Министерство образования и науки Хабаровского края

краевое государственное бюджетное профессиональное

образовательное учреждение

«Хабаровский технический колледж»

Цикловая комиссия Информационные системы и естественно научные дисциплины

(наименование)

Специальность 09.02.07«Информационные системы и программирование

(код, наименование специальности)

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

«ПП.03 «Проектирование и дизайн информационных систем»

Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ Коломыцкий А. М.

Номер группы дата подпись

Руководитель работы \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ Александрова Л. В.

оценка дата подпись

Хабаровск – 2022 г.

Оглавление

[**Введение** 3](#_Toc119748552)

[**Анализ предметной области** 4](#_Toc119748553)

[**Основные понятия** 5](#_Toc119748554)

[**Аналоги** 5](#_Toc119748555)

# **Введение**

Здесь будет информация о том, зачем это вообще нужно, то

Это за сфера такая и пр.

# 1 Анализ предметной области

В настоящее время спорт и спортивные мероприятия являются предметом внимания множества людей. Начиная от небольших региональных, заканчивая крупными международными турнирами они объединят, сплочают и дарят миллионам людей эмоции. Возможность самому поучаствовать в подобном и почувствовать дух соперничества, а также поработать в команде или в личном первенстве теперь может каждый.

Для структурированной организации турниров и помощи организациям в разработке и налаживании всех процессов и подходит моя предметная область. Данная система полностью удовлетворяет целям организации-заказчика и покрывает весь необходимый функционал для дальнейшего использования.

Данная система будет:

* заниматься организацией мероприятие по проведению турнира
* брать на себя ответственность за поиск места проведения мероприятия
* администрировать и сопровождать процесс проведения мероприятия от его планирования (проектирования) до завершения.
* привлекать органы здравоохранения и безопасности
* предоставлять услуги фото- и видео-съёмки
* предоставлять ведущего.

Система не будет:

* брать на себя ответственность за поведение участников мероприятия
* оплачивать возможные награды и титулы
* сотрудничать и подстраиваться под изменившиеся требования заказчика, если те противоречат законодательству РФ, а также в случае, если компания сочтёт их невозможными или неприемлемыми.

## **1.1 Основные понятия**

Турнир – спортивное соревнование по круговой системе, когда все участники имеют между собой по одной (иногда более) встреч.

Заказчик – лицо (физическое или юридическое), заинтересованное в выполнении исполнителем работ, оказании им услуг или приобретении у продавца какого-либо продукта (в широком смысле).

Организатор – тот, кто организует, устраивает спортивное мероприятие, турнир, сама компания (в широком смысле).

Турнирная сетка – это древовидная диаграмма, представляющая серию игр, сыгранных во время турнира на выбывание.

Приз – это награда, которая вручается человеку или группе людей в знак признания и вознаграждения их действий и достижений.

Участник – лицо, команда, принимающая участие в соревновании, мероприятии, турнире.

## **1.2 Аналоги**

*Компания ИнтелСпорт*.

Интеллектуальное Event-агентство "ИнтелСпорт". Их подход заключается в максимально креативном и эффективном решении бизнес-задач. Компания занимается как организацией небольших турниров, так и крупных мероприятий с количеством участников до 6000 человек.

География мероприятий распространяется не только по всей России, но и за её пределами. Соревнования проводятся также в Турции, Австрии, Испании, Италии, Франции и на Карибских островах!

Основные направления деятельности компании "ИнтелСпорт":

* Спортивные мероприятия
* Интеллектуальные мероприятия
* Корпоративные мероприятия
* Тимбилдинги
* Квесты
* Парусные регаты

Компания является лидирующей на рынке, исходя из поисковых запросов и самопрезентации. На сайте предоставлена вся необходимая информация для пользователя (заказчика). Формы обратной связи и контактная информация также предоставлена и работают. Дизайн сайта выглядит приятно и аккуратно. Внутренне сайт наполнен информацией. Небольшие задержки в подгрузке приложения не мешают, так как для наименее важных блоков используются скелетоны, что является плюсом.

*Компания WORK & SPORT*

С 2014 года организовывает спортивные мероприятия в Петербурге. Крупнейшие в Санкт Петербурге: начинали с небольших турниров, а сейчас проводит масштабные спортивные мероприятия, тимбилдинги и корпоративы для «Ростелекома», «Сбербанка», «Алмаз-Антея» и других больших корпораций.

Цель организации — показать, что спортивные мероприятия могут быть захватывающими и оригинальными. Без советских конкурсов, устаревших сценариев и скучающих зрителей. С вызовами, драйвом и хорошими воспоминаниями.

Оказываемые услуги:

* Однодневные турниры
* Фотографии и видео
* Ведущие и диджеи мероприятия
* Чемпионаты Liga Work & Sport
* Спецпроекты

Сайт компании располагает всей необходимой информацией для пользователя (заказчика). Есть форма заявки, в которой можно оставить свои контактные данные и задать интересующий вопрос, после чего, в ближайшее время с вами свяжутся. Компания достаточно узкоспециализирована и направленна преимущественно на организацию и проведение спортивных мероприятий. Дизайн сайта приятный, все элементы расположены отлично и не мешают знакомится с содержимым.

Проанализировав две компании, я добавил необходимые услуги и возможности в свою систему, а также получил представление о работе и возможностях компаний-аналогов на рынке, исходя из чего буду проектировать свою систему и создавать связи между сущностями.

# 2 Разработка

**2.1 Начало разработки. Конфигурация проекта**

Первым этапом был создан макет сайта, его визуальная составляющая и компоненты пользовательского интерфейса, свойственные проекту, а также отображение портала под разные устройства и поведение интерактивных элементов.

Необходимо определиться с выбором основного способа для создания веб-приложения. Хорошим инструментов является React в связке с языком программирования TypeScript. Данный фреймворк имеет множество расширений, дополнений, хорошо оптимизирован и в купе с хорошей типизацией, сделанной с помощью TypeScript, позволяет создавать строго типизированные, отлаженные приложения. В качестве дополнительных модулей приложения, для улучшения разработки были добавлены: интернализация(i18n), тестирование(jest), управление состоянием и данными в приложении(redux-toolkit), линтинг(проверка) стилей, кода и отлавливание ошибок ещё на этапе разрботки (stylelint, eslint).

После начинается конфигурация проекта и настройка кодовой базы. В качестве сборщика модулей (проекта) я выбрал современный и многофункциональный инструмент - webpack.

В качестве средства хранения данных о пользователе использован firebase[https://firebase.google.com/] – облачное хранилище данных с высокой безопасностью. Местом хранения списка статей, авторов стал имитированный бэкенд сервер, созданный с помощью json-server[https://www.npmjs.com/package/json-server].

Перед начал основного этапа разработки, согласно макету, я вынес цвета, шрифты и другие части пользовательского интерфейса в отдельный файл с переменными.

Листинг глобального файла с переменными:

*:root {*

*--font-family-main: "Montserrat";*

*--font-size-m: 20px;*

*--font-line-m: 35px;*

*--font-m: var(--font-size-m) / var(--font-line-m) var(--font-family-main);*

*--font-size-accent: 18px;*

*--font-line-accent: 35px;*

*--font-accent: var(--font-size-accent) / var(--font-line-accent) var(--font-family-main);*

*--font-size-l: 25px;*

*--font-line-l: 30px;*

*--font-l: var(--font-size-l) / var(--font-line-l) var(--font-family-main);*

*--font-size-x: 44px;*

*--font-line-x: 50px;*

*--font-x: var(--font-size-x) / var(--font-line-x) var(--font-family-main);*

*--font-size-xl: 52px;*

*--font-line-xl: 70px;*

*--font-xl: var(--font-size-xl) / var(--font-line-xl) var(--font-family-main);*

*--font-size-ultra: 80px;*

*--font-line-ultra: 56px;*

*--font-ultra: var(--font-size-ultra) / var(--font-line-ultra) var(--font-family-main);*

*/\* Размеры \*/*

*--navbar-height: 50px;*

*--sidebar-width: 300px;*

*--sidebar-width-collapsed: 100px;*

*/\* Цвета \*/*

*--red-light: #953838;*

*--red-dark: #720000;*

*--white-primary: #fff;*

*--black-primary: #000;*

*--black-secondary: #050505;*

*--footer-bg: #89aabc;*

*--accent-green: #01cd7d;*

*// z-index*

*--modal-index: 10;*

*}*

Первый этап разработки завершён. Далее идёт более подробное описание связей между сущностями, архитектурой приложения и модулями.

**2.2 Основной этап разработки**

В качестве среды разработки я выбрал IDE от JetBrains – Webstorm. Данная программа отлично подходит для написания frontend приложений, и обладает рядом преимуществ, по сравнению с другими средами разработки и текстовыми редакторами. Автоматические импорты, глобальный рефакторинг кода, возможность форматирования кода, автоматическое подхватывание типов данных, встроенная система контроля версий git и множество способ работы с ней, поиск по коду, по файлам и многое другое выделяет эту IDE.

В качестве сборщика проекта выбран webpack, так как этот инструмент обладает очень большим функционалом, имеет полную документацию, множество дополнений, плагинов и пр.

Интернализация при помощи i18n, сейчас это монополист в данной сфере. Бесплатный, настраиваемый инструмент с широкими возможностями в переводе на разные языки и хорошей интеграцией с другими современными инструментами разработки.

Для тестирования отдельных модулей и сущностей выбран фреймвор jest. Идеально подходит для написания как модульных, так и интеграционных тестов. Широкая поддержка, простая и обширная документация, простота использования, лидер рынка в этой области. Его расширение также подходит для тестирования целых react компонентов.

Анализ кода и отлавливание ошибок ещё на этапе разработки осуществляется при помощи инструментов: Eslint (для typescript и tsx кода); Stylelint (для scss кода). Два этих инструмента почти не имею аналогов и широко распространены среди разработчиков. Внедряются при помощи одной команды, гибки в настройке, покрывают все потребности относительно линтинга (Линтинг (от англ. linting) — автоматический поиск программных и стилистических ошибок в коде, оценка его качества) и стиля кода, прекрасно документированы и обладают большой популярностью, и, соответственно, широкой поддержкой в обществе программистов.

Архиважной частью является выбранная методология Feature-sliced design. Архитектура достаточно молодая и по сей день дорабатывается и имеет свои слабые стороны, однако в этом и плюс. Авторы данного паттерна прислушиваются к пользователям и вносят изменения. Любой интересующий вопрос или затруднительный момент можно обсудить в группе разработчиков, которые используют данный подход. Технические преимущества данной архитектурной методологии:

* Явная бизнес-логика. Архитектуру легко осваивать, поскольку она состоит из доменных модулей
* Адаптивность. Компоненты архитектуры можно гибко заменять, добавлять под новые условия
* Технический долг. Каждый модуль можно независимо модифицировать / переписать без сайд-эффектов
* Явная переиспользуемость. Сохраняется баланс между DRY и локальной кастомизацией

В качестве главного инструмента разработки используется React – один из самых популярных инструментов разработки Single page application приложений, имеет много дополнительных модулей, просто интегрируется с другими решениями и инструментами. Явные преимущества:

* Возможность написания JavaScript кода рядом с HTML разметкой
* Компонентный (модульный подход) подход
* Много возможностей для тестирования и оптимизации.
* Широкая поддержка сообщества и гибкость в настройке
* Поддерживается всеми браузерами и не прекращает улучшаться.

Типизация — это способность языка распознавать типы данных. С каждым отдельным типом данных язык программирования может осуществлять определенные действия. Поэтому важно различать разные типы данных и понимать, как с ними может взаимодействовать язык программирования, с которым работаете. Поэтому выбор языка программирования упал на Typescript. В связке с React сводит количество ошибок к минимуму.

Управление состоянием производится при помощи Redux toolkit. Достаточно простая библиотека для React, позволяющая глобально изменять состояния, а добавив расширения и используя все возможности инструмента, и делать запросы, нормализацию и выборку данных. Typescript также отлично дополняет данный контейнер состояния и облегчает процесс разработки.

Для написания стилей страниц и компонентов используется препроцессор для CSS – SCSS. Расширенные возможности языка стилей, вложенность, шаблоны кода, функции, а также простота использования привлекают. Существует достаточно много других препроцессоров, однако данный наиболее популярен и прост.

Процессы авторизации и регистрации производятся при помощи firebase, потому что данный сервис не нуждается в настройке и администрировании, не нужно беспокоиться об конфиденциальности, создавать полноценный сервер и покупать что-либо. Удобный инструмент от известной мировой компании с прекрасной документацией и приятным функционалом, которого хватает для разрабатываемого проекта.

Местом хранения списка статей и заявок является сымитированная база данных

Всё приложение состоит из 6 страниц для клиента и 7 для администратора, соответственно в директории pages будут расположены 7 папок с названием страниц в стиле kebab-case.

Стили для страниц и компонентов пишутся в css-modules подходе, что позволяет обеспечить стопроцентную уникальность и изолированность применяемых стилей. В качестве препроцессора для CSS выбран SCSS.

# **3. Описание разработки**

Feature-Sliced Design – архитектурная методология для фронтенд проектов. Данная концепция лежит в основе проекта и задаёт структура и связь между сущностями в информационной системе.

Первым пунктом в данном этапе является примерное разделение модулей проекта (бизнес-сущности, логика, интерфейс и т.д.) по папкам, согласно методологии. Проанализировав созданный макет сайта создаются папки:

1. app (глобальная логика всего приложения), в которой будет храниться: маршрутизация к страницами, конфигурация, стили;
2. entities (бизнес-сущности, специфичные конкретному бизнесу), здесь размещаются описание пользовательских данных, заказа и типов мероприятий, а также некоторые компоненты пользовательского интерфейса, которые нельзя использовать в другом приложении в силу их специфичности;
3. pages (страница приложения);
4. widgets (композиционный слой, объединяющий в себе сущности entities и действия пользователей features воедино): подвал сайта, навигационная панель(шапка), страница ошибки, переключатель языка;
5. shared (переиспользуемый код, не имеющий отношения к специфике приложения/бизнеса): конфигурация API, тестовой среды, маршрутизации, вспомогательные функции, графические файлы, картинки и набор пользовательского интерфейса.

На глобальном уровне в папку app я также вынес обёртку приложения – контейнер, который будет ограничивать контент по ширине и задавать отступы по бокам, чтобы элементы страницы не примыкали к краям устройств.

Отличительной особенностью главной страницы, страницы услуг и подробного описания статьи является наличие кнопки «Оставить заявку». Данная кнопка открывает пользователю модальное окно, в котором вписываются запрошенные данные, выбирается тип мероприятия и после нажатия на кнопку информация отправляется на сервер.

Модальное окно – UI компонент, расположенный в папке shared, дочерним элементом в данном случае является форма отправки заказа. Также блок формы обёрнут в «DynamicModuleLoader» компонент, который загружает состояние (reducer) только в момент появления страницы/компонента и не занимает память компьютера клиента, улучшает производительность. Вспомогательной функцией является classNames, которая объединяет несколько стилей в один, отлично подходит для написания стилей модулями. В функцию «t» оборачивается текст, который будет переведён на другой язык(интернализация), это встроенная функция в пакет i18n.

Листинг функции classNames:

*export type Mods = Record<string, string | boolean | undefined>;*

*export const classNames = (*

*cls: string,*

*mods: Mods = {},*

*additional: Array<string | undefined> = []*

*): string => {*

*return [*

*cls,*

*...additional.filter(Boolean),*

*...Object.entries(mods)*

*.filter(([\_, value]) => Boolean(value))*

*.map(([className, \_]) => className),*

*].join(' ');*

*};*

Листинг компонента OrderForm*:*

*export interface OrderFormProps {*

*className?: string;*

*}*

*const reducers: ReducerList = {*

*order: orderReducer,*

*};*

*const OrderForm = (props: OrderFormProps) => {*

*const { className } = props;*

*const dispatch = useAppDispatch();*

*const { t } = useTranslation();*

*const name = useSelector(getName);*

*const email = useSelector(getEmail);*

*const phone = useSelector(getPhone);*

*const comment = useSelector(getComment);*

*const eventType = useSelector(getEventType);*

*const onChangeName = useCallback(*

*(value: string) => {*

*dispatch(orderActions.setName(value));*

*},*

*[dispatch]*

*);*

*const onChangePhone = useCallback(*

*(value: string) => {*

*dispatch(orderActions.setPhone(value));*

*},*

*[dispatch]*

*);*

*const onChangeEmail = useCallback(*

*(value: string) => {*

*dispatch(orderActions.setEmail(value));*

*},*

*[dispatch]*

*);*

*const onChangeComment = useCallback(*

*(value: string) => {*

*dispatch(orderActions.setComment(value));*

*},*

*[dispatch]*

*);*

*const onChangeType = useCallback(*

*(value: TypeEvents) => {*

*dispatch(orderActions.setType(value));*

*},*

*[dispatch]*

*);*

*return (*

*<DynamicModuleLoader reducers={reducers}>*

*<form className={classNames(cls.OrderForm, {}, [className])}>*

*<Input*

*type='text'*

*value={name}*

*className={cls.inp}*

*onChange={onChangeName}*

*placeholder={t('Имя')}*

*/>*

*<Input*

*type='text'*

*value={phone}*

*className={cls.inp}*

*onChange={onChangePhone}*

*placeholder={t('Номер телефона')}*

*/>*

*<Input*

*type='text'*

*value={email}*

*className={cls.inp}*

*onChange={onChangeEmail}*

*placeholder={t('Электронная почта')}*

*/>*

*<EventsSelect*

*value={eventType}*

*className={cls.select}*

*onChange={onChangeType}*

*/>*

*<TextArea*

*value={comment}*

*className={cls.textarea}*

*onChange={onChangeComment}*

*placeholder={t('Комментарий')}*

*/>*

*<SendOrder>{t('Оставить заявку')}</SendOrder>*

*</form>*

*</DynamicModuleLoader>*

*);*

*};*

*export default memo(OrderForm);*

Кнопка «Отправить заявку» - это явная feature, хранится в соответствующей папке. Основная логика по отправке данных о заказе на сервер осуществляется с помощью функции onSendOrder.

*Листинг функции onSendOrder:*

*…*

На сайте используется процессы авторизации и регистрации, данные сущности вынесены в отдельные features, согласно методологии. Их основное назначение – проводить валидацию данных и отправлять запрос на firebase и возвращать статус выполнения операции.

В шапке сайта расположена кнопка «Войти», при нажатии на которую открывается модальное окно с формой об авторизации, в которую вводятся данные для идентификации пользователя. При успешной авторизации вместо кнопки отображается имя клиента.

Вёрстка других страниц и компонентов достаточно проста и не имеет сложной логики и состоит из стандартных тегов HTML5. В соответствующей папке pages располагаются страницы: регистрации, о компании, главная, новостей, услуг, заявок, контактов.

Для отлавливания непредвиденных ошибок всё приложение я обернул в компонент ErrorBoundary[https://reactjs.org/docs/error-boundaries.html].