

**Отчет по студента группы ИСТ-412 Кузина Ивана**  
**по проекту „Веб-сервис для голосования в реальном времени“**

## **План отчета**

### **1. Цели и задачи**

- Определение целей и задач проекта.

### **2. Описание проекта**

- Описание функционала веб-сервиса для голосования.
- Целевая аудитория и примеры использования.

### **3. Используемые технологии**

- Выбор базы данных (MySQL) и обоснование.
- Разработка серверной части (Node.js, Express, WebSocket).
- Разработка клиентской части (React).
- Использование Docker для сборки и развертывания приложения.

### **4. Структура программы**

- Описание верхнего уровня проекта (настройки, зависимости).
- Структура папки backend (routers, controllers, middleware, wsfunctions).
- Структура папки frontend (vite настройки, public, src).

### **5. Модули программы**

- Модуль создания голосования.
- Модули голосования (взаимодействие пользователя и организатора).
- Модуль отображения результатов голосования.

### **6. Демонстрация работы**

- Скриншоты или видео функционала веб-сервиса.

### **7. Проблемы, с которыми столкнулись**

- Описание возникших сложностей и пути их решения.

### **8. Результаты и выводы**

- Оценка итогов проекта и уроки, полученные в процессе.

### **9. Используемые источники**

- Литература и ресурсы, примененные в работе.

### **10. Ссылка на репозиторий**

## 1. Цели и задачи

**Цель проекта:** создать веб-сервис для голосования в реальном времени.

**Задачи:**

- Определение логики работы веб-сервиса.
- Создание базы данных.
- Разработка серверной части.
- Разработка клиентской части.

## 2. Описание проекта

Веб-сервис предоставляет возможность создавать собственные голосования с индивидуальными вопросами и ответами. В сервисе предусмотрена базовая кастомизация голосований, например, возможность прикрепления к вопросам соответствующих картинок или GIF-анимаций.

Пользователи могут подключаться к голосованию и выбирать один из предложенных ответов. Для организатора голосования доступна функция просмотра результатов по окончании голосования.

Данный веб-сервис предназначен для локального корпоративного использования. Например, представьте ситуацию, когда на собрании один из сотрудников рассказывает о новостях компании и одновременно демонстрирует презентацию. В начале выступления он включает в презентацию QR-код, который ссылается на наше голосование. Коллеги сканируют код и могут выразить свое мнение по вопросам, представленным в голосовании, выбрав один из предложенных вариантов ответов.

## 3. Используемые технологии

### База данных: MySQL

В качестве базы данных была выбрана MySQL по следующим причинам:

1. **Надежность:** MySQL является одной из самых популярных и проверенных временем систем управления базами данных. Она отлично зарекомендовала себя в различных проектах и приложениях.
2. **Производительность:** MySQL обеспечивает высокую скорость выполнения запросов, что особенно важно для веб-сервисов с интерактивной природой, таких как голосование в реальном времени.
3. **Гибкость и расширяемость:** MySQL поддерживает множество возможностей для работы с данными, включая масштабирование и интеграцию с другими технологиями, что делает ее подходящей для роста проекта.

### Серверная часть: Node.js и Express

Для разработки серверной части использовались Node.js и Express.

- **Node.js** позволяет создавать масштабируемые сетевые приложения, использующие неблокирующую архитектуру. Это особенно важно для нашего проекта, где требуется обработка многопользовательских соединений и запросов в режиме реального времени.

- **Express** выбран из-за своей популярности, широкого комьюнити и гибкости. Этот фреймворк предоставляет удобные средства для маршрутизации и обработки HTTP-запросов, что ускоряет разработку и упрощает поддержку серверного кода.

### Docker

В проекте также используется Docker, что позволяет упростить процесс сборки и развертывания приложения. С Docker можно создать изолированные контейнеры, в которых будут находиться все необходимые зависимости, что обеспечивает одинаковую среду как для разработки, так и для продакшна. Это значительно упрощает настройку окружения и тестирование приложения на различных платформах.

### WebSocket

Для реализации соединения в реальном времени был использован WebSocket. Эта технология позволяет установить двустороннюю связь между клиентом и сервером, обеспечивая мгновенную передачу данных и обновления без необходимости перезагрузки страницы.

### Клиентская

### часть:

### React

Для разработки клиентской части был использован React. Это решение было выбрано из-за реактивной природы библиотеки, которая позволяет обновлять только необходимые компоненты, не перерисовывая всю страницу. React облегчает создание динамичного пользовательского интерфейса и позволяет сосредоточиться на разработке клиентского кода, а не на оптимизации проекта. Мне очень понравилась работа с этой библиотекой, так как она значительно упрощает процесс разработки.

## 4. Структура программы

На верхнем уровне проекта находятся следующие файлы и папки:

- **Docker настройки:** файлы, необходимые для настройки контейнеров.
- **.env файл:** содержит настройки окружения, такие как параметры подключения к базе данных.
- **JSON файлы:** хранят зависимости проекта.
- **Папка frontend :** содержит клиентскую часть приложения.
- **Папка backend :** содержит серверную часть приложения, включая файлы с зависимостями и настройками.

### Структура папки backend

Внутри папки backend расположены следующие ключевые элементы:

- **routers:** файлы, определяющие маршрутизацию запросов.
- **controllers:** модули, обрабатывающие бизнес-логику.
- **middleware:** промежуточные обработчики для управления запросами.
- **wsfunctions:** функции для работы с WebSocket, обеспечивающие реальное время.

### Структура папки frontend

В папке frontend вы найдете:

- **vite настройки:** файлы конфигурации для сборки проекта.
- **public:** папка для медиа файлов (например, иконок или изображений).

- **src:** папка, где находятся компоненты и страницы приложения.
  - **components:** повторно используемые элементы интерфейса.
  - **pages:** страницы приложения.
  - **api:** папка с запросами к серверу.

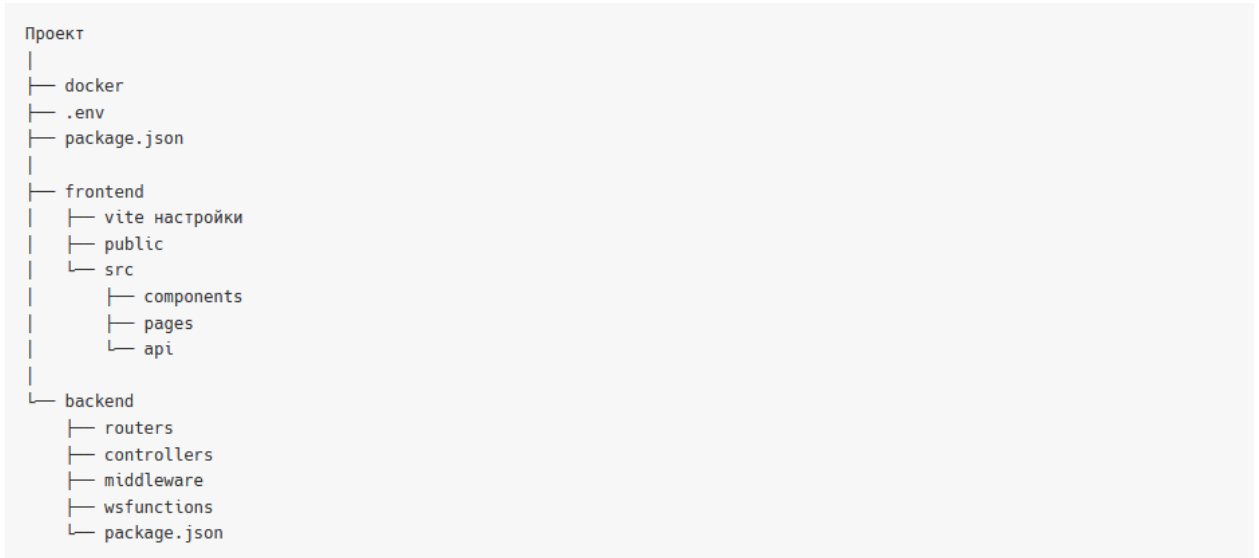


Рис. 1 Примерная схема проекта

## 5. Модули программы

### 1. Модуль создания голосования:

- На стороне клиента существует отдельная страница, где пользователи могут добавлять и редактировать голосования.
- При создании нового голосования клиент отправляет запрос на сервер, включающий название и описание голосования.
- Сервер обрабатывает запрос, взаимодействуя с базой данных, и возвращает обновленные или вновь созданные данные клиенту.

### 2. Модули голосования:

- Во время голосования организатор может переключать вопросы вперед и назад.
- Участник голосования может выбрать один из предложенных ответов. После подтверждения выбор сохраняется в базе данных, и повторное голосование по тому же вопросу становится невозможным.
- По окончании голосования пользователи перенаправляются на страницу с результатами, где они не могут вернуться к прежнему голосованию (можно начать новое голосование в тестовой версии).
- Организатор голосования может видеть результаты в виде диаграммы, а также выбирать, по какому вопросу хочет просмотреть статистику.

## 6. Демонстрация работы

### 1. Список голосований на странице администратора

На странице администратора отображается список всех доступных голосований.

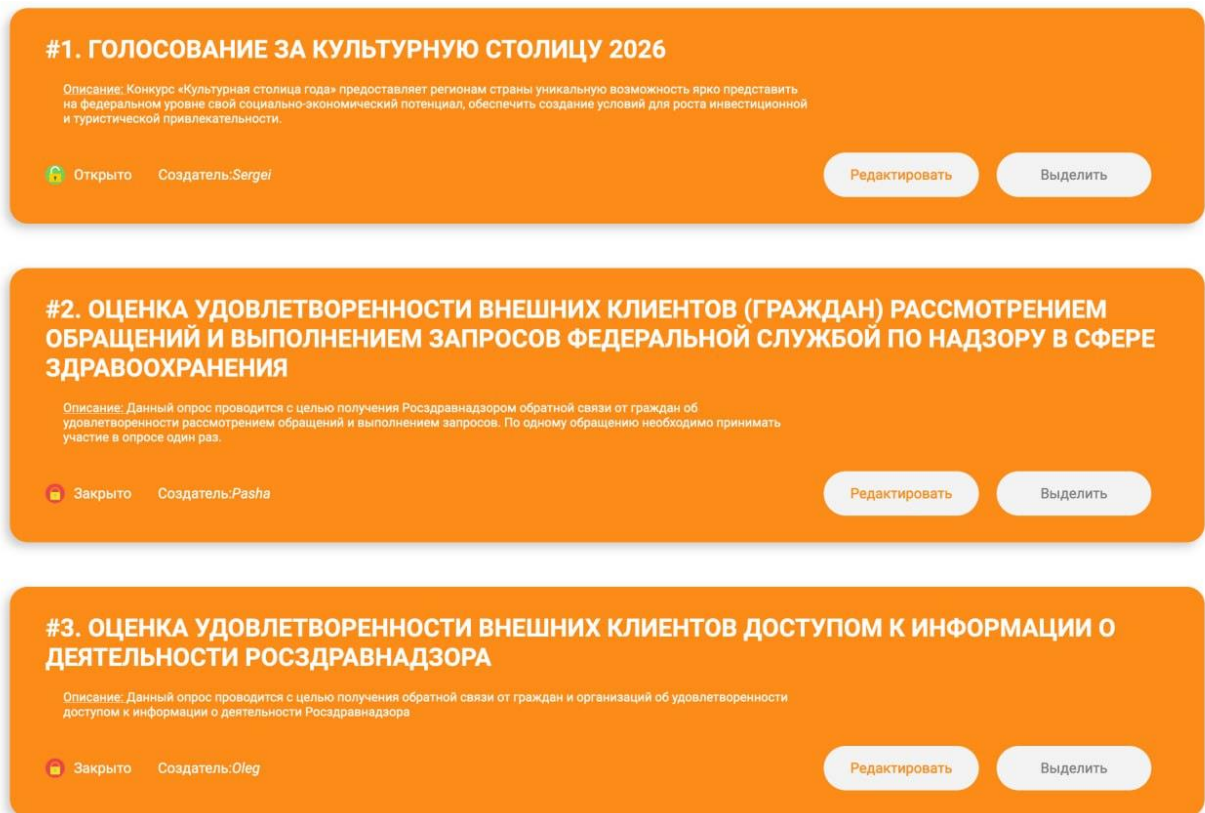


Рис. 2 Список всех доступных голосований

### 2. Создание голосования

С самого начала я продемонстрирую процесс создания голосования.



Рис. 3 Форма для создания голосования

### 3. Управление вопросами и ответами

В созданном голосовании администратор может добавлять и изменять вопросы, а также управлять ответами к этим вопросам.

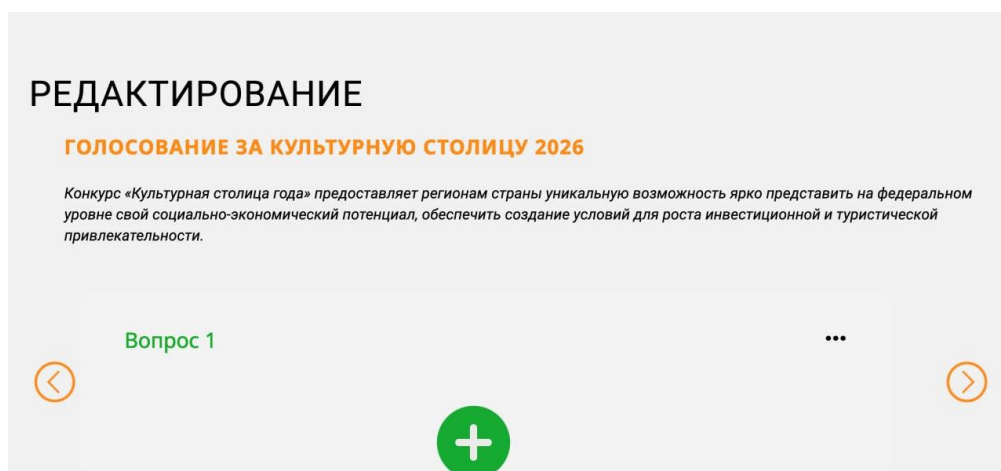



Рис. 5 Интерфейс редактирования вопросов



Рис. 6 Интерфейс редактирования ответов

#### 4. Список голосований на клиентской странице

Посмотрим, как выглядит список всех голосований на клиентской странице.


Голосования

Текущее время: 17:31:46

**Голосование за Культурную столицу 2026**

Конкурс «Культурная столица года» предоставляет регионам страны уникальную возможность ярко представить на федеральном уровне свой социально-экономический потенциал, обеспечить создание условий для роста инвестиционной и туристической привлекательности.


Автор голосования: Sergei

 Голосование открыто

**Оценка удовлетворенности внешних клиентов (граждан) рассмотрением обращений и выполнением запросов Федеральной службой по надзору в сфере здравоохранения**

Данный опрос проводится с целью получения Росздравнадзором обратной связи от граждан об удовлетворенности рассмотрением обращений и выполнением запросов. По одному обращению необходимо принимать участие в опросе один раз.

Автор голосования: Pasha

 Голосование закрыто

**Оценка удовлетворенности внешних клиентов доступом к информации о деятельности Росздравнадзора**

Данный опрос проводится с целью получения обратной связи от граждан и организаций об удовлетворенности доступом к информации о деятельности Росздравнадзора

Автор голосования: Oleg



 Голосование закрыто

Рис. 7 Список голосований для клиентов

#### 5. Начало голосования и QR-код

Когда администратор переходит к голосованию, он имеет возможность его запустить. На этом экране отображается QR-код, по которому пользователи могут перейти к голосованию.


Голосования

Текущее время: 17:32:41

**ГОЛОСОВАНИЕ ЗА КУЛЬТУРНУЮ СТОЛИЦУ 2026**

Конкурс «Культурная столица года» предоставляет регионам страны уникальную возможность ярко представить на федеральном уровне свой социально-экономический потенциал, обеспечить создание условий для роста инвестиционной и туристической привлекательности.

Автор голосования: admin



Сканируйте QR-код, чтобы перейти на страницу голосования

Начать опрос

Рис. 8 Админская страница с QR-кодом

## 6. Интерфейс во время голосования

После начала голосования интерфейсы выглядят следующим образом: слева — интерфейс администратора, справа — интерфейс клиента.

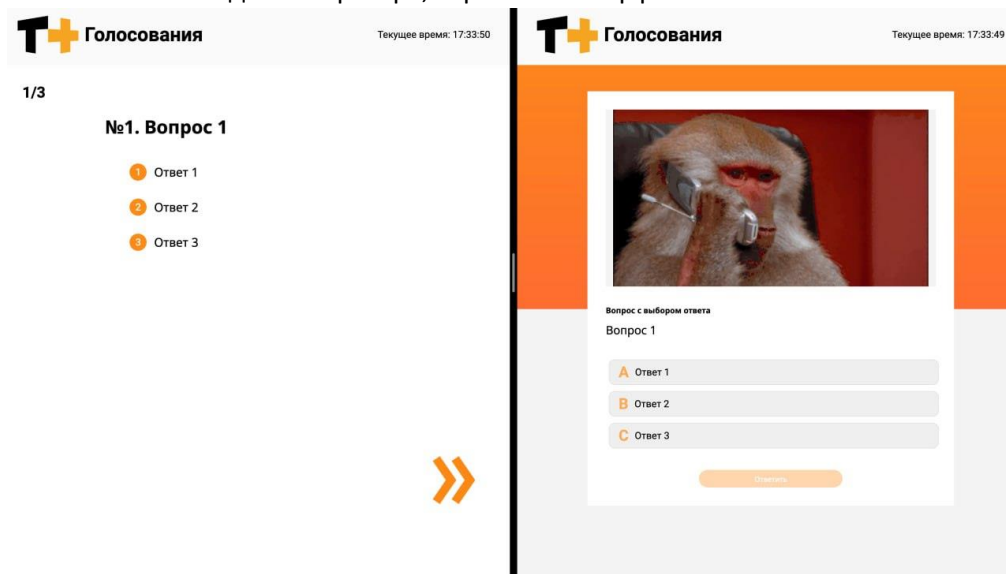


Рис. 9 Интерфейс админа и клиента (Вопрос 1)

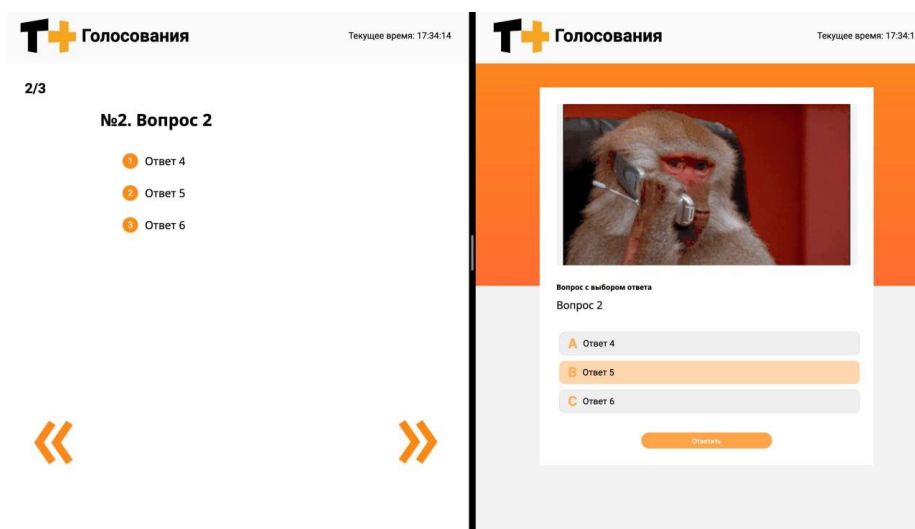


Рис. 10 Интерфейс админа и клиента (Вопрос 2)



## 7. Страница для пользователей после завершения голосования

Когда голосование завершается, пользователи видят такую страницу.

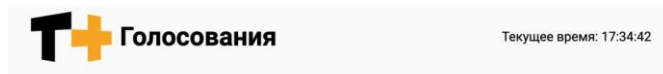


Рис. 11 Страница окончания голосования для клиентов

## 8. Страница с результатами для администратора

Администратор перенаправляется на страницу с результатами голосования, которая выглядит следующим образом:



Рис. 12 Страница с результатами голосования

## 7. Проблемы, с которыми столкнулись

### 1. Настройка Docker:

Первая проблема, с которой я столкнулся, заключалась в настройке Docker. Учитывая, что у меня не было опыта работы с этой технологией, разобраться было довольно сложно. Я сталкивался с непривычными ошибками и запутанной документацией. Мне потребовалось 2-3 дня, чтобы исследовать различные настройки и попытаться найти решения, чтобы проект, наконец, заработал. Даже сейчас некоторые аспекты работают с "костылями".

### 2. Знакомство с серверной частью:

В начале разработки у меня было ограниченное понимание работы серверной части приложения. Я не знал, что такое роутинг, middleware и как функционируют контроллеры. Приходилось разбираться с этими концепциями по ходу разработки, что потребовало значительных усилий.

### 3. Изучение библиотеки React:

Похожей трудностью стало знакомство с библиотекой React. Хотя это было легче, чем работа с сервером, понимание концепции хуков (таких как `useEffect`, `useLocation`, `useState`), а также создание кастомных хуков и организация запросов с клиента на сервер требовали времени на изучение и практику для их эффективного применения.

### 4. Реализация обновления данных в реальном времени:

Когда я более-менее освоился с работой сервера, я понял, что реализовать свою идею на простом сервере не удастся, так как мне требовалось обновление данных в реальном времени. Это подтолкнуло меня к изучению технологии WebSocket, что добавило ещё одну составляющую в проект.

### 5. Индивидуальная разработка и дизайн:

Работа в одиночку означала, что мне нужно было самостоятельно продумывать и разрабатывать дизайн приложения, что оказалось довольно сложно. Разработка дизайна занимала значительное количество времени и требовала дополнительных усилий, чтобы достичь приемлемого результата.

## 8. Результаты и выводы

Проект по разработке веб-сервиса для голосования в реальном времени стал успешным опытом, позволившим не только достичь поставленных целей, но и приобрести новые навыки в области технологий Node.js, React, WebSocket и Docker. Несмотря на возникшие трудности, связанные с настройкой окружения и изучением новых концепций, работая над проектом, я понял важность структурированного подхода к планированию и разработке, а также значимость дизайна и пользовательского опыта. Этот опыт открывает перспективы для дальнейшего развития и совершенствования проекта, а также вдохновляет меня на изучение новых технологий и методов в будущих разработках.

## 9. Используемые источники

1. Видео урок по созданию сайта на React + Nodejs: [https://www.youtube.com/watch?v=H2GckRF9eko&list=PL6DxKON1uLOFJ5\\_dDcX7G1osKnsBICa\\_aT](https://www.youtube.com/watch?v=H2GckRF9eko&list=PL6DxKON1uLOFJ5_dDcX7G1osKnsBICa_aT)
2. Мой лучший интернет друг: <https://chatgpt.com/>
3. Документация WebSoket: <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>
4. Документация Docker: <https://docs.docker.com/>

## 10. Ссылка на репозиторий

<https://github.com/nehoroshiyparen/Voating>