

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Кафедра «Проектирование, управление и разработка информационных систем»
Направление 09.03.01 – «Информатика и вычислительная техника»

К ЗАЩИТЕ ДОПУСКАЮ

Заведующий кафедрой

_____ А.Н. Петрова

«___» _____ 2022 г.


ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КОМПЛЕКСНЫЙ ПРОЕКТ

Разработка web-сервиса
«Конструктор хакатонов»


Н. КОНТР

 _____ Е.Б. Абарникова

РУКОВОДИТЕЛЬ

 _____ Абарникова Е.Б.

СТУДЕНТ группы 8ВТб-1

 _____ Шаповалов Е.Э.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Кафедра «Проектирование, управление и разработка информационных систем»

УТВЕРЖДАЮ

Зав. кафедрой

_____ А.Н. Петрова
« ____ » _____ 2022г.

ЗАДАНИЕ
на комплексный проект

Выдано студенту Шаповалову Евгению Эдуардовичу

Тема комплексного проекта _____

Разработка web-сервиса «Конструктор хакатонов»

утверждена распоряжением по ФКТ № 16 от 12.01.2022

Срок сдачи студентом законченного комплексного проекта 23.04.2022

Исходные данные к работе Научная и учебная литература. Техническое задание от кандидата тех. наук, доцента Е.Б. Абарниковой

Перечень подлежащих разработке вопросов в расчетно-пояснительной записке:

1 Спецчасть _____

а) изучение предметной области – организация хакатонов;

б) сравнительный анализ аналогичных web-сервисов создания хакатонов;

в) создание объектной модели хакатонов;

г) создание функциональной карты сервиса;

д) разработка требований к web-сервису;

е) проектирование web-сервиса в нотации BPMN 2.0;

ё) разработка дизайна и кликабельного прототипа web-сервиса;

ж) разработка web-сервиса;

з) тестирование и отладка web-сервиса.

2 Экономическая часть не предусмотрена.

3 Экологичность и безопасность не предусмотрены.

4

Перечень графического материала (с точным указанием обязательных чертежей)
не предусмотрен.

Консультанты по ВКР (с указанием относящихся к ним разделов работы)
Е.Б. Абарникова.

Задание принял к исполнению  « » 2022 г.
(подпись)

Руководитель,


(подпись)

Е.Б. Абарникова
(Ф.И.О.)

должность, ученая степень доцент, канд. техн. наук «13» 01 2022 г.

Аннотация

Web-сервис «Конструктор хакатонов»

Пояснительная записка 54 с., 19 рис., 4 табл., 9 источников, приложения отсутствуют

Целью работы, выполняемой в комплексном проекте является разработка гибкого и адаптивного под потребности пользователя web-сервиса. Данный программный продукт упростит процесс организации хакатонов, предоставит доступ к единой платформе проведения хакатонов как для участников, так и для организаторов.

Данная работа состоит из введения, где указана актуальность разрабатываемого сервиса, основная часть, где указаны этапы выявления объектных моделей предметной области, процесс проектирования сервиса, создание функциональных карт, анализ аналогов. Также основная часть содержит в себе информацию по разработке дизайна и кликабельного прототипа, описание создания MVP версии продукта и процесса тестирования. Заключительная часть содержит в себе выводы о проделанной работе и соответствия программного продукта требованиям заказчика

Разработка программного обеспечения производится с использованием следующего набора технологий: MongoDB, Express.js, Vue.js, Node.js. Данный набор технологий позволит поддерживать изменчивое состояние документоориентированных баз данных и динамическое изменение веб-страниц.

Проектирование сервиса осуществляется в нотации BPMN 2.0 с точки зрения протекания бизнес-процессов. Рассматривается набор под процессов основных процессов: организации мероприятия и участия в мероприятии.

Abstract

Web-application for a mining enterprise

An explanatory note 54 C., 19 fig., 4 table., 9 sources, no appendices

The purpose of the work performed in a complex project is to develop a flexible and adaptive web service for the needs of the user. This software product will simplify the process of organizing hackathons, provide access to a single platform for conducting hackathons for both participants and organizers.

This work consists of an introduction, which indicates the relevance of the service being developed, the main part, which indicates the stages of identifying object

models of the subject area, the process of designing the service, creating functional maps, analyzing analogues. Also, the main part contains information on the development of a design and a clickable prototype, a description of the creation of the MVP version of the product and the testing process. The final part contains conclusions about the work done and the compliance of the program product with the customer's requirements

Software development is performed using the following set of technologies: MongoDB, Express.js, Vue.js, Node.js. This set of technologies will support the changeable state of document-oriented databases and the dynamic change of web pages.

The design of the service is carried out in BPMN 2.0 notation from the point of view of the flow of business processes. A set of sub-processes of the main processes is considered: organization of the event and participation in the event.

Оглавление

Введение.....	8
1 Техническое задание.....	10
1.1 Общие сведения	10
1.1.1 Наименование программы	10
1.1.2 Краткая характеристика области применения программы	10
1.2 Основания для разработки	10
1.2.1 Перечень документов, на основании которых.....	10
1.2.2 Наименование и условное обозначение темы разработки	10
1.2.3 Сведения о разработчике и заказчике.....	10
1.2.4 Плановые сроки начала и окончания работы	11
1.2.5 Сведения об источниках и порядке финансирования работ..	11
1.2.6 Порядок оформления и предъявления заказчику	11
1.3 Назначение разработки.....	12
1.3.1 Назначение программы	12
1.3.2 Цели создания программы	12
1.4 Требования к программе или программному изделию.....	12
1.4.1 Требования к функциональным характеристикам	12
1.4.2 Требования к надежности	13
1.4.3 Условия эксплуатации.....	13
1.4.4 Требования к составу и параметрам технических средств	14
1.4.5 Требования к информационной и программной совместимости.....	15
1.4.6 Требования к транспортированию и хранению.....	16
1.5 Требования к программной документации	16
1.6 Техничко-экономические показатели	17

1.7 Стадии и этапы разработки.....	19
1.8 Порядок контроля и приемки	19
1.9 Источники разработки.....	20
2 Описание программы.....	21
2.1 Общие сведения	21
2.1.1 Обозначение и наименование программы	21
2.1.2 Программное обеспечение, необходимое для функционирования программного комплекса	21
2.1.3 Языки программирования и программное обеспечение, с использованием которых велась разработка.....	21
2.2 Функциональное назначение	21
2.3 Описание логической структуры	25
2.4 Используемые технические средства	26
2.5 Вызов и загрузка	27
2.6 Входные данные.....	27
2.7 Выходные данные	27
3 Текст программы.....	29
3.1 Текст кода проекта «server».....	29
3.1.1 Текст файла проекта controllers/api.js	29
3.1.2 Текст файла проекта models/posts.js	30
3.1.3 Текст файла проекта routes.js.....	30
3.1.4 Текст файла проекта .env	31
3.1.5 Текст кода app.js.....	31
3.1.6 Текст кода package.json	31
3.2 Текст проекта «client».....	32
3.2.1 Текст файла проекта .gitignore.....	32
3.2.2 Текст файла проекта babel.config.js	32

3.2.3	Текст файла проекта package.json	32
3.2.4	Текст кода README.md	33
3.2.5	Текст кода vue.config.js	33
3.2.6	Текст кода src/plugins/vuetify.js	34
3.2.7	Текст кода src/router/index.....	34
3.2.8	Текст кода src/views/About.vue.....	35
3.2.9	Текст кода src/views/AddPost.vue	35
3.2.10	Текст кода views/EditPost.vue	36
3.2.11	Текст кода views/Home.vue	38
3.2.12	Текст кода views/Post.vue.....	39
3.2.13	Текст кода api.js	40
3.2.14	Текст кода main.js	40
3.2.15	Текст кода App.vue	41
4	Руководство программиста	42
4.1	Назначение и условия применения программы.....	42
4.2	Характеристика программы.....	42
4.2.1	Описание интерфейса.....	42
4.2.2	Подсистема работы с базой данных	44
4.3	Обращение к программе.....	47
4.4	Выходные и входные данные.....	48
4.5	Сообщения	48
5	Руководство оператора	49
5.1	Назначение программы	49
5.2	Условия выполнения программы	49
5.3	Выполнение программы.....	49
5.3.1	Предварительные настройки	49

5.3.2 Работа на главной странице сайта	49
5.4 Сообщения оператору	52
Заключение	54
Список использованных источников	55

Введение

Актуальность данной работы обусловлена потребностью рынка в сервисе, который предоставит возможность организации различного рода мероприятий, учитывающий разнообразные сферы деятельности и потребности пользователей.

При выполнении интернет-запроса формируется немалый список сервисов, предлагающих создать или организовать свой онлайн хакатон, однако, в основном предоставляются услуги организации хакатонов (отдельный сайт или офлайн-организация мероприятия), площадки для конкретного свободного создания отсутствуют.

Разрабатываемый сервис решает данную проблему, путем предоставления функционала для создания собственного хакатона в удобном, быстром и адаптированном под потребности пользователя формате. Таким образом, например, преподавателю или менеджеру отдела не составляет труда организовать среди своих учеников или работников хакатон на любую тематику.

Объектом данной работы является web-сервис в сети Internet.

Предметом данной работы является система организации страницы мероприятия, где пользователь помимо заполнения базовых полей может генерировать свои собственные под потребности той предметной области под которую создается мероприятие. Система авторизации пользователей. Возможность участие в ранее созданных мероприятиях.

Целью данной работы является спроектировать и разработать гибкий и адаптивный web-сервис «Конструктор хакатонов», предназначенный для упрощения процесса организации хакатонов, предоставления доступности как для участников, так и для организаторов к единой платформе проведения хакатонов.

Для достижения цели необходимо решить следующие **задачи**:

- изучить предметную область и аналоги;

- составить объектную модель хакатона и пользователей (в рамках предметной области «хакатон»).
- составить функциональную карту;
- проектирование сервиса в нотации BPMN 2.0;
- разработать дизайн сервиса и кликабельный прототип;
- реализация механизма адаптивно изменяющихся структур баз данных и web-страниц;
- разработка структуры базы данных;
- реализация клиентской и серверной части приложения;
- комплексные тестовые испытания разработанного сервиса;
- доработка сервиса по результатам тестирования;
- разработка дополнительного функционала (мобильное приложение для зрительского голосования);
- регистрация продукта в ФИПС;
- публикация всего сервиса в сети Интернет;

Пояснительная записка выпускной квалификационной работы состоит из 6 документов. Коды документов приведены в таблице 1.

Таблица 1

Обозначение	Наименование	Примечание
4217.02067988.18 - 1777 12	Текст программы	
4217.02067988.18 - 1777 13	Описание программы	
4217.02067988.18 - 1777 33	Руководство программиста	
4217.02067988.18 - 1777 34	Руководство пользователя	
4217.02067988.18 - 1777 51	Программа и методика испытаний	
4217.02067988.18 - 1777 90	Техническое задание	

1 Техническое задание

1.1 Общие сведения

1.1.1 Наименование программы

Наименование программного обеспечения: «web-сервис Конструктор хакатонов».

Краткое наименование – «Конструктор Хакатонов».

1.1.2 Краткая характеристика области применения программы

Программное обеспечение предназначено для упрощения и автоматизации процесса организации мероприятий, а также участия.

1.2 Основания для разработки

1.2.1 Перечень документов, на основании которых создаётся программа

Документы, на основании которых создается программное обеспечение:

- распоряжение на комплексный проект;
- техническое требование на разработку и внедрение;
- календарный план-график выполнения этапов работы;
- техническое задание, рассмотренное ниже.

1.2.2 Наименование и условное обозначение темы разработки

Наименование темы разработки: «web-сервис конструктор хакатонов».

Условное обозначение: «Конструктор хакатонов».

Шифр темы: комплексный проект.

Код работы: 4217.02067988.18 - 1777.

1.2.3 Сведения о разработчике и заказчике

Разработчик: студент группы 8ВТб-1 ФГБОУ ВО «КНАГУ» Шаповалов Евгений Эдуардович.

Заказчик – ИТ-управление ФГБОУ ВО «КНАГУ», в лице руководителя ИТ-управления, кандидата тех. наук, доцента Е.Б. Абарниковой.

1.2.4 Плановые сроки начала и окончания работы по созданию программы

Плановый срок начала работ: 05.02.2022 г.

Плановый срок окончания работ: 23.04.2022 г.

1.2.5 Сведения об источниках и порядке финансирования работ

Работа носит инициативный характер. Финансирование отсутствует.

1.2.6 Порядок оформления и предъявления заказчику результатов работ

Оформление и предъявление заказчику результатов работ по созданию программного обеспечения «Конструктор хакатонов» производится в соответствии с календарным планом-графиком выполнения работ.

Порядок взаимодействия разработчика с заказчиком:

- 1) проектирование сервиса;
- 2) предъявление спроектированного сервиса заказчику;
- 3) устранение выявленных недостатков и недоработок с точки зрения проектирования;
- 4) разработка дизайна и прототипа;
- 5) создание рабочей программы;
- 6) предъявление программного обеспечения заказчику;
- 7) выявление заказчиком недостатков и недоработок в рамках настоящего технического задания;
- 8) устранение выявленных недостатков и недоработок;
- 9) создание пакета технической документации на программное обеспечение, оформленного в соответствии с ГОСТ 19.101 и ЕСПД.

1.3 Назначение разработки

1.3.1 Назначение программы

В настоящий момент рынок не имеет сервисов по организации хакатонов с обширным функционалом охватывающий различные сферы деятельности. Как правило такие сервисы заточены либо под какую-либо конкретную тематику, либо предлагают услуги создания отдельных сайтов под проведение мероприятия, что является очень дорогим решением. Создание подобного сервиса позволит занять соответствующую нишу рынка, и предоставить дешевое решение проблемы организации мероприятий, с большим охватом разнообразных сфер деятельности.

1.3.2 Цели создания программы

Цели создания программного обеспечения:

- ускорить процесс организации мероприятий (хакатоны, идеатоны, чемпионаты);
- снижение стоимости организации мероприятий (хакатоны, идеатоны, чемпионаты);
- увеличение доступности и открытости для участия благодаря онлайн-формату.

1.4 Требования к программе или программному изделию

1.4.1 Требования к функциональным характеристикам

Заказчиком были выдвинуты следующие требования к программе:

- авторизация и регистрация;
- восстановление пароля;
- создание, удаление и редактирование мероприятий;
- просмотр истории прошедших мероприятий;
- просмотр текущего, активного мероприятия;
- участие в мероприятии;

- создание команд участниками мероприятий;
- просмотр данных участников в рамках одного мероприятия;
- возможность оценки результатов команд и участников;
- минималистичный интерфейс, с использованием стилистики Vuetify;
- web-сервис должен функционировать на любых устройствах, поддерживающие основные веб-браузеры.

1.4.2 Требования к надежности

Во время взаимодействия пользователя и web-сервиса при условии соблюдения всех требований, перечисленных выше в пункте 1.4 настоящего «Технического задания», общий процент отказов программы не должен превышать 2 %. Все возможные ошибки обрабатываются заранее прописанными программными методами. Типичные ошибки предусмотрены методами отлова ошибок языка javascript.

Оценку надежности программы необходимо производить в результате многочасового тестирования с использованием всех функциональных возможностей web-сервиса «Конструктор хакатонов».

1.4.3 Условия эксплуатации

Требования к условиям эксплуатации к техническому обслуживанию. Условия эксплуатации web-сервиса «Конструктор хакатонов» определяются условиями эксплуатации используемого аппаратного обеспечения. Web-сервис «Конструктор хакатонов» не имеет ограничения по времени эксплуатации.

Для нормальной работы web-сервиса необходимо произвести ряд действий:

- проверить исправность работы браузера;
- стабильность Интернет-соединения.

Требования по диагностированию программного обеспечения.

Диагностирование неисправностей web-сервиса должно производиться при соблюдении условий эксплуатации.

Требования развития и модернизации программного обеспечения

В дальнейшем должно быть реализовано PWA приложения для смартфонов для возможности зрительского голосования. Улучшение пользовательского интерфейса, в зависимостей от изменения метрик продукта (AARRR и HEART).

Доработка нереализованного функционала в пост-MVP версиях.

Требования к численности и квалификации персонала и режиму его работы.

Для взаимодействия с web-сервисом «Конструктор хакатонов» требуется один пользователь, имеющий навык работы на персональном компьютере в ОС Microsoft Windows, Mac OS, Linux, Chrome OS.

Помимо навыков работы в операционных системах, так же требуется опыт работы с веб-браузерами (Google Chrome, Mozilla Firefox, Vivaldi, Opera, Microsoft Edge, Safari, Tor Browser, Яндекс).

1.4.4 Требования к составу и параметрам технических средств

Требования к техническому обеспечению для разработчика. Для разработки web-сервиса «Конструктор хакатонов» необходимы следующие технические средства:

- процессор Intel Core i5 8400;
- оперативная память 16 Гб и больше;
- свободное дисковое пространство 1000 Гб и больше;
- видеоадаптер MSI GeForce GTX1050 Ti;
- клавиатура;
- манипулятор «мышь».

Требования к техническому обеспечению для пользователя. Для корректной работоспособности web-сервиса «Конструктор хакатонов» необходимы следующие технические средства:

- процессор Intel Core i3 4000M;
- оперативная память 4 Гб и больше;
- свободное дисковое пространство 4 Гб и больше;
- видеоадаптер SVGA и выше;
- клавиатура;
- манипулятор «мышь».

1.4.5 Требования к информационной и программной совместимости

Требования к информационному обеспечению.

Для разработки web-сервиса «Конструктор хакатонов» рекомендуется следующее информационное обеспечение:

- документация по Node.js;
- документация по Vue.js;
- документация по MongoDB;
- документация по Vuetify;
- теоритическая информация по прототипированию в Figma;
- сведения по проектированию в нотации BPMN 2.0.

Требования к лингвистическому обеспечению.

В качестве языка программирования используется JavaScript с применением синтаксиса ES6, в контексте фреймворка Vue.js и его особенностей. Языком разметки выступает HTML5, средством стилизации страниц CSS3, фреймворк Vuetify.

Требования по диагностированию программного обеспечения.

Диагностирование неисправностей программного комплекса должно производиться при соблюдении условий эксплуатации.

Требования безопасности. В процессе функционирования программное обеспечение «Конструктор хакатонов» не должно влиять на работу любых других программных средств и не приводить к сбоям в работе компьютера и операционной системы.

Требования по обеспечению безопасности технических средств и дополнительного программного обеспечения устанавливаются в соответствии с предъявляемыми к ним требованиями безопасности.

Требования к эргономике и технической эстетике.

Создаваемое программное обеспечение должно обеспечивать понятный и удобный интерфейс, в соответствии с современными тенденциями; использование цветовой гаммы и шрифтового оформления в соответствии с правилами композиции; Оконный интерфейс веб-приложения должен быть адаптирован под разные разрешения экранов и устройства.

Требования по стандартизации и унификации.

Программное обеспечение «Конструктор хакатонов» должно предоставлять пользователю привычный, общепринятый в среде браузера интерфейс. Программная документация, поставляемая с программным обеспечением (ПО), должна быть оформлена в соответствии со стандартом ЕСПД.

1.4.6 Требования к транспортированию и хранению

Для хранения web-сервиса «Конструктор хакатонов» должна иметься возможность применять любые носители емкостью не менее 500 Мбайт.

Требования к транспортированию и хранению определяются соответствующими характеристиками транспортирования и хранения для выбранного типа носителя данных и используемого компьютерного оборудования.

1.5 Требования к программной документации

Состав программной документации, предъявляемой заказчику по окончании работ:

- «Техническое задание», ГОСТ 19.201-78;
- «Описание программы», ГОСТ 19.402-78;
- «Текст программы», ГОСТ 19.401-78;
- «Руководство программиста», ГОСТ 19.504-79;

- «Руководство оператора», ГОСТ 19505-79;
- «Программа и методика испытаний». ГОСТ 19.301-79.

Пояснительная записка оформляется в соответствии с РД ФГБОУ ПО «КНАГУ» 013-2016 «Текстовые студенческие работы. Правила оформления» и ЕСПД.

1.6 Техничко-экономические показатели

Web-сервис «Конструктор хакатонов» разрабатывается по индивидуальному заказу. Программа повышает эффективность и скорость работы организатора какого-либо мероприятия, либо участника.

Разработка web-сервиса «Конструктор хакатонов» носит инициативный характер, поэтому затраты на основную и дополнительную заработную плату разработчикам не предусмотрены, как и отчисления на социальные нужды.

Экономический эффект при использовании web-сервиса «Конструктор хакатонов» будут рассчитаны по формуле:

$$\Delta\Pч = (\text{Эз} - \Delta\text{Зтек}) * (1 - \text{Нп}) \quad (1.1)$$

где Эз – экономия текущих затрат, полученная в результате применения ПО, руб.;

$\Delta\text{Зтек}$ – прирост текущих затрат, связанных с использованием ПО, руб.;

Нп – ставка налога на прибыль, в соответствии с действующим законодательством, %.

Так как в итоге будет уменьшены затраты на заработную плату разработчиков, которым не придется проводить работу над описанными модулями, то экономия текущих затрат будет рассчитывается как экономия заработной платы последующих разработчиков по формуле:

$$\text{Зо} = \text{Кпр} * \sum_{i=1}^n \text{Зчi} * \text{ti} \quad (1.2)$$

n $i=1$ где n – количество исполнителей, занятых разработкой конкретного ПО;

Кпр – коэффициент премий (1,5-2,0);

Зчi – часовая заработная плата i -го исполнителя (руб.);

t_i – трудоемкость работ, выполняемых i -м исполнителем (ч).

В текущих расчетах не будем учитывать коэффициент премий. Также будет взята средняя заработная плата программиста среднестатистической частной компании по производству ПО. Средняя заработная плата программиста составляет 65000 рублей. Согласно трудовому кодексу статье 91 «Понятие рабочего времени. Нормальная продолжительность рабочего времени» для сотрудников установлена 40-часовая рабочая неделя, как правило, это означает, что они трудятся 5 дней в неделю по 8 ч в день. В месяц выпадает от 21 до 23 трудовых дней, если на будние дни не приходятся праздники. Так что возьмем средний показатель дней в месяце 22. Таким образом, средняя часовая заработная плата программиста в компании будет составлять около 370 рублей. Предполагается, что над проектом бы работали как минимум один программист и один тестировщик и дизайнер в течение месяца. Тогда $n = 3$, $K_{пр} = 1$, $3\text{Ч}_i = 370$ (руб.), $t_i = 22 \cdot 8 = 176$ (ч). В итоге основная заработная плата:

$$З_0 = (1 \cdot 370 \cdot 176) \cdot 3 = 195\,360 \text{ (руб.)} \quad (1.3)$$

Так как в данных расчетах основная заработная плата эквивалентна экономии текущих затрат, полученная в результате применения ПО, то $Эз = 65\,120$ руб.

В связи с тем, что разработчиком потребуется время, чтобы ознакомиться с текущими Web-сервиса «Конструктор хакатонов», то прирост текущих затрат будет эквивалентен заработной платы работников в течении одного рабочего дня.

$$\Delta Z_{\text{тек}} = (370 \cdot 8) \cdot 3 = 8880 \text{ (руб.)} \quad (1.4)$$

Основная ставка по налогу на прибыль составляет 20 процентов (п. 1 ст. 284 НК РФ), что означает, что $Н_п = 0,2$.

Таким образом, экономический эффект при использовании Web-сервиса «Конструктор хакатонов» будет составлять:

$$\Delta Пч = (65120 - 8880) \cdot (1 - 0,2) = 44\,992 \text{ (руб.)} \quad (1.5)$$

1.7 Стадии и этапы разработки

Этапы выполнения работ по созданию web-сервиса «Конструктор Хака-тонов» приведены в таблице 1.1.

Таблица 1.1 – Стадии и этапы разработки

Этап	Дата начала	Дата
Исследование предметной области и аналогов	08.10.2021	03.12.2021
Разработка и утверждение технического задания	04.03.2022	07.03.2022
Разработка концепции сервиса и проектирование бизнес-процессов	10.03.2022	20.03.2022
Разработка дизайна и кликабельного прототипа	21.03.2022	28.03.2022
Программная реализация web-сервиса	01.04.2022	30.04.2022
Тестирование и отладка сервиса	01.05.2022	17.05.2022
Подготовка документации	18.05.2022	02.06.2022
Опытная эксплуатация	03.06.2022	15.06.2022

1.8 Порядок контроля и приемки

Контроль программного обеспечения и его приемка осуществляются в следующей последовательности:

- 1) тестирование программы осуществляется по следующим параметрам: сервис выдает желаемые логичные результаты (создает мероприятие, публикует мероприятия, предоставляет возможность участия, корректное отображение атрибутов, создаваемых организатором);
- 2) выполняется исправление недостатков и ошибок программы, выявленных в результате тестирования.
- 3) проводится повторное тестирование сервиса и исправление его ошибок.
- 4) после устранения всех выявленных ошибок web-сервис сдается в эксплуатацию.

1.9 Источники разработки

Разработку программного обеспечения проводить на основе анализа существующих аналогов. Разрабатываемый сервис должен учитывать достоинства и недостатки исследуемых аналогов. Анализ проводить среди сервисов в сети Internet по следующим критериям:

- высокая техническая реализация – анализ продуктов и компаний производящих данный продукт на технические и кадровые возможности, учет скорости реагирования на добавление тех или иных возможностей, программных модулей;
- свободное создание мероприятий – критерий который указывает на возможность пользователя самому создавать мероприятие, а не отправлять контактные данные для связи с разработчиками для выяснения условий мероприятия;
- возможность проведения как онлайн, так и офлайн – критерий, показывающий мобильность продукта в отношении формата проведения мероприятия;
- охват различных сфер деятельности – критерий указывающий на предоставлении пользователю возможности создания мероприятия по той тематике, которая нужна;
- адаптивно настраиваемые параметры – критерий показывающий, присутствует ли в продукте возможность выбирать и настраивать те параметры мероприятия, которые необходимы.

2 Описание программы

2.1 Общие сведения

2.1.1 Обозначение и наименование программы

Наименование web-сервиса: «Конструктор хакатонов».

Web-сервис предназначен для организации хакатонов, идеатонов, чемпионатов и прочих мероприятий, предоставит доступ к единой платформе проведения для участников, экспертов и организаторов.

2.1.2 Программное обеспечение, необходимое для функционирования программного комплекса

Для корректного стабильной работы и функционирования web-сервиса необходимо следующее программное обеспечение:

- любой из популярных браузеров: Google Chrome, Mozilla Firefox, Vivaldi, Opera, Microsoft Edge, Safari, Tor Browser, Яндекс;
- любая из популярных операционных систем: Windows, Linux, Mac OS.

2.1.3 Языки программирования и программное обеспечение, с использованием которых велась разработка

Web-сервис реализован с помощью языка программирования JavaScript. Фреймворк Vue.js для клиентской части. Для серверной части node.js с фреймворком express.js.

Среда программирования Visual Studio Code. Система контроля версий GitHub.

2.2 Функциональное назначение

Web-сервис «Конструктор хакатонов» представляет из себя сайт, который упростит процесс организации различного рода мероприятий и предоставит доступ к единой платформе для их проведения, как для участников, так и для ор-

ганизаторов.

Аналоги сервисов для организации мероприятий приведены ниже.

Codenrock – сервис сильно пропагандирует проведение онлайн хакатонов, предоставляют поддержку на протяжении всего мероприятия (функциональные возможности представлены ниже). При попытке создания Хакатона просит оставить контактные данные (рисунок 2.1) для связи и оглашение цены, общего видения мероприятия. Не является универсальным сервисом. В основном Codenrock – это компания организатор, с хорошей технической поддержкой.

Функциональные возможности:

- формирование команд - сбор и общение команды внутри платформы;
- брендинг страницы - индивидуальное представление команды;
- оценка и рейтинг - различные форматы голосования и наблюдение за рейтингом;
- различные типы задач;
- курсы для обучения проведения хакатонов (youtube плейлист).

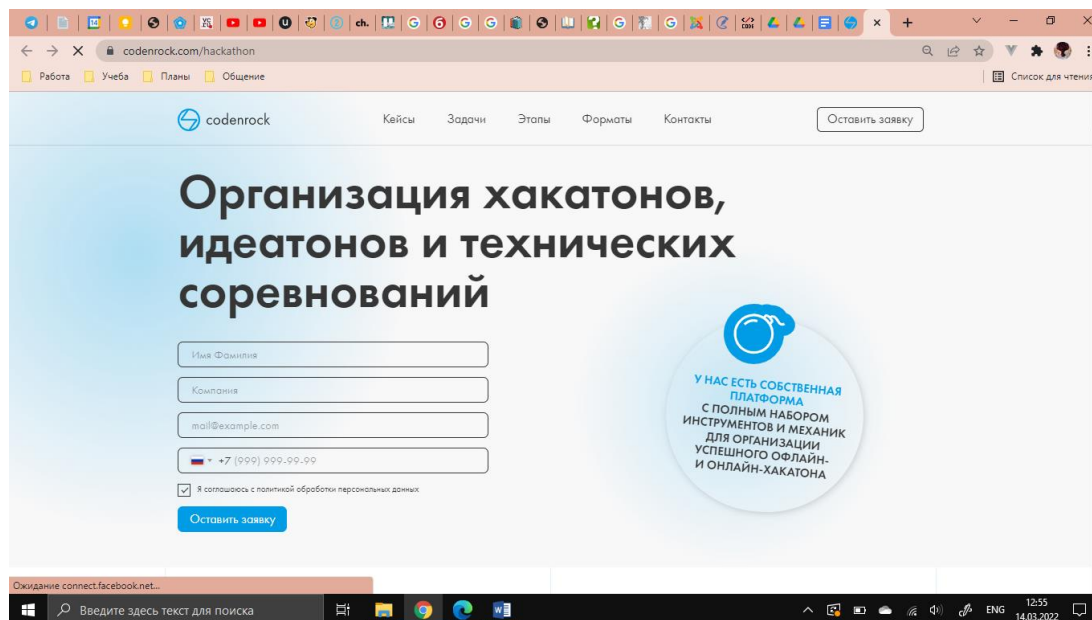
The image is a screenshot of a web browser displaying the 'codenrock.com/hackathon' page. The browser's address bar shows the URL. The website has a light blue header with the 'codenrock' logo and navigation links: 'Кейсы', 'Задачи', 'Этапы', 'Форматы', 'Контакты', and a button 'Оставить заявку'. The main content area has a large heading 'Организация хакатонов, идеатонов и технических соревнований'. Below this is a contact form with fields for 'Имя Фамилия', 'Компания', 'mail@example.com', and a phone number '+7 (999) 999-99-99'. There is a checkbox for 'Я соглашаюсь с политикой обработки персональных данных' and a blue 'Оставить заявку' button. To the right of the form is a circular graphic with text: 'У НАС ЕСТЬ СОБСТВЕННАЯ ПЛАТФОРМА С ПОЛНЫМ НАБОРОМ ИНСТРУМЕНТОВ И МЕХАНИК ДЛЯ ОРГАНИЗАЦИИ УСПЕШНОГО ОФЛАЙН- И ОНЛАЙН-ХАКАТОНА'. The browser's taskbar at the bottom shows various icons and the system clock indicating 12:55 on 14.03.2022.

Рисунок 2.1 – Форма обратной связи

Cups от mail.ru и YandexCup – примеры сервисов частных компаний, целью которых является проведение хакатонов для собственных целей и поиска новых сотрудников. Организовать свой на данной площадке невозможно. Явля-

ется просто хорошей площадкой для участия именно в ИТ-хакатонах (Рисунок 2.2).

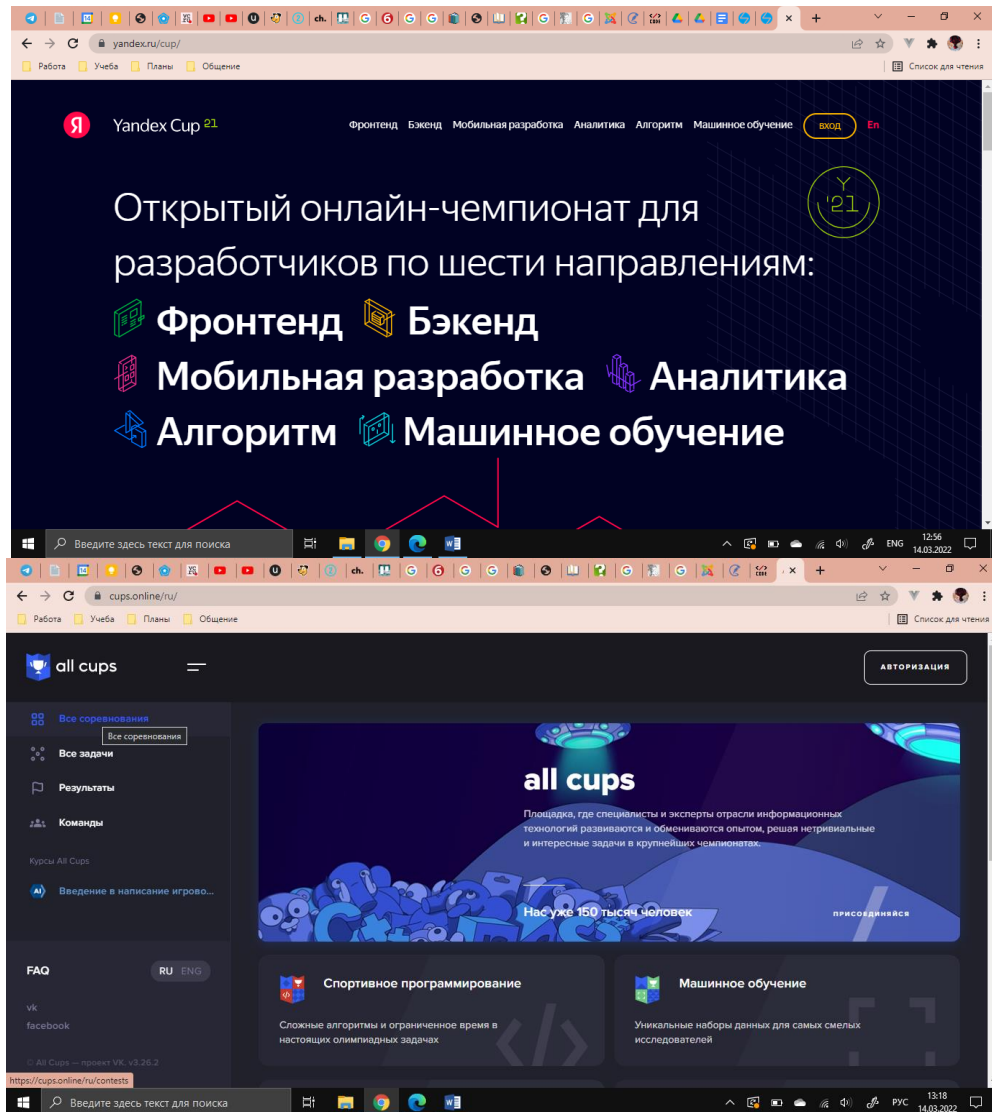


Рисунок 2.2 – Площадки Mail.ru и Yandex

Russianhackers - сервис предоставляет создания отдельных сервисов-хакатонов под ключ (отдельный сайт). Команда учитывает многие нюансы данных мероприятий. Основной уклон они делают на максимальное взаимодействие организаторов и участников. Они создают комфортную среду для того чтоб участники, не «разбегались» со своими проектами, а организаторы имели доступ к их персональным данным и контактной информации. Является не универсальным сервисом, а компанией организатором с хорошей технической поддержкой (рисунок 2.3).

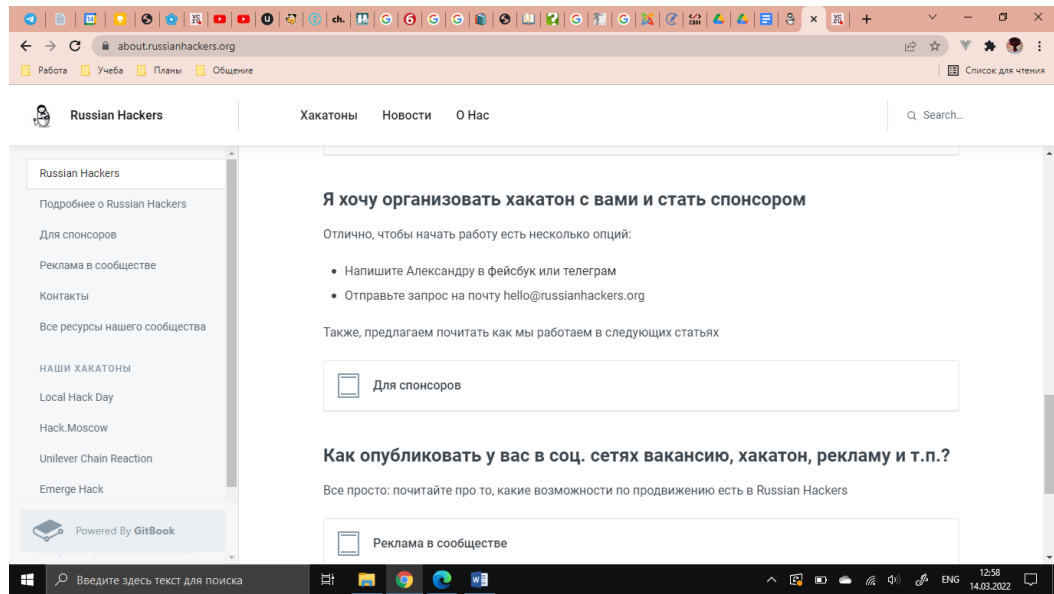


Рисунок 2.3 – Форма обратной связи Russianhackers

ZuckerStudio- аналогичен russianhackers, но нет сообщества (рисунок 2.4).

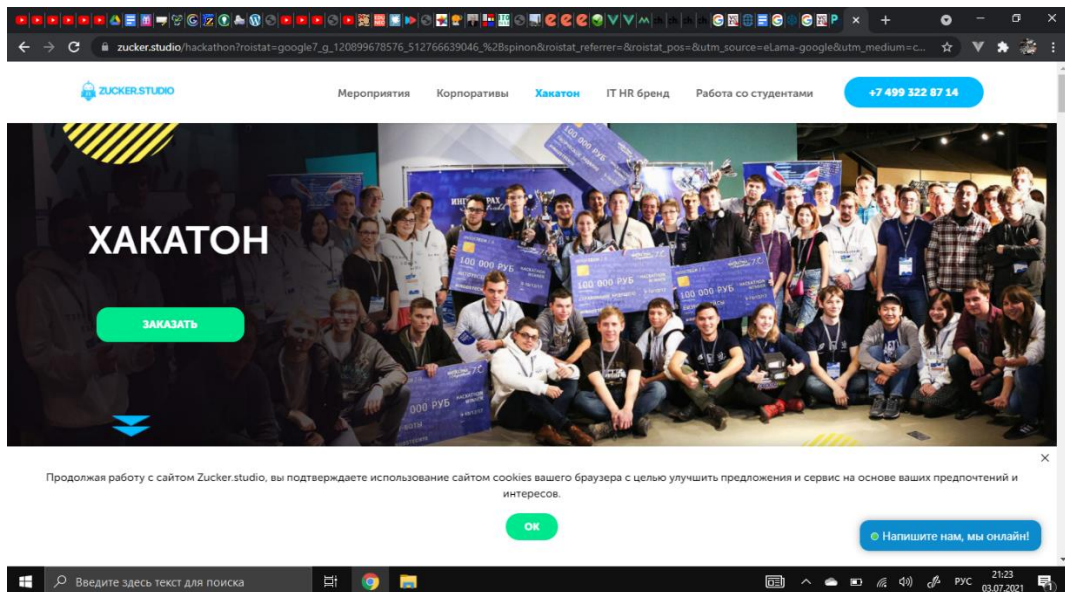


Рисунок 2.3 – Форма обратной связи ZuckerStudio

В результате проведенных исследований, были выявлены наиболее значимые критерии. Результаты анализа в сжатой форме представлены на таблице 2.1.

Таблица 2.1 – Результаты анализа существующих решений

	Yandex Cup	All Cups	Zucker.Studio	Codenrock	russianhackers
Высокая техническая реализация	+	+	+/-	+/-	+/-
Свободное создание мероприятий	-	-	-	-	-
Возможность проведения как онлайн, так и офлайн	-	-	+/-	+/-	+/-
Охват различных сфер деятельности	-	-	+	+	-
Адаптивно настраиваемые параметры	-	-	+/-	+/-	+

Данный критерий присутствует в полной мере «+», отсутствует «-», присутствует частично «+/-». При выполнении интернет-запроса формируется немалый список сервисов, предлагающих создать или организовать свой онлайн хакатон, однако, в основном предоставляются услуги организации хакатонов (отдельный сайт или офлайн-организация мероприятия), площадки для конкретного свободного создания отсутствуют.

2.3 Описание логической структуры

Главная страница сервиса встречает пользователя лэндингом с описанием функций сервиса и его преимуществ. На лэндинге пользователь может нажать на кнопку «Организатор» или «Участник». Кнопки перенаправляют пользователя на соответствующую роль только в том случае, если он предварительно зарегистрирован.

В зависимости от роли у пользователя изменяется функционал:

- **Окно организатора.** Первое, что встречает пользователя в режиме организатора это список его созданных мероприятий. Далее пользователь может создать мероприятие, после нажатия на кнопку «Создать мероприятие», попадая в меню конструктора. Если пользователь нажимает на одно из ранее созданных мероприятий, то он переходит в меню редактирования этого мероприятия. Из данного окна можно попасть в меню распределения ролей. Там организатор имеет возможность сформировать группу участников, группу экспертов, и наблюдателей (если предусмотрена возможность открытого просмотра результатов команд).

- **Окно участника.** Первое, что встречает пользователя в качестве участника это список всех опубликованных мероприятий на сервисе. Далее пользователь заходит на нужный ему хакатон (выбор из общего списка или переход по ссылке). В зависимости от того как настроил мероприятие организатор, пользователь, либо проходит предварительную регистрацию, либо сразу получает доступ к участию. В этом же окне участники между собой распределяются на команды, если организатор предоставил такую возможность.

- **Окно эксперта.** Эксперт имеет возможность просматривать этапы работы команд или участников. Так же эксперту доступно форма для оценки итоговых и промежуточных результатов.

Приложение имеет REST API архитектуру, где происходит обмен данными между клиентским и серверным приложением.

На основе описанной логической структуры можно приступить к разработке web-сервиса. Исходный код компонентов web-сервиса представлен в разделе 3.

2.4 Используемые технические средства

При разработки web-сервиса использовались следующие технические средства:

- процессор Intel Core i5 8400;

- оперативная память в объеме 16 Гб;
- дискретная видеокарта MSI Geforce GTX 1050 ti;
- SSD накопитель размером 512 Гб;
- манипулятор «мышь».

При разработки программного обеспечения использовались следующие программные средства:

- редактор кода Microsoft Visual Studio Code;
- браузер Google Chrome для отображения результатов разработки;
- графический редактор для дизайна и прототипирования Figma;
- графический редактор базы данных MongoDB Compass.

2.5 Вызов и загрузка

Для функционирования web-сервиса на устройстве должна быть установлена любая из известных операционных систем (Microsoft Windows, Mac OS, Linux, Chrome OS). Для запуска web-сервиса требуется в браузере (Google Chrome, Mozilla Firefox, Vivaldi, Opera, Microsoft Edge, Safari, Tor Browser, Яндекс) ввести следующий URL запрос: <https://hakaton.knastu.ru/>. Данный запрос откроет главную страницу сайта.

2.6 Входные данные

Входными данными web-сервиса являются, данные при регистрации пользователя и данные, вносимые при создании мероприятия (текст, прикрепляемые файлы). Данные вводятся пользователем с помощью клавиатуры, и манипулятор «мышь».

2.7 Выходные данные

Выходными данными web-сервиса являются созданные мероприятия различными организаторами (карточки мероприятий), которые предоставляют возможность ознакомления с мероприятием (просмотр названия мероприятия, описания, формулировок заданий) и скачивание прикрепленных файлов. Данные участников, такие как контактные данные, фамилия, имя, отчество, так же являются выходными.

3 Текст программы

3.1 Текст кода проекта «server»

3.1.1 Текст файла проекта controllers/api.js

```
const Post = require("../models/posts");
const fs = require("fs");

const { post } = require("../routes/routes");

module.exports = class API {
  static async fetchAllPost(req, res) {
    try {
      const posts = await Post.find();
      res.status(200).json(posts);
    } catch (err) {
      res.status(404).json({ message: err.message });
    }
  }

  static async fetchPostByID(req, res) {
    const id = req.params.id;
    try {
      const post = await Post.findById(id);
      res.status(200).json(post);
    } catch (err) {
      res.status(404).json({ message: err.message });
    }
  }

  static async createPost(req, res) {
    const post = req.body;
    const imagename = req.file.filename;
    post.image = imagename;
    try {
      await Post.create(post);
      res.status(201).json({ message: "Hack created successfully!" });
    } catch (err) {
      res.status(400).json({ message: err.message });
    }
  }

  static async updatePost(req, res) {
    const id = req.params.id;
    let new_image = "";
    if (req.file) {
      new_image = req.file.filename;
      try {
        fs.unlink("./uploads/" + req.body.old_image);
      } catch (err) {
        console.log(err);
      }
    } else {
      new_image = req.body.old_image;
    }
    const newPost = req.body;
    newPost.image = new_image;

    try {
```



```

    await Post.findByIdAndUpdate(id, newPost);
    res.status(200).json({ message: "Hack update successfully!" });
  } catch (err) {
    res.status(404).json({ message: err.message });
  }
}

static async deletePost(req, res) {
  const id = req.params.id;
  try {
    const result = await Post.findByIdAndDelete(id);
    if (result.image !== "") {
      try {
        fs.unlinkSync("./uploads/" + result.image);
      } catch (err) {
        console.log(err);
      }
    }
    res.status(200).json({ message: "Hack deleted successfully!" });
  } catch (err) {
    res.status(404).json({ message: err.message });
  }
}
};

```

3.1.2 Текст файла проекта models/posts.js

```

const mongoose = require("mongoose");
const postSchema = mongoose.Schema({
  title: String,
  category: String,
  content: String,
  image: String,
  created: {
    type: Date,
    default: Date.now,
  },
});
module.exports = mongoose.model("Post", postSchema);

```

3.1.3 Текст файла проекта routes.js

```

const express = require("express");
const router = express.Router();
const API = require("../controllers/api");
const multer = require("multer");

let storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "./uploads");
  },
  filename: function (req, file, cb) {
    cb(null, file.fieldname + "_" + Date.now() + "_" +
file.originalname);
  },
});

let upload = multer({
  storage: storage,
}).single("image");

```

```

router.get("/", API.fetchAllPost);
router.get("/:id", API.fetchPostByID);
router.post("/", upload, API.createPost);
router.patch("/:id", upload, API.updatePost);
router.delete("/:id", API.deletePost);

```

```
module.exports = router;
```

3.1.4 Текст файла проекта .env

```

PORT = 5000
DB_URI = mongodb://localhost:27017/mevn_full_stack

```

3.1.5 Текст кода app.js

```

//imports
require("dotenv").config();
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const app = express();
const port = process.env.PORT; // 5000;

// middlewares
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static("uploads"));

//database connection
mongoose
  .connect(process.env.DB_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useFindAndModify: true,
    useCreateIndex: true,
    family: 4,
  })
  .then(() => console.log("Соединение с MongoDB есть!"))
  .catch((err) => console.log(err));

//routes prefix
app.use("/api/post", require("../routes/routes"));

//Start even http://localhost:5000/
app.listen(port, () => console.log(`Сервер работает на хосте : ${port}`));

```

3.1.6 Текст кода package.json

```

{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  }
}

```

```

},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "cors": "^2.8.5",
  "dotenv": "^10.0.0",
  "express": "^4.17.2",
  "mongoose": "^5.13.14",
  "multer": "^1.4.4"
},
"devDependencies": {
  "nodemon": "^2.0.15"
}
}

```

3.2 Текст проекта «client»

3.2.1 Текст файла проекта .gitignore

```

.DS_Store
node_modules
/dist

# local env files
.env.local
.env.*.local

# Log files
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*

# Editor directories and files
.idea
.vscode
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

```

3.2.2 Текст файла проекта babel.config.js

```

module.exports = {
  presets: [
    '@vue/cli-plugin-babel/preset'
  ]
}

```

3.2.3 Текст файла проекта package.json

```

{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },
  "dependencies": {
    "axios": "^0.24.0",
    "core-js": "^3.6.5",
    "vue": "^2.6.11",
    "vue-router": "^3.2.0",
    "vuetify": "^2.4.0"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.5.0",
    "@vue/cli-plugin-router": "~4.5.0",
    "@vue/cli-service": "~4.5.0",
    "sass": "~1.32.0",
    "sass-loader": "^10.0.0",
    "vue-cli-plugin-vuetify": "~2.4.5",
    "vue-template-compiler": "^2.6.11",
    "vuetify-loader": "^1.7.0"
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}

```

3.2.4 Текст кода README.md

```

# client

## Project setup
```
npm install
```

### Compiles and hot-reloads for development
```
npm run serve
```

### Compiles and minifies for production
```
npm run build
```

### Customize configuration
See [Configuration Reference](https://cli.vuejs.org/config/).

```

3.2.5 Текст кода vue.config.js

```

module.exports = {
  transpileDependencies: [
    'vuetify'
  ],

```

```
devServer: {
  proxy: "http://localhost:5000",
},
}
```

3.2.6 Текст кода src/plugins/vuetify.js

```
import Vue from 'vue';
import Vuetify from 'vuetify/lib/framework';

Vue.use(Vuetify);

export default new Vuetify({
});
```

3.2.7 Текст кода src/router/index.

```
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";
import AddPost from "../views/AddPost.vue";
import Post from "../views/Post.vue";
import EditPost from "../views/EditPost.vue";

Vue.use(VueRouter);

const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/add-post",
    name: "add-post",
    component: AddPost,
  },
  {
    path: "/post/:id",
    name: "post",
    component: Post,
  },
  {
    path: "/edit-post/:id",
    name: "edit-post",
    component: EditPost,
  },
  {
    path: "/about",
    name: "About",
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () =>
      import(/* webpackChunkName: "about" */ "../views/About.vue"),
  },
];

const router = new VueRouter({
  mode: "history",
```

```

    base: process.env.BASE_URL,
    routes,
  });

```

```

export default router;

```

3.2.8 Текст кода src/views/About.vue

```

<template>
  <h1>О нас</h1>
</template>

```

3.2.9 Текст кода src/views/AddPost.vue

```

<template>
  <v-container>
    <v-row no-gutters>
      <v-col sm="10" class="mx-auto">
        <v-card class="pa-5">
          <v-card-title>Добавить мероприятие</v-card-title>
          <v-divider></v-divider>
          <v-form
            ref="form"
            @submit.prevent="submitForm"
            class="pa-5"
            enctype="multipart/form-data"
          >
            <v-text-field
              label="Заголовок"
              v-model="post.title"
              prepend-icon="mdi-note"
              :rules="rules"
            ></v-text-field>

            <v-text-field
              label="Категория"
              v-model="post.category"
              prepend-icon="mdi-view-list"
              :rules="rules"
            ></v-text-field>

            <v-textarea
              label="Описание"
              v-model="post.content"
              prepend-icon="mdi-note-plus"
              :rules="rules"
            ></v-textarea>

            <v-file-input
              @change="selectFile"
              :rules="rules"
              show-size
              counter
              multiple
              label="Выбрать фон мероприятия"
            ></v-file-input>

            <v-btn type="submit" class="mt-3" color="primary"
              >Добавить мероприятие</v-btn
          >

```



```

        </v-form>
      </v-card>
    </v-col>
  </v-row>
</v-container>
</template>

<script>
import API from "../api";
export default {
  data() {
    return {
      rules: [(value) => !!value || "This field is required!"],
      post: {
        title: "",
        category: "",
        content: "",
        image: "",
      },
      image: "",
    };
  },
  methods: {
    selectFile(file) {
      this.image = file[0];
    },
    async submitForm() {
      const formData = new FormData();
      formData.append("image", this.image);
      formData.append("title", this.post.title);
      formData.append("category", this.post.category);
      formData.append("content", this.post.content);
      if (this.$refs.form.validate()) {
        const response = await API.addPost(formData);
        this.$router.push({
          name: "Home",
          params: { message: response.message },
        });
      }
    },
  },
};
</script>

```

3.2.10 Текст кода views/EditPost.vue

```

<template>
  <v-container>
    <v-row no-gutters>
      <v-col sm="10" class="mx-auto">
        <v-card class="pa-5">
          <v-card-title>Редактирование параметров мероприятия</v-card-
title>
          <v-divider></v-divider>
          <v-form
            ref="form"
            @submit.prevent="updateForm"
            class="pa-5"
            enctype="multipart/form-data"
          >
            <v-text-field

```

4217.02067988.18 – 1777

```

        label="Заголовок"
        v-model="post.title"
        prepend-icon="mdi-note"
        :rules="rules"
    ></v-text-field>

    <v-text-field
        label="Категория"
        v-model="post.category"
        prepend-icon="mdi-view-list"
        :rules="rules"
    ></v-text-field>

    <v-textarea
        label="Описание"
        v-model="post.content"
        prepend-icon="mdi-note-plus"
        :rules="rules"
    ></v-textarea>

    <v-file-input
        @change="selectFile"
        show-size
        counter
        multiple
        label="Выбрать фон мероприятия"
    ></v-file-input>
    <v-img :src="`/${post.image}`" width="120"></v-img>
    <v-btn type="submit" class="mt-3" color="success"
        >Обновить мероприятие</v-btn
    >
    </v-form>
</v-card>
</v-col>
</v-row>
</v-container>
</template>

<script>
import API from "../api";
export default {
    data() {
        return {
            rules: [(value) => !!value || "This field is required!"],
            post: {
                title: "",
                category: "",
                content: "",
                image: "",
            },
            image: "",
        };
    },
    async created() {
        const response = await API.getPostByID(this.$route.params.id);
        this.post = response;
    },
    methods: {
        selectFile(file) {
            this.image = file[0];
        },
        async updateForm() {
            const formData = new FormData();

```

4217.02067988.18 – 1777

```

    formData.append("image", this.image);
    formData.append("title", this.post.title);
    formData.append("category", this.post.category);
    formData.append("content", this.post.content);
    formData.append("old_image", this.post.image);
    if (this.$refs.form.validate()) {
      const response = await API.updatePost(this.$route.params.id,
formData);
      this.$router.push({
        name: "Home",
        params: { message: response.message },
      });
    }
  },
};
</script>

```

3.2.11 Текст кода views/Home.vue

```

<template>
  <v-container>
    <v-alert
      border="left"
      close-text="Close Alert"
      color="green accent-4"
      dark
      dismissible
      v-if="this.$route.params.message"
    >
      {{ this.$route.params.message }}
    </v-alert>
    <v-row no-gutters>
      <v-col sm="4" class="pa-3" v-for="post in posts" :key="post._id">
        <v-card class="pa-1" :to="{ name: 'post', params: { id: post._id
} }">
          <v-img height="250" :src="'/${post.image}'"></v-img>
          <v-btn class="ml-4 mt-3" small outlined color="indigo">
            {{ post.category }}
          </v-btn>
          <v-card-title class="headline">
            {{ post.title }}
          </v-card-title>
          <v-card-text class="py-0">
            <p>{{ post.content.substring(0, 100) + "..."}</p>
          </v-card-text>
        </v-card>
      </v-col>
    </v-row>
  </v-container>
</template>

<script>
import API from "../api";
export default {
  name: "Home",
  data() {
    return {
      posts: [],
    };
  },
};

```

```

    async created() {
      this.posts = await API.getAllPost();
    },
  };
</script>

```

3.2.12 Текст кода views/Post.vue

```

<template>
  <v-container>
    <v-row no-gutters>
      <v-col sm="10" class="pa-4 mx-auto">
        <v-card class="pa-2">
          <v-img :src="`${post.image}`"> </v-img>
          <v-card-actions class="pb-0">
            <v-row class="mt-1 mx-1">
              <v-col sm="2">
                <v-btn small outlined color="primary">{{
                  post.category
                }}</v-btn>
              </v-col>
              <v-col sm="10" class="d-flex justify-end">
                <v-btn
                  color="success"
                  text
                  :to="{ name: 'edit-post', params: { id: post._id } }"
                >Редактировать</v-btn>
                >
                <v-btn color="red" text @click="removePost(post._id)"
                >Удалить</v-btn>
              </v-col>
            </v-row>
          </v-card-actions>
          <v-card-subtitle class="headline">
            <h3>{{ post.title }}</h3>
          </v-card-subtitle>
          <v-card-text class="grey--text">
            <p>{{ post.content }}</p>
            <p>{{ post.created }}</p>
          </v-card-text>
        </v-card>
      </v-col>
    </v-row>
  </v-container>
</template>

<script>
import API from "../api";

export default {
  data() {
    return {
      post: {},
    };
  },

  async created() {
    const response = await API.getPostByID(this.$route.params.id);
    this.post = response;
  }
}

```

```

    },
    methods: {
      async removePost(id) {
        const response = await API.deletePost(id);
        this.$router.push({
          name: "Home",
          params: { message: response.message },
        });
      },
    },
  };
</script>

```

3.2.13 Текст кода api.js

```

import axios from "axios";
const url = "/api/post";

export default class API {
  // Получить все Хакатоны с сервера
  static async getAllPost() {
    const res = await axios.get(url);
    return res.data;
  }

  // Получить один Хакатон с сервера
  static async getPostByID(id) {
    const res = await axios.get(`${url}/${id}`);
    return res.data;
  }

  //Добавление в базу данных
  static async addPost(post) {
    const res = await axios.post(url, post);
    return res.data;
  }

  //Обновления в базе данных
  static async updatePost(id, post) {
    const res = await axios.patch(`${url}/${id}`, post);
    return res.data;
  }

  // Удалить один Хакатон с сервера
  static async deletePost(id) {
    const res = await axios.delete(`${url}/${id}`);
    return res.data;
  }
}

```

3.2.14 Текст кода main.js

```

import Vue from 'vue'
import App from './App.vue'
import router from './router'
import vuetify from './plugins/vuetify'

Vue.config.productionTip = false

new Vue({

```

```

router,
vuetify,
render: h => h(App)
}).$mount('#app')

```

3.2.15 Текст кода App.vue

```

<template>
  <v-app id="inspire">
    <v-navigation-drawer v-model="drawer" app>
      <v-list-item>
        <v-list-item-content>
          <v-list-item-title> DeCode </v-list-item-title>
          <v-list-item-subtitle> Конструктор Хакатонов </v-list-item-
subtitle>
        </v-list-item-content>
      </v-list-item>
      <v-divider></v-divider>
      <v-list dense>
        <v-list-item-group color="primary">
          <v-list-item v-for="(item, i) in items" :key="i"
:to="item.link" link>
            <v-list-item-icon>
              <v-icon v-text="item.icon"></v-icon>
            </v-list-item-icon>
            <v-list-item-content>
              <v-list-item-title v-text="item.title"></v-list-item-
title>
            </v-list-item-content>
          </v-list-item>
        </v-list-item-group>
      </v-list>
    </v-navigation-drawer>
    <v-app-bar app>
      <v-app-bar-nav-icon @click="drawer = !drawer"></v-app-bar-nav-
icon>
      <v-toolbar-title>Конструктор Хакатонов</v-toolbar-title>
    </v-app-bar>
    <v-main>
      <router-view></router-view>
    </v-main>
  </v-app>
</template>

<script>
export default {
  data: () => ({
    drawer: null,
    items: [
      { title: "На главную", icon: "mdi-home", link: "/" },
      {
        title: "Создать Мероприятие",
        icon: "mdi-note-plus",
        link: "/add-post",
      },
      { title: "О нас", icon: "mdi-help-box", link: "/about" },
    ],
  }),
};
</script>

```

4 Руководство программиста

4.1 Назначение и условия применения программы

Полное наименование: web-сервис «Конструктор хакатонов».

Web-сервис «Конструктор хакатонов» представляет из себя сайт, который упростит процесс организации различного рода мероприятий и предоставит доступ к единой платформе для их проведения, как для участников, так и для организаторов.

Программное обеспечение необходимое для функционирования модулей:

- редактор кода Microsoft Visual Studio Code;
- браузер Google Chrome для отображения результатов разработки;
- графический редактор для дизайна и прототипирования Figma;
- графический редактор базы данных MongoDB Compass.

Для корректной работы программного обеспечения требуется:

- процессор Intel Core i3 4000M;
- оперативная память 4 Гб и больше;
- свободное дисковое пространство 4 Гб и больше;
- видеоадаптер SVGA и выше;
- клавиатура;
- манипулятор «мышь».

4.2 Характеристика программы

4.2.1 Описание интерфейса

Главная веб-страница сайта является лендингом, то есть это длинная страница, на которой расположена информация о ранее созданных мероприятиях. Слева расположено меню с помощью которого осуществляются переходы на другие разделы сайта.

Переходы по разделам реализована с помощью визуального каркаса «base wireframe» фреймворка «Vuetyfy». Разметка каркаса располагается в теге <template> главного компонента приложения App.vue.

Далее с помощью технологии связывания v-bind в данный <template> будут встраиваться компоненты (страницы: «О нас», «Создание мероприятия»)

Первый компонент Post.vue располагается внутри страницы «Главная». В шаблоне данного компонента реализуется с помощью тега <v-card-actions> (рисунок 4.1).

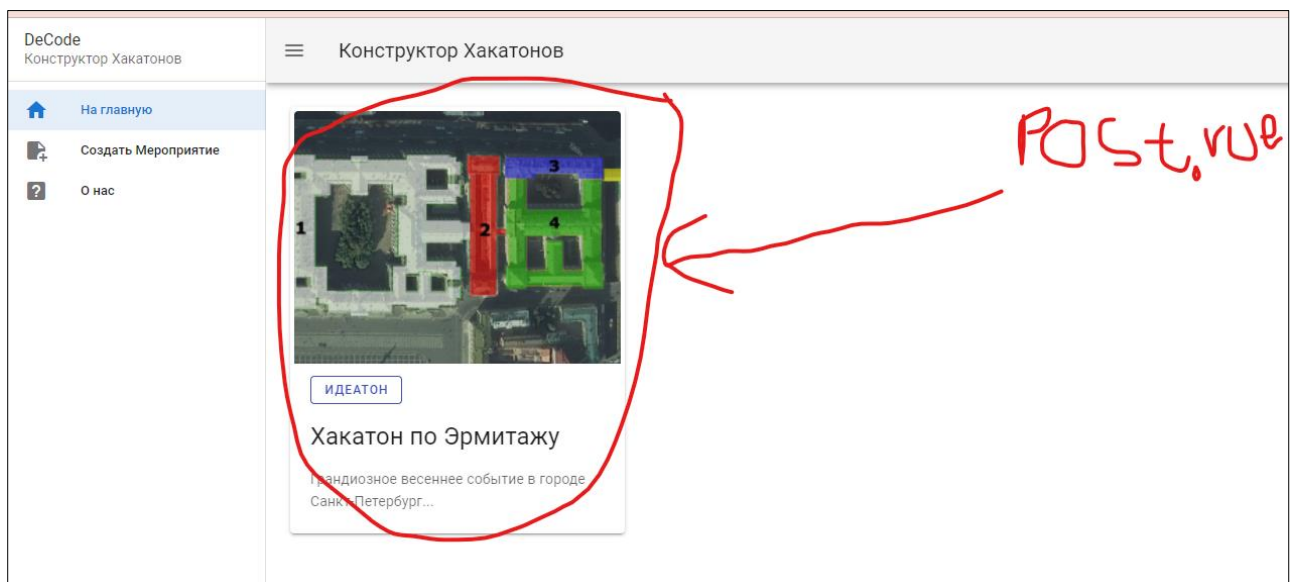


Рисунок 4.1 – Компонент карточки мероприятия

На странице создания мероприятия встроен компонент AddPost.vue. Шаблон компонента использует тег <v-card>. Название и категория реализованы с помощью тега <v-text-field>. Тег <v-textarea> используется для поля описания. Тег <v-file-input> реализует возможность выбор файла.

Кнопка создания мероприятия реализована с помощью тега <v-btn> (рисунок 4.2).

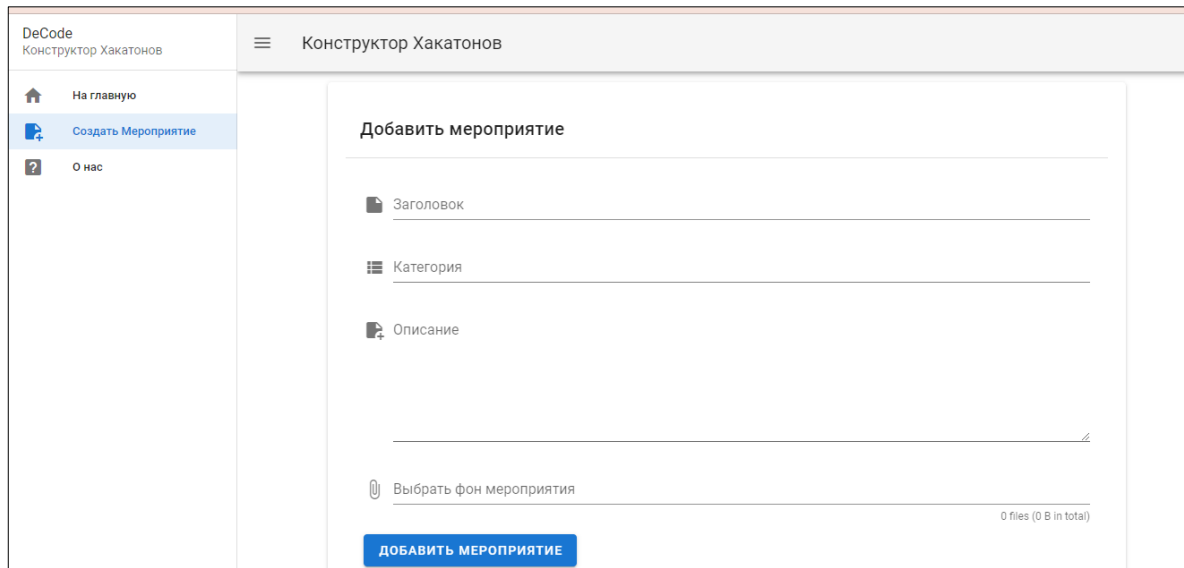


Рисунок 4.2 – Компонент создания

На странице «О нас» компонент `About.vue` в шаблоне имеет один тег `<h1>` (рисунок 4.3).

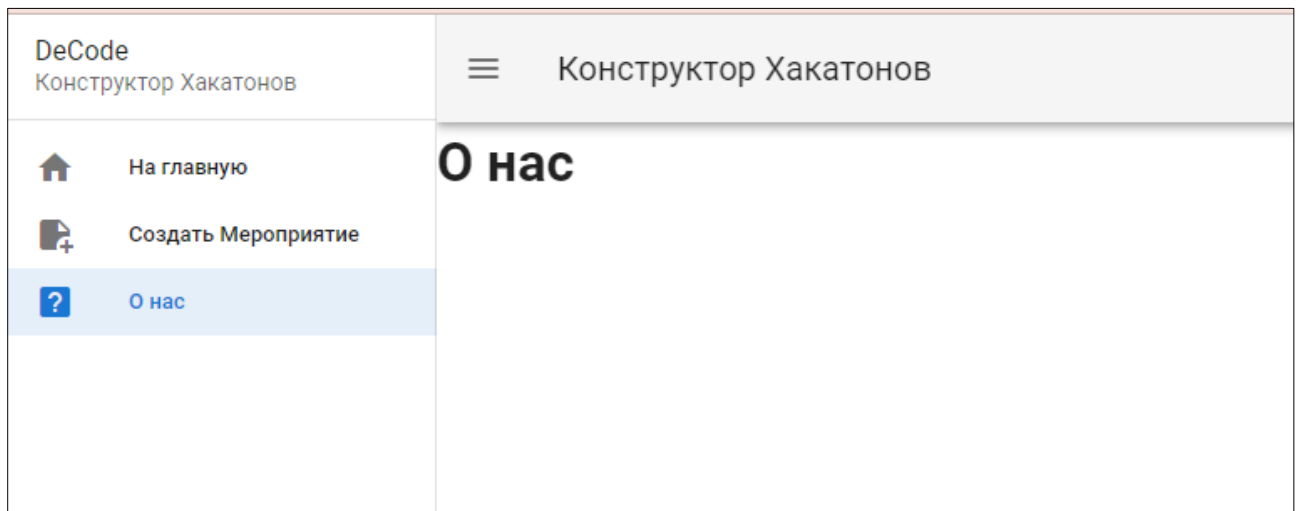


Рисунок 4.3 – Блок «О нас»

4.2.2 Подсистема работы с базой данных

Разработка и взаимодействие с базой данных происходит в отдельном приложении «server». База данных будет размещаться на `localhost:27017`. Создание базы данных с одной коллекцией показано на рисунке 4.4.

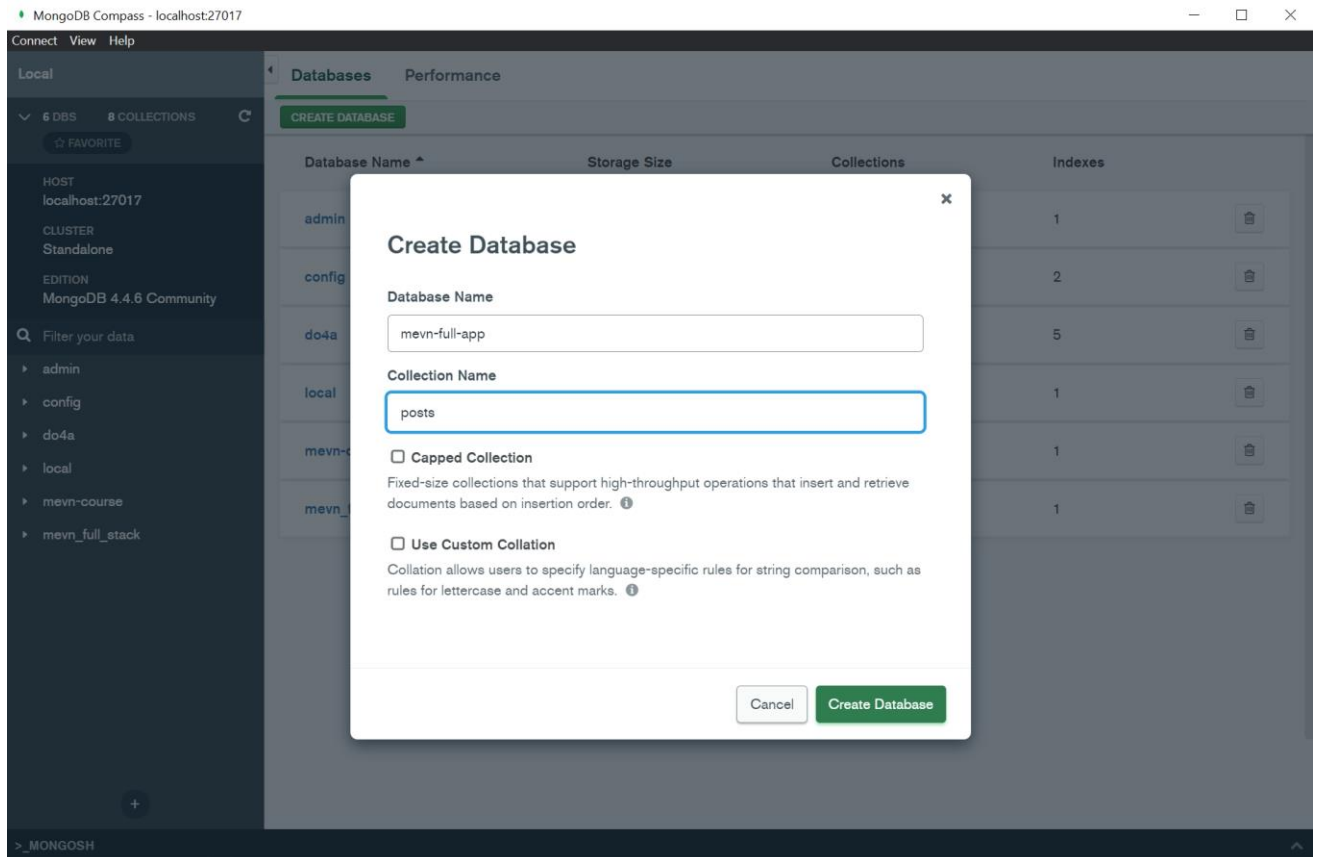


Рисунок 4.4 – Создание базы данных

Структура серверной части показана на рисунке 4.5.

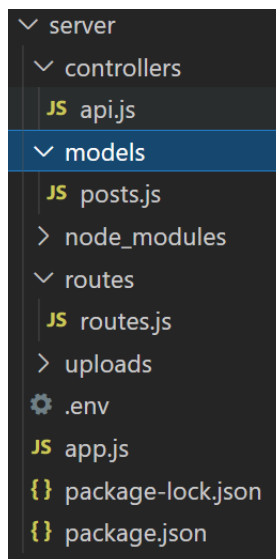
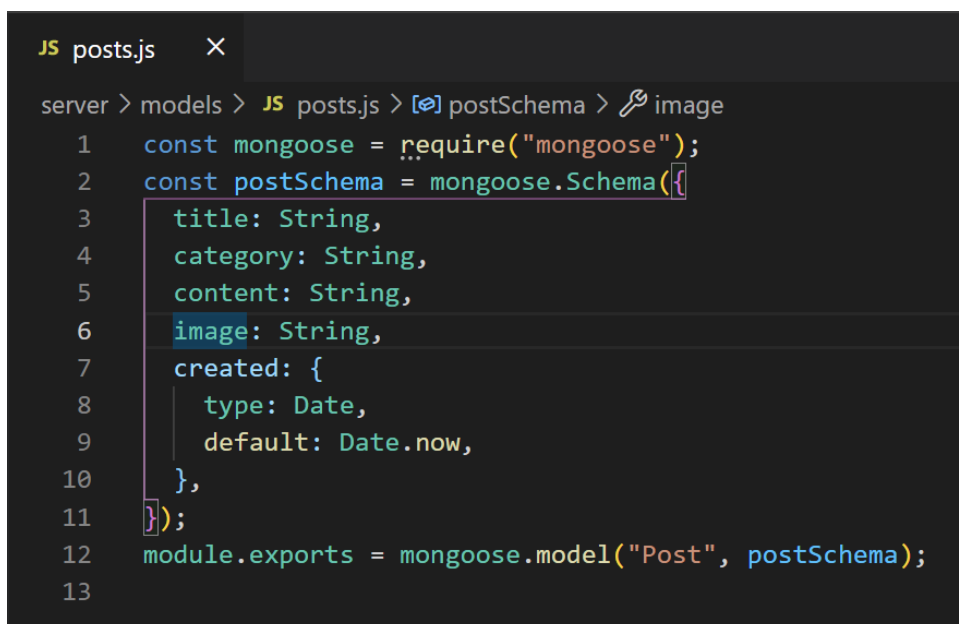


Рисунок 4.5 – Создание серверной части

`package.json` – это файл инициализации проекта, который очень плотно связан с директорией `node_modules`. В нем содержится информация о проекте (название, автор, версия), прописаны скрипты, выполняющиеся NPM, а также все установленные зависимости, которые как раз содержатся в `node_modules`. Зависимости обозначены значениями ключей «`dependencies`» (зависимости, используемые на продакшн) и «`devDependencies`» (зависимости, используемые при разработке). Этот файл нужен прежде всего для того, чтоб проект можно было развернуть на любой машине, одной лишь командой `npm i`. Можно попробовать удалить директорию `node_modules`, а затем выполнить команду `npm i` в корне проекта: она заново подтянет все необходимые зависимости, которые указаны в `package.json`.

`package-lock.json` – моментальный снимок всего дерева зависимостей. Дело в том, что пакеты имеют зависимости верхнего уровня. При установке это не заметно, но всегда можно посмотреть в `package-lock.json`.

`models.js` представляет собой структуру объектов с необходимыми атрибутами. Структура модели в JavaScript коде представлена на рисунке 4.6.



```
JS posts.js  X
server > models > JS posts.js > [🔍] postSchema > 🔗 image
1  const mongoose = require("mongoose");
2  const postSchema = mongoose.Schema({
3    title: String,
4    category: String,
5    content: String,
6    image: String,
7    created: {
8      type: Date,
9      default: Date.now,
10   },
11 });
12 module.exports = mongoose.model("Post", postSchema);
13
```

Рисунок 4.6 – Объектная модель

app.js – данный файл предоставляет подключения ко всем необходимым зависимостям (например к Mongoose представляет специальную ODM библиотеку для работы с MongoDB, которая позволяет сопоставлять объекты классов и документы коллекций из базы данных.), а так же само подключение к базе (Рисунок 4.7)

```
//database connection
mongoose
  .connect(process.env.DB_URI, {
    useUrlParser: true,
    useUnifiedTopology: true,
    useFindAndModify: true,
    useCreateIndex: true,
  })
  .then(() => console.log("Соединение с MongoDB есть!"))
  .catch((err) => console.log(err));

//routes prefix
app.use("/api/post", require("./routes/routes"));

//Start even http://localhost:5000/
app.listen(port, () => console.log(`Сервер работает на хосте : ${port}`));
```

Рисунок 4.8 – Подключение к базе

api.js – описывает все необходимые методы представляющие из себя API запросы, а так же обработчики событий на удачное или неудачное действие в базу.

4.3 Обращение к программе

Запуск приложения осуществляется с помощью ввода в адресную строку браузера соответствующего URL-запроса (рисунок 4.9).

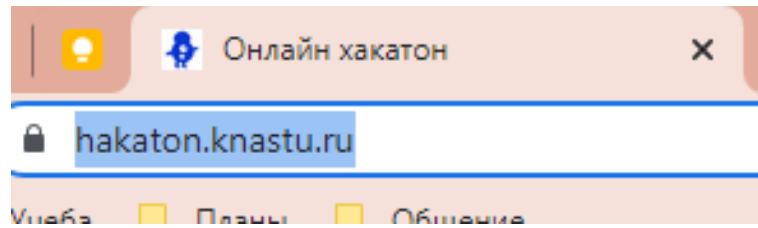


Рисунок 4.9 – запуск web-приложения

4.4 Выходные и выходные данные

Входными данными web-сервиса являются, данные при регистрации пользователя и данные, вносимые при создании мероприятия (текст, прикрепляемые файлы). Данные вводятся пользователем с помощью клавиатуры, и манипулятор «мышь».

Выходными данными web-сервиса являются созданные мероприятия различными организаторами (карточки мероприятий), которые предоставляют возможность ознакомления с мероприятием (просмотр названия мероприятия, описания, формулировок заданий) и скачивание прикрепленных файлов. Данные участников, такие как контактные данные, фамилия, имя, отчество, так же являются выходными.

4.5 Сообщения

Информационные сообщения для программиста будут отображены в консоли, вызванные программными блоками обработки исключений. «Соединение с MongoDB есть!»: Сообщение появляется если нет ошибок при подключении к базе данных.

5 Руководство оператора

5.1 Назначение программы

Наименование программного обеспечения: «Web-сервис Конструктор хакатонов».

Краткое наименование – «Конструктор хакатонов».

Web-сервис «Конструктор хакатонов» представляет из себя сайт, который упростит процесс организации различного рода мероприятий и предоставит доступ к единой платформе для их проведения, как для участников, так и для организаторов.

5.2 Условия выполнения программы

Для корректного взаимодействия пользователя с сайтом необходимо:

- операционная система Windows 7 и выше;
- web-браузер с выходом в Интернет.

5.3 Выполнение программы

5.3.1 Предварительные настройки

Для корректной работы web-сервиса необходимо проверить соединение с сетью Интернет.

Чтобы открыть приложение в браузере, необходимо написать в поисковой строке URL адрес сайта.

5.3.2 Работа на главной странице сайта

При открытии web-сервиса появляется главная страница где отображаются данные загруженные из базы данных (рисунок 5.1).

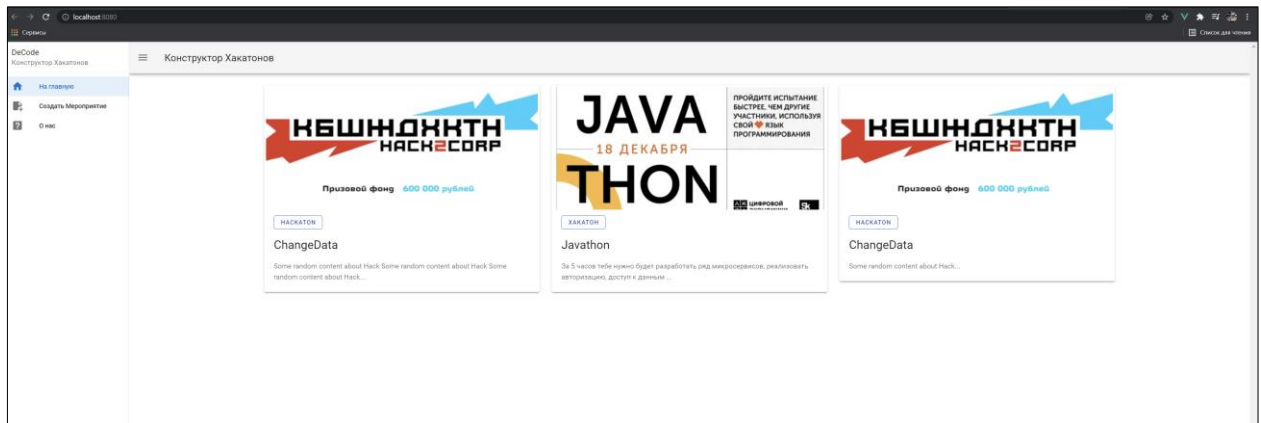


Рисунок 5.1 – Загрузка данных с базы

При открытии мероприятия, его карточка раскрывается. Есть возможность редактирования и удаления мероприятия.

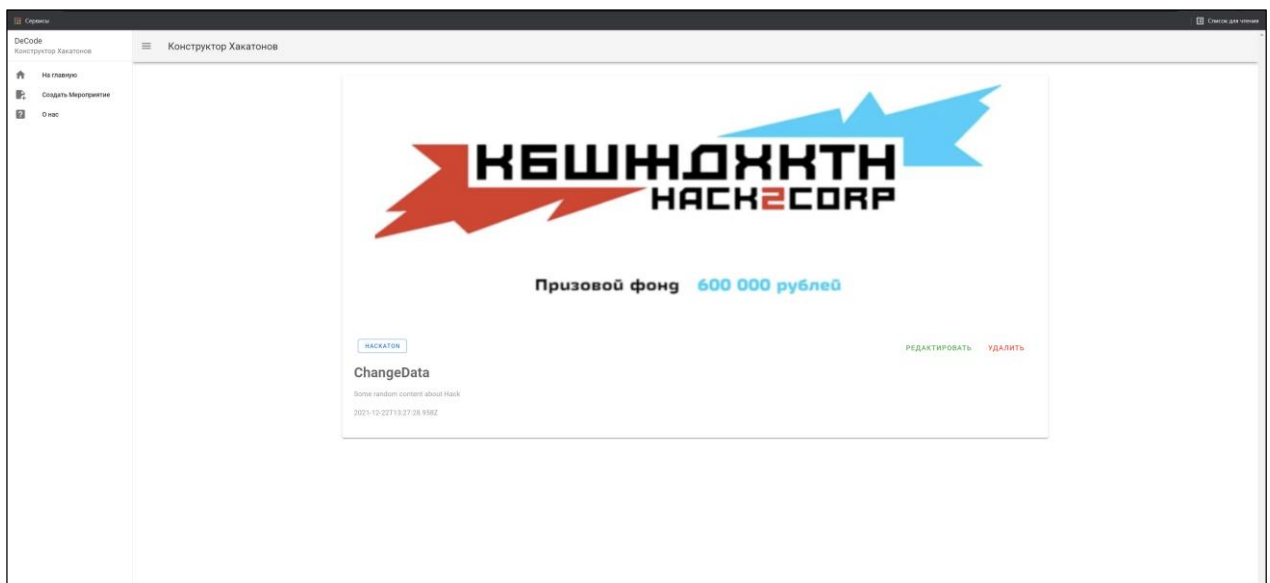


Рисунок 5.2 – Открытие мероприятия

После редактирования мероприятия на главной странице появляется уведомление об успешном редактировании мероприятия.

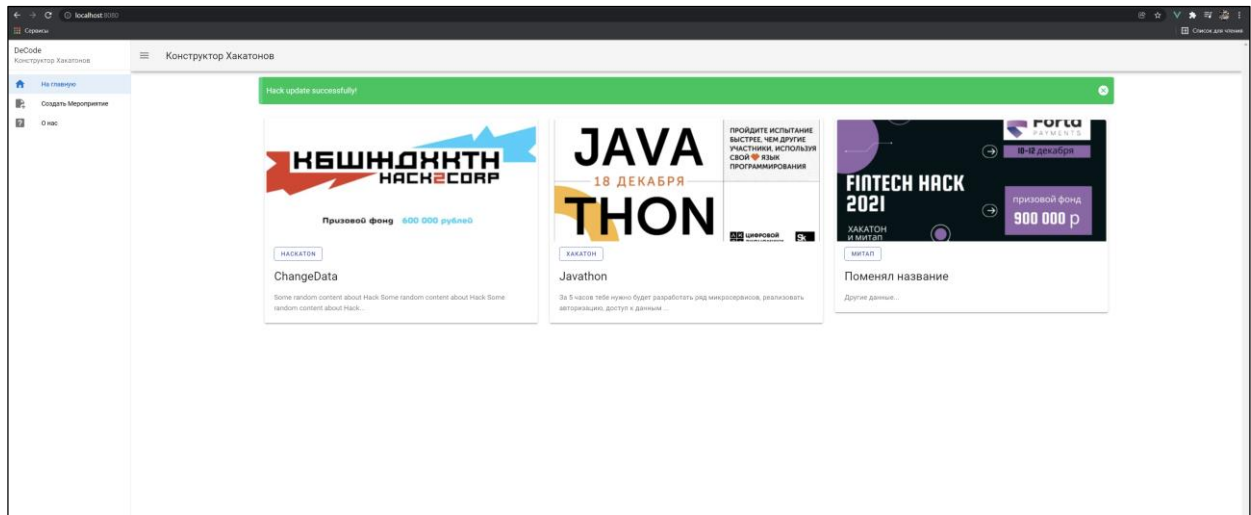


Рисунок 5.3 – Главная страница после редактирования мероприятия

После удаления мероприятия на главной странице появляется уведомление об успешном удалении мероприятия.

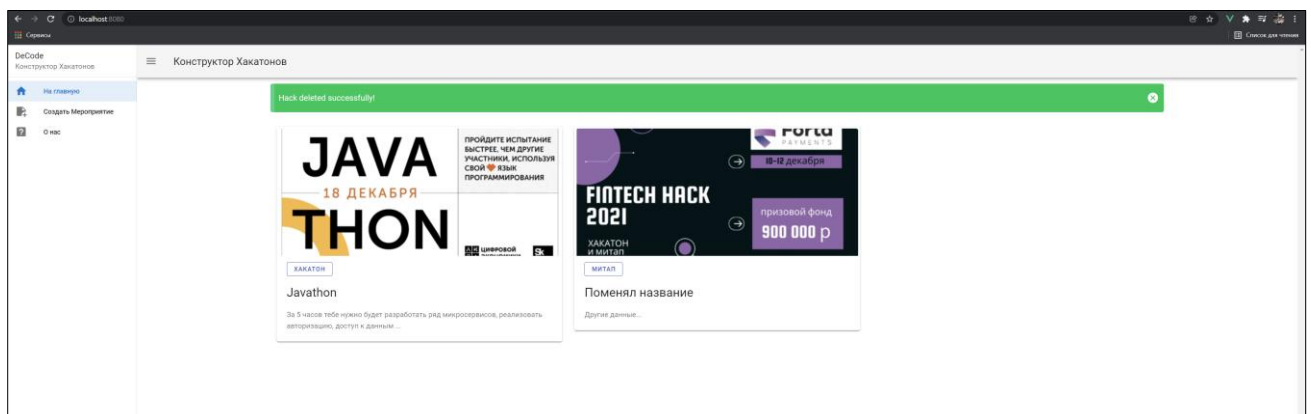


Рисунок 5.4 – Главная страница после удаления мероприятия

Меню создания открывается отдельной вкладкой, предоставляя пустые поля для заполнения. После заполнения полей и загрузки необходимых файлов появляется возможность внести новую запись в коллекцию, тем самым создав новое мероприятие.

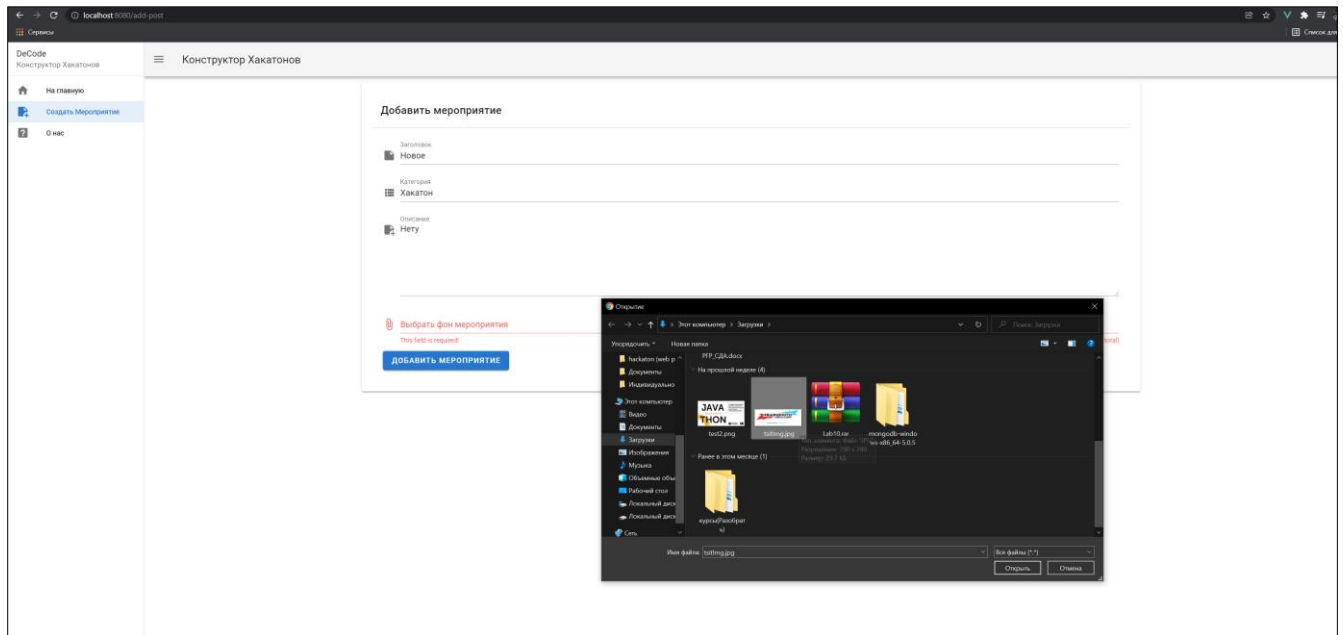


Рисунок 5.5 – Создание мероприятия

После создания мероприятия на главной странице появляется уведомление об успешном создании мероприятия.

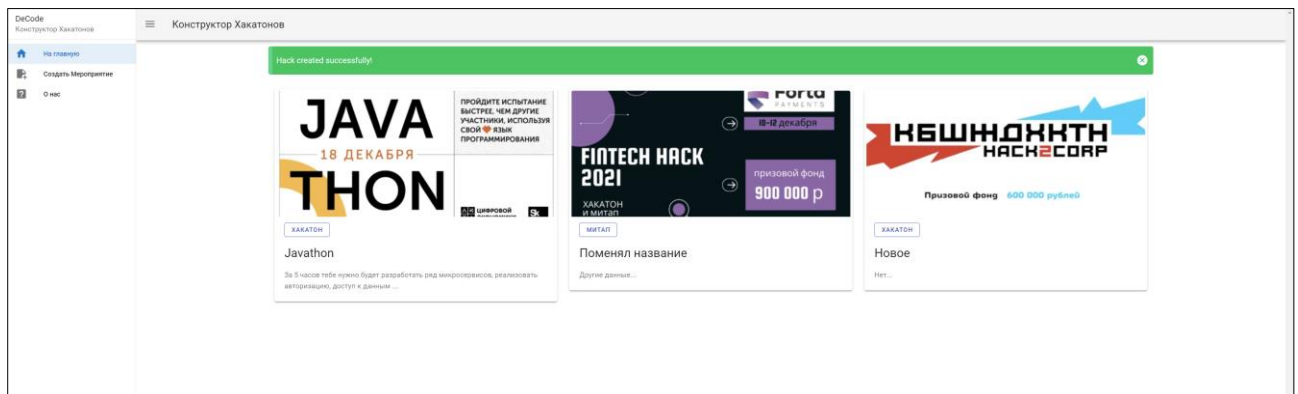


Рисунок 5.6 – Главная страница после создания мероприятия

5.4 Сообщения оператору

После создания мероприятия оператор получает сообщение: «Hack created successfully».

После редактирования карточки мероприятия оператор получает сообщение: «Hack update successfully».

После удаления мероприятия оператор получает сообщение: «Hack delete successfully».

Заключение

Целью работы, выполняемой в комплексном проекте является разработка web-сервиса.

В ходе выполнения комплексного проекта была изучена предметная область, а также проанализированы аналоги. Определены методы разработки и функциональное назначение проекта. Приложение разрабатывается на языках HTML, CSS и JavaScript с использованием фреймворка vue.js в редакторе кода Visual Studio Code. Хранение данных производится в документно-ориентированной базы данных MongoDB.

Таким образом, с помощью использования правильно подобранных программных инструментов и составленной объектной модели, включающая в себя полный перечень данных касающейся предметной области, появляется возможность создать программный продукт, соответствующий требованиям надежности, функциональности, производительности, удобство сопровождения, переносимости и масштабируемости.

В дальнейшем разработка будет продолжаться и в конце будет произведено тестирование web-приложения.

Список использованных источников

- 1 ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению.
- 2 ГОСТ 19.402-78 Единая система программной документации. Описание программы.
- 3 ГОСТ Р ИСО/МЭК 15271-02 Процессы жизненного цикла программных средств
- 4 ГОСТ 19.201 – 78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
- 5 ГОСТ 19.301 – 79 Единая система программной документации. Программа и методика испытаний. Требования к содержанию и оформлению.
- 6 ГОСТ 19.401 – 78 Единая система программной документации. Требования к содержанию и оформлению.
- 7 ГОСТ 19.404 – 79 Единая система программной документации. Пояснительная записка. Требования к содержанию и оформлению.
- 8 ГОСТ 19.504-79 Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению.
- 9 ГОСТ 19.505-79 Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению.