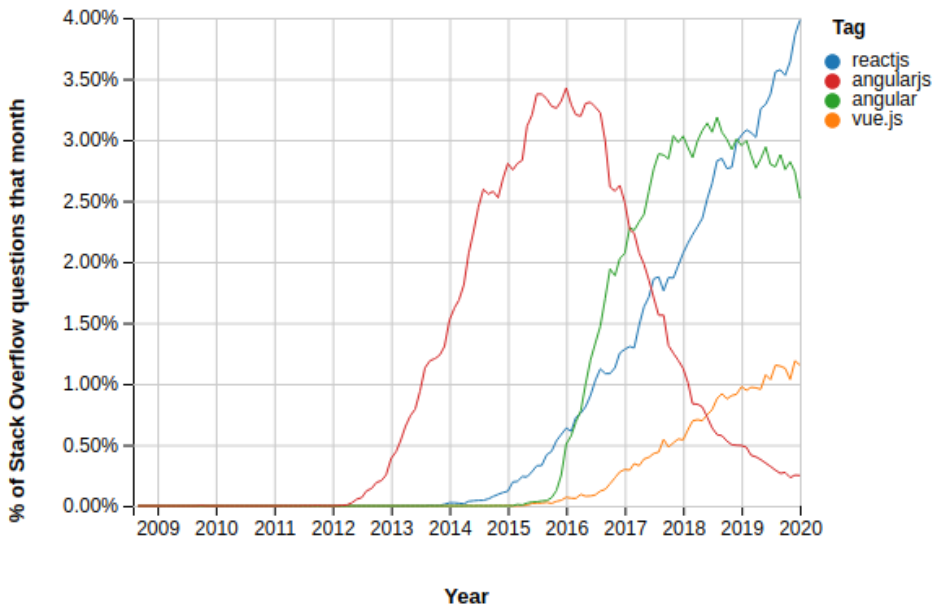


Figure A.1 Stack Overflow Trends

Developed by Facebook, React is a popular framework used to create and manage dynamic user interface of web pages. On the other hand, Angular is the most powerful, efficient and open source Javascript framework from Google that is used to create SPA (Single Page Application). However, Evan You created Vue with the support of a huge number of developers contributing to the development of the framework's ecosystem. It's not wrong to say that Vue has all the ethical aspects of React and Angular. And for this reason, Vue has gained wide popularity in a very short time, and throughout the entire time it has not lost its position.

The speed of customization depends solely on the number of libraries available to the developer. Therefore, it is also concluded that developing a web application is faster and easier in Angular than in React, due to the large number of customized tools out of the box. While the architecture of React is easier to scale than Angular.

Overview of the React framework

Launched in 2013, React had a simple goal of separating the user interface into a set of components to simplify the visual experience. React is used to build single page applications by implementing an extraordinary new feature, the virtual DOM.

Facebook first launched React in 2010 as an open source form syntax. However, when Facebook ads became difficult to manage with simple HTML, they started looking for a suitable solution, and Jordan Walke developed its prototype framework and came up with React in 2012.

React was not only open source, but it was warmly received by an estimated 3 million front-end developers. After that, coding problems were gradually resolved, and in 2014 the React company built a solid reputation. Facebook and Instagram are major companies using this framework.

One of the key features of React is its versatility. This library can be used on server and mobile platforms using React Native. This is the Learn Once, Write Anywhere or Learn Once, Write Anywhere principle.

Another important feature of the library is declarativeness. Using React, the developer describes how the components of the interface look in different states. A declarative approach shortens the code and makes it understandable.

React is component based, which is another key feature of the library. Each component returns a part of the user interface with its own state. By combining the components, the programmer creates a complete web application interface.

An important feature of React is its use of JSX. This is a JavaScript syntax extension that is useful for describing an interface. JSX is like HTML, but it's still JavaScript. A JSX example can be seen in the figure 2.

```
const header = text ? <h1>{text}</h1> : null;

const vdom = (
  <div>
    {header}
    <Hello />
  </div>
);
```

Figure A.2 JSX example

JSX allows to write JavaScript code using ready-made components that almost completely duplicate HTML. This makes development easier.

An important feature of React is the use of the Virtual DOM. The virtual DOM is an object that stores information about the state of an interface. When the state changes, such as after submitting a form or clicking a button, React calculates the difference between the old and new state. The library then renders the new state. Using the virtual DOM allows the library to efficiently update the real DOM.

React is based on one-way data management. This provides better control over the entire project.

But this technology has its own disadvantages. React is constantly changing, developers working with React need to regularly update their skills to keep up with the times. Also React is not a framework that gives a lot of functionality, architecture like Angular. Therefore, React is a suitable framework for creating modern web development, and designing most of the architecture for yourself.

While React doesn't come with the same tools as Angular or Vue, it does allow for flexibility trade-offs. With React, developer can combine and use any library. With the growth of the ecosystem, we have the opportunity to use tools such as CLI, Create React App and Next.js.

It was introduced by Facebook in 2013 as a JavaScript library that provides with rich framework functionality. The main companies that use React are Instagram, Netflix, New York Times, Yahoo, Whatsapp, Microsoft, Airbnb, Dropbox, and so on.

Angular Framework Overview

Angular was originally created by Google employees Misko Hevery and Adam Abrons in 2008. Then it was called AngularJS and was developed with plain JavaScript. This was at a time when most sites were based on a multi-page application approach: when a user clicked a link, the browser had to receive a request for an HTML document from the server. This could take a long time, depending on Internet connection and server performance.

Gradually, the overall performance of user devices increased, so it became possible to execute application logic in the browser. This led to the approach of building single page applications or SPA.

AngularJS was one of the first frameworks for SPA development. It also provided the ability to replace low-performance jQuery, offering developers features such as two-way data binding and the ability to organize modules for importing external scripts.

In the summer of 2014, Angular 2 was announced. Angular 2 meant that the framework was completely rewritten. Along with this rewrite, many of the core concepts of the framework have changed. While AnjularJS has focused on the MVC pattern, Angular 2 relies entirely on component hierarchy. Angular 2 lacks controllers or view models, instead there are components that consist of a template (such as a view), classes and metadata (decorators).

An important innovation in the framework is TypeScript. TypeScript is an extension of the JavaScript language and is the main development language using the Angular2 framework. It adds strong typing to the project, as well as various new functionality such as decorators, enums, etc. Subsequently, TypeSrypt is compiled to JavaScript to run in a browser.

Angular2, like many other frameworks, is component-based. This means that the components are the basic building blocks. They can display information, display templates, and perform actions on data. Best practice assumes that components are made up of three separate files: an HTML file for the template, a CSS file for styling, and a TS file for control. Following this approach, the principle of separation of concerns is implemented, and the code becomes more organized and structured.

Components are organized hierarchically, which means that information can flow between parent and child components, between two or more child components, and between two or more completely unrelated components. One of the special components is the application root directory. It represents the top-level node of the component tree and is the entry point at which the framework initializes the application.

Also, worth mentioning is data binding within a component. Essentially, this is about the exchange of data between the view (i.e. the HTML template) and the model (i.e. the TypeScript file). Again, there are three different types:

- Data binding: data is passed from component to template.
- Event binding: Data flows from the template to the component.
- Double-sided binding: a combination of both of the above types.

In its work, Angular tracks changes by adding internal observers to the components. Moreover, it implements an extensive system of lifecycle hooks. This system runs continuously and sits in the background.

There is also a command line interface. The command line interface is the "core" of the framework. It serves as the entry point for almost every Angular application. It can be installed using NPM.

Angular2 implements its own routing system for SPA usability. By defining routes in a separate file or module, Angular handles the logic of which component should be displayed based on the currently active URL path.

Since its launch in 2010, Angular has been continuously supported by Google, with frequent updates every six months. Moreover, according to the Medium report, Angular is a framework used by well-known companies such as Microsoft, Autodesk, Apple, Adobe, Freelancer, Upwork, Telegram, and so on.

Vue framework overview

Vue was published in 2014 by Evan Yu, a former Google employee who worked with AngularJS. Vue is often described as a progressive framework that can be used to create user interfaces for web applications. Although not strictly related to the Model-ViewViewmodel (MVVM) pattern, Vue's design principles were inspired in part by this pattern.

According to the official site, the framework can be used for small projects where the main library is used among other technologies, and for full-fledged single page web applications.

Scalability is one of the main advantages of this framework among other technologies. One of the features of Vue is that it is completely community developed and is an open source application.

It should be noted that Vue components are very similar to Custom Elements, which are part of the web components specification. This relatively new approach to web development allows developers to create new HTML tags or modify existing ones to complement them with functionality only using vanilla JavaScript, HTML and CSS.

Vue differs from other frameworks in some key aspects and offers its advantages. Vue works reliably in all supported browsers. In addition, Vue components add functionality that native elements cannot provide, such as dispatching custom events and cross-component data flow.

When it comes to structuring components, the Vue manual suggests using a single root file and the rule that each component must have a single root element. Example of component is shown in the figure 3.

```
<template>
    <div class="homeworkTasks">
        <p class="homeworkDescription">{{homework.description}}</p>
        <div class="homeworkDone" @click="done(homework)">
            x
        </div>
    </div>
</template>
```

Figure A.3 Vue component example

Internally, templates are compiled into Virtual DOM rendering functions. Combined with the reactivity system, Vue can effectively determine the minimum number of components to re-render and apply the required amount of DOM manipulation when the application state changes.

Similar to Angular, Vue uses directives in its templates. They can be used for data binding, event handling, and more. In the template, they are visually marked with the v prefix.

Vue data binding provides two options for data binding between model and view:

• One-way: v-bind

• Double sided: v-shaped

The latter keeps the view and model in sync regardless of where the change originally occurred. And also, props can be used to pass data between parent and child components.

Vue provides an official command line interface to improve development. A special feature of this command line interface is the fact that the development team provides a set of pre-configured templates for different use cases.

As mentioned earlier, the core library is only targeted at the presentation layer. Hence, additional libraries are needed to extend the functionality of the application, similar to React.

As a new member of the family, Vue has the smallest community compared to React and Angular. Vue is the smallest of the other frameworks and libraries as mentioned above and is ideal for lightweight web development and single page SPA applications. Compared to Angular and React, Vue's programming framework allows to quickly deploy application without sacrificing performance. With a simple command, developer can use exactly what he need for development. Developing an app is quick and easy with Vue, making it ideal for startups.

It may be a new framework and did not have such an active community as Angular and React, but still, it is used by leading companies including EuroNews, Alibaba, WizzAir, Xiaomi, Gitlab, Grammarly and many others.

Conclusion

Which framework is the best is a subjective question. All three offers similar functionality, and mastering any of them will certainly benefit any developer.

React and Vue are similar in many ways, they both use Virutal DOM, provide a component structure, and third-party issues are moved to additional libraries. Due to the fact that React and Vue are mostly focused on building user interfaces, while Angular is focused on applications, the latter is more difficult for beginners. Working with React and Vue becomes more difficult in large applications.

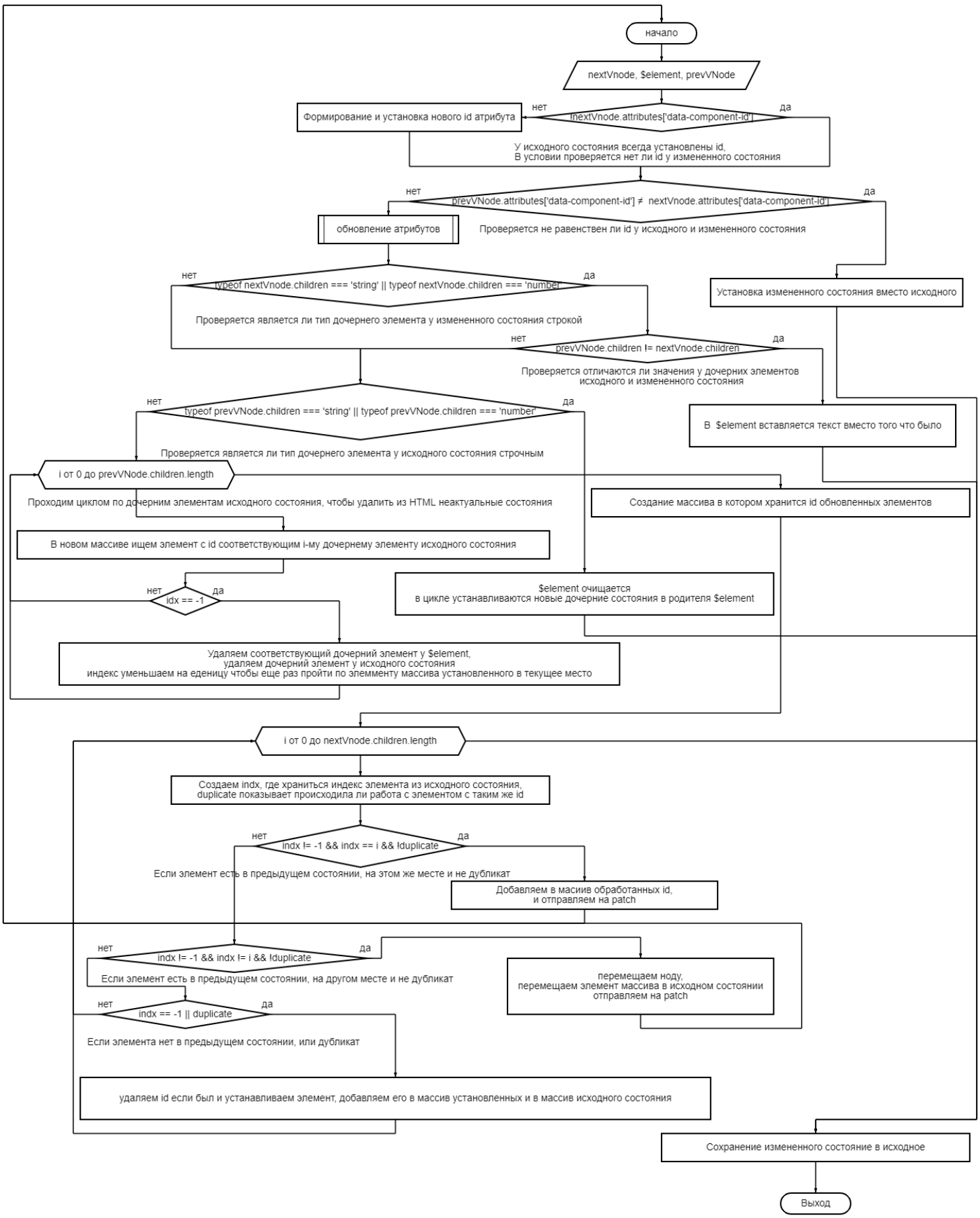
React beats Vue in the diversity of the ecosystem and the abundance of custom renderers available. In React, absolutely everything is JavaScript, while Vue is more based on classic web technologies, for example, any valid HTML will also be a valid Vue template.

Compared to Angular, both React and Vue are more flexible. But this is also the advantage of Angular: developer don't have to come up with his own implementation, just follow the rules.

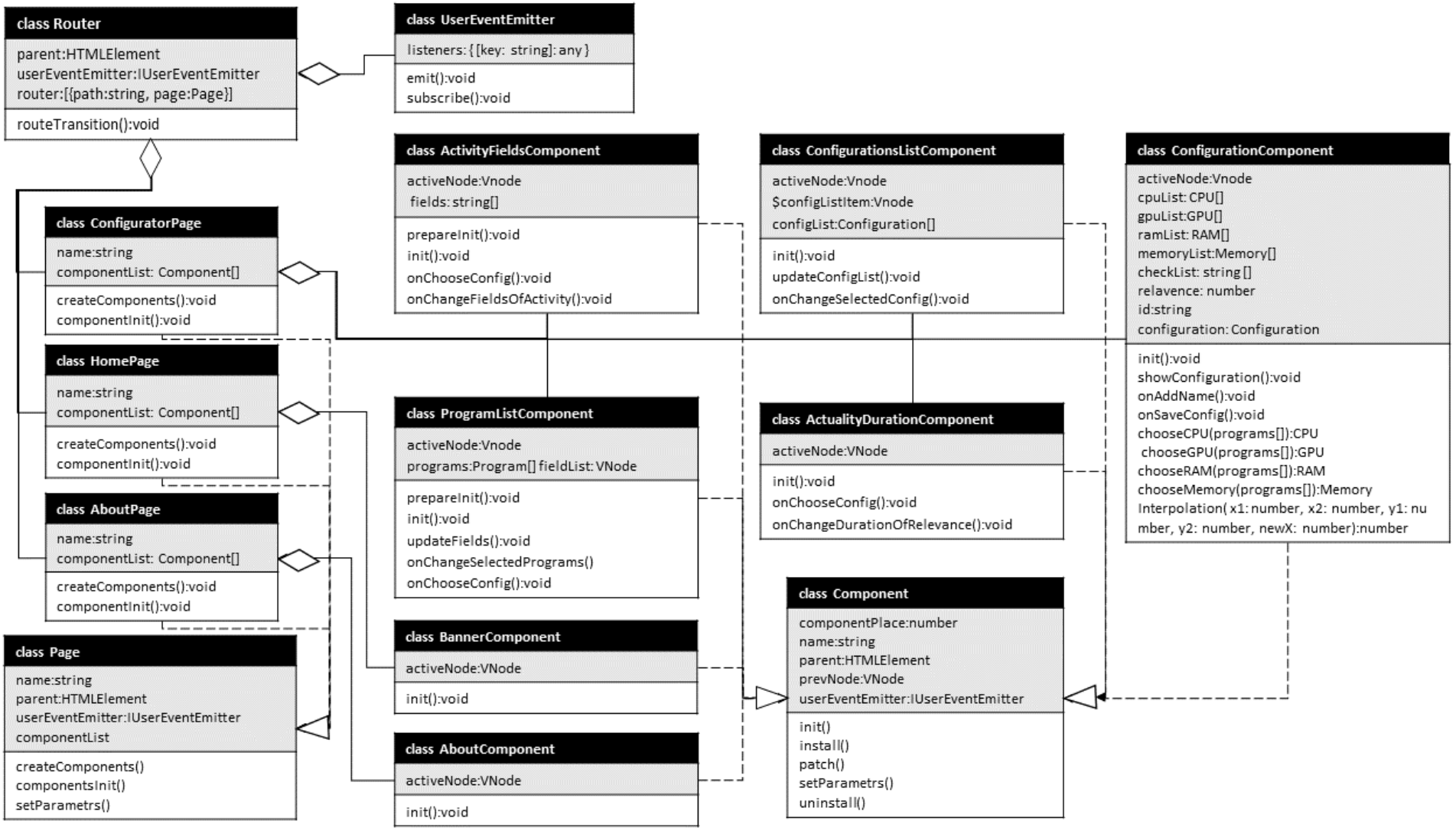
All three frameworks provide a CLI, making it easier to create new projects and support continuous development. Plus, they all get along well with popular IDEs like VS Code and Atom.

Performance can vary depending on the situation, but for the most part, all three frameworks are pretty fast - for good reason they are so popular. Performance is usually not the main criterion when choosing between the listed frameworks.

Приложение Б. Алгоритм обновления пользовательского интерфейса



Приложение В. Диаграмма классов демонстрационного приложения



Приложение Г. SWOT-анализ

	Сильные стороны научно-исследовательского проекта: С1. Заявленная экономичность и функциональность технологии. С2. Стоимость использования ниже аналогичных технологий. С3. Легкость в самостоятельной поддержке С4. Независимость от других компаний С5. Наличие вспомогательного веб приложения для более грамотного цикла разработки	**Слабые стороны научно-исследовательского проекта:** Сл1. Недостаток финансовой поддержки. Сл2. Небольшое сообщество поддержки Сл3. Маленькая команда разработки Сл4. Узкая направленность разработки
В1. Использование инновационной инфраструктуры ТПУ В2. Замещение импортных продуктов отечественными В3. Возникновение спроса на инновационный продукт В4. Повышение стоимости использования конкурентных разработок	1. Провести исследования проекта 2. Продвинуть проект на рынок 3. Улучшение эффективности поддержки проекта 4. Разработка новых функций проекта	1. Поиск деловых контактов 2. Обеспечить финансовую среду в рамках программы импортозамещения 3. Привлечь больше кадров для роста масштабов использования
У1. Отсутствие спроса на новые продукт У2. Развитая технология конкурентных технологий разработки У3. Снижение финансовой поддержки У4. Проблемы с интернет-соединением У5. Прекращение поддержки зависимых проектов	1. Провести рекламную компанию 2. Найти инвесторов 3. Устранить проблемы с техническим обеспечением 4. Постепенный отказ от сторонних решений в пользу, разрабатываемых в рамках проекта	1. Анализ рынка спроса на стадии разработки проекта 2. Планирование работ 3. Использование надежных сторонних зависимостей, за которыми стоит команда разработки и регулярность обновления проекта