Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки – 09.03.01 «Информатика и вычислительная техника»

Кафедра вычислительной техники

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка web-приложения для управления проектами с применением технологий REST и SPA

УДК 004.738.5.001.63

Студент

Группа	ФИО	Подпись	Дата
8В2А	Бенц Андрей Сергеевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Хаустов П.А.	-		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Николаенко Валентин Сергеевич	-		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Невский Егор Сергеевич	-		

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ВТ	Марков Н.Г.	д.т.н. профессор		

Томск – 2016 г.

ЗАПЛАНИРОВАННЫЕ РЕЗУЛЬТАТЫ ПО ОСНОВНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЕ ПОДГОТОВКИ БАКАЛАВРОВ 09.03.01 «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА», ИК ТПУ, ПРОФИЛЬ «ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, КОМПЛЕКСЫ, СИСТЕМЫ И СЕТИ»

Код результатов	Результат обучения (выпускник должен быть готов)
Профессиональные компетенции	
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.
Универсальные компетенции	
Р7	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.
Р8	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.
Р9	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.
Р10	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.
Р11	Демонстрировать способность к самостоятельной к самостоятельному обучению в течение всей жизни и непрерывному самосовершенствованию в инженерной профессии.

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики
Направление подготовки – 09.03.01 «Информатика и вычислительная техника»
Кафедра Вычислительной техники

УТВЕРЖДАЮ:
Зав. кафедрой
_____ _____ Марков Н.Г.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студенту:

Группа	**ФИО**
8В2А	Бенц Андрею Сергеевичу

Тема работы:

Разработка web-приложения для управления проектами с применением технологий REST и SPA	
Утверждена приказом директора (дата, номер)	От 19.02.2016 №1319/с

Срок сдачи студентом выполненной работы:	10.06.2016

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Исходными данными являются следующие требования: - Система должна быть многопользовательской с возможностью одновременной работы нескольких пользователей; - Проект должен состоять из заданий с ограниченным временем выполнения, исполнителем и ответственным за исполнение; - Стабильность и безотказность работы. Средства разработки должны быть выбраны исходя из требований к системе.

Перечень подлежащих исследованию, проектированию и разработке вопросов	Обзор функционала аналогичных систем, выбор средств реализации и тестирования разработанного приложения, проектирование архитектуры приложения, разработка элементов архитектуры.
Перечень графического материала	Схема архитектуры, используемой в приложении. Диаграммы вариантов использования. Скриншоты интерфейса разработанной системы.

Консультанты по разделам выпускной квалификационной работы

Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Николаенко Валентин Сергеевич
Социальная ответственность	Невский Егор Сергеевич

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	10.09.2015

Задание выдал руководитель:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Ассистент кафедры ВТ	Хаустов Павел Александрович	-		10.09.2015

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8В2А	Бенц Андрей Сергеевич		10.09.2015

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт Кибернетики
Направление подготовки – 09.03.01 «Информатика и вычислительная техника»
Уровень образования – Бакалавриат
Кафедра Вычислительной техники
Период выполнения осенний / весенний семестр 2015/2016 учебного года

Форма представления работы:

Бакалаврская работа

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	10.06.2016

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
2.09.2015	Составление и утверждение ТЗ	5
07.10.2015	Изучение материала по системам управления проектами	5
04.11.2015	Выбор средств реализации приложения	5
02.12.2015	Разработка архитектуры приложения	20
03.02.2016	Реализация базового функционала системы	20
06.04.2016	Тестирование, отладка и доработка системы	20
04.05.2016	Завершение работы над системой	15
20.05.2016	Социальная ответственность	5
27.05.2016	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	5

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент кафедры ВТ	Хаустов Павел Александрович	-		10.09.2015

СОГЛАСОВАНО:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
Вычислительной техники	Марков Николай Григорьевич	д.т.н., профессор		10.09.2015

РЕФЕРАТ

Выпускная квалификационная работа содержит 103 страницу, 17 рисунков, 4 таблицы, 18 источников.

Ключевые слова: система управления проектами, web-приложение, REST, SPA, одностраничное приложение, ReactJS.

Объектом исследования является система управления проектами.

Цель работы – разработка web-ориентированной системы управления проектами, использующее технологию SPA на клиентской части приложения, и RESTful сервис на серверной части.

В процессе выполнения работы были выполнены анализ и исследование существующих аналогов. Были произведен выбор средств, используемых при разработке, тестировании и отладке приложения.

В результате работы была разработана и протестирована web-ориентированная система управления проектами, использующая технологии REST и SPA.

Область применения – организация рабочего процесса в небольших компаниях или для персонального использования.

Значимость работы заключается в использовании современных web-технологий, позволяющих повысить производительность как серверной части, так и клиентской части приложения, а также предоставить пользователям опыт использования, схожий с предоставляемым прикладными приложениями.

В будущем планируется увеличение и расширение функционала приложения.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И НОРМАТИВНЫЕ ССЫЛКИ

Система управления проектами – это программный продукт, позволяющий осуществлять контроль за процессом выполнения как проекта в целом, так и отдельных его частей, получать систематизированную информацию о его составных частях, назначать ответственных на выполнение.

Токен – некоторый объект, который представляет право на осуществление некоторых операций.

SPA – одностраничные приложения (single page application) это web-приложения, которые, фактически, размещаются в одном html-документе, и осуществляют взаимодействие с пользователем путем динамической загрузки HTML, JavaScript и CSS, необходимых для работы, с целью обеспечения пользователем опытом использования приложения схожим с настольными приложениями.

JSON (JavaScript Object Notation) – форма представления данных, в который сами данные представляются в виде объекта языка JavaScript, вида «ключ»:«значение».

JWT (JSON Web Tokens) – открытый стандарт (RFC 7519) для передачи так называемых «запросов» (англ. claims) между двумя сторонами, взаимодействующими с помощью веб-приложений.

REST (Representational State Transfer) – стиль архитектуры программного обеспечения, при котором вся информация о запросе хранится в самом запросе, отсутствует хранение сервером состояний.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Управление проектом является одной из важных его частей, и во многом определяет, будет ли проект успешен и будет ли работа над ним завершена. Это применимо как к большим командам и организациям, так и к проектам, над которыми работает один человек. Как следствие, присутствует необходимость наличия средств для удобной, оперативной и надежной работы над проектом.

Множество систем управления проектами по умолчанию ограничивают функционал своих приложений в бесплатной ее версии, и предполагают ее направленность на использование в личных целях, либо как пробную версию перед покупкой полной.

Разработанная в ходе работы система предполагает бесплатную модель распространения. Кроме того, она предусматривает возможность установки на сервер конечного пользователя (hosted-on-premises), что может обеспечить доступность при отключении доступа к сети Интернет, и при этом отсутствует необходимость хранения конфиденциальных данных на серверах сторонних организаций.

Целью работы является разработка web-приложения для управления проектами с применением технологий REST и SPA.

Объектом исследования в работе являются проекты и системы управления проектами, предмет исследования – проектирование и разработка системы управления проектами.

Практическая новизна работы проявляется в использовании при разработке современных web-технологий, что позволит достигнуть высокой производительности и удобства работы пользователя с системой.

В работе реализуется клиентская часть приложения.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Особенности существующих систем

1.1.1 Классификация по назначению

По назначению системы управления проектами делятся на:

- Персональные – такие системы обычно используются отдельными пользователями на одном устройстве, например, для управления личными проектами;
- Однопользовательские – такие системы разрабатываются исходя из того, что план проекта будет редактироваться только одним пользователем. Такие системы могут использоваться в небольших компаниях, либо в компаниях, в которых планированием проектов занимается небольшое количество участников. Зачастую – приложения для ПК;
- Многопользовательские – такие системы разрабатываются для поддержки одновременной работы над проектом нескольких пользователей. Так, некоторые пользователи имеют возможность изменять те участки проекта, за выполнение которых они ответственны. В свою очередь, другие участники могут быть ответственны за проверку результатов выполнения заданий подчиненных им участников. Большинство web-ориентированных систем относятся к этой категории.

1.1.2 Классификация по платформе назначения

По платформе назначения системы управления проектами можно разделить на две категории: прикладные и web-ориентированные.

Прикладные приложения представляют собой приложения, создаваемые под конкретные операционные системы, реализованные (обычно) на компилируемых языках программирования. Требуют установки на устройства пользователей.

Плюсом таких приложений является их скорость работы, а также расширенный функционал по сравнению с web-ориентированными, т.к. они не

ограничены браузером (что, однако, не всегда является необходимым). Минусами таких приложений являются необходимость дополнительных средств при разработке, если целью является поддержка нескольких платформ, а также необходимость пользователями загрузки установочных пакетов.

Web-ориентированные приложения основываются на работе в браузере, с расчетом на то, что, на сегодняшний день, браузер установлен на большинстве устройств. Они представляют собой приложения, написанные на JavaScript, использующие HTML и CSS для отображения информации. Производители браузеров берут на себя задачу портирования его на различные платформы, что позволяет разработчикам сосредоточиться только разработке приложения.

Однако такие приложения будут иметь меньшую производительность по сравнению с прикладными, а также ограничиваться функционалом, предоставляемым им браузером.

1.1.3 Функциональные возможности аналогичных систем

Как правило, аналогичные системы обладают следующими функциональными возможностями:

- Совместная работа – функционал, позволяющий нескольким пользователям создавать и обмениваться различными задачами и файлами, что позволяет двум и более удаленным пользователям осуществлять совместную работу над одними задачами или проектами;
- Отслеживания ошибок (issue tracking system, bug tracking system) – функции, позволяющие отслеживать список ошибок и проблем, возникших при реализации проекта. Обычно, необходимость в таком функционале возникает у организаций, которые занимаются разработкой ПО, т.к. такие организации ведут учет ошибок, найденных в реализуемых модулях исходного кода, а также учет различных пожеланий пользователей;

- Управления документооборотом – функционал, используемый для хранения в системе электронных документов, а также их возможное редактирование;
- Управление рабочими процессами – функционал, позволяющий создавать и отслеживать некоторую последовательность задач, представленных в виде диаграммы рабочих процессов;
- Планирование – организация выполнения работ в рамках проекта, которое включает в себя такие процессы, как выстраивание выполняемых задач по времени, определение сроков выполнения задач, определение исполнителей и используемых ресурсов.

1.2 Примеры существующих систем

1.2.1 Microsoft Project

Одна из известных прикладных систем управления проектами – Microsoft Project. Она принадлежит семейству Microsoft Office, является проприетарным ПО, также возможен доступ по месячной подписке [4].

Microsoft Project обладает следующими возможностями:

- Microsoft Project оперирует так называемыми ресурсами, обладающими стоимостью. В свою очередь ресурсы могут представлять собой людей, оборудование, материалы и т.п. Присутствует возможность распределять ресурсы по отдельным заданиям. Далее, вычисляется стоимость использования ресурсов для выполнения задания, групп заданий и всего проекта. При необходимости, ресурсы могут быть распределены между несколькими проектами;
- Для каждого ресурса в системе располагается расписание, содержащее сведения, необходимые для планирования его использования. Это осуществляется исходя из того, в какие дни ресурс свободен;
- Возможность нахождения критического пути для проекта;
- Производится разделение пользователей системы по различным уровням доступа;
- Присутствует функционал, позволяющий отображать состояние проекта средством различных графиков.

1.2.2 Wrike

Wrike – одна из популярных web-ориентированных систем управления проектами. Распространяется по условно бесплатной модели, в которой бесплатная версия предоставляет доступ ограниченному количеству участников в проект (до 5), а платная версия снимает ограничения и распространяется по годовой подписке [3].

Основные возможности приложения:

- Лента новостей, показывающая последние события в проекте, позволяющая в удобном виде просматривать сводную информацию о произошедших событиях в проекте;
- Поддержка мобильных платформ;
- Поддержка e-mail уведомлений и push-уведомлений как в браузере, так и на мобильных устройствах;
- Построение диаграммы Гантта, создание визуальных отчетов о состоянии проекта.

Общий вид программы Microsoft Project представлен на рисунке 1.2.

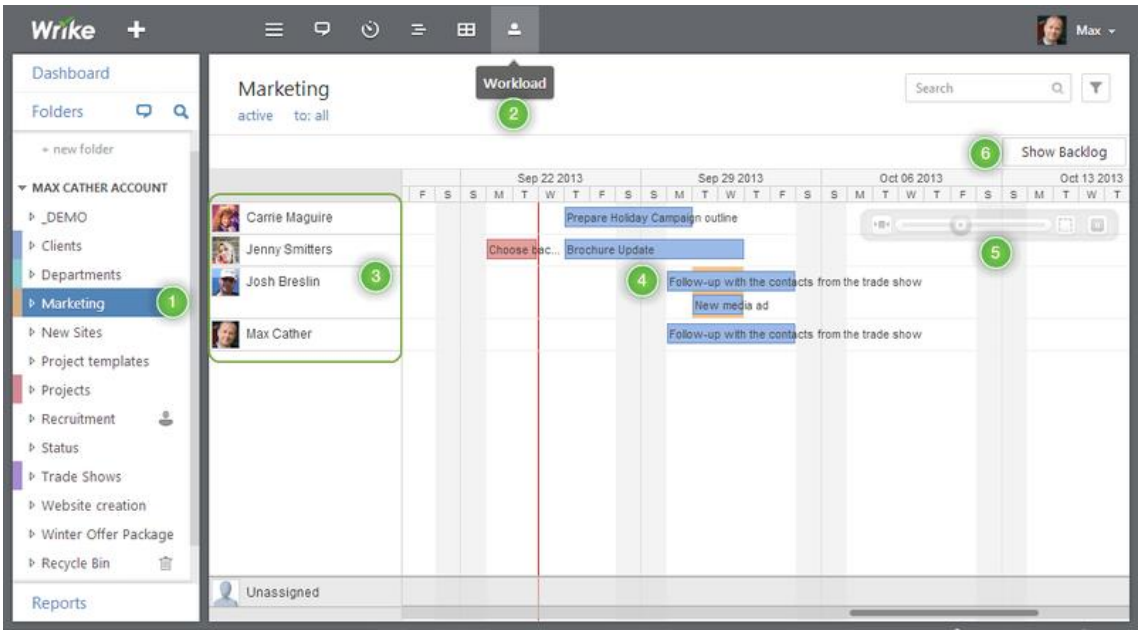

Рисунок 1.1. Общий вид приложения Wrike

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 Описание разрабатываемой системы

Разрабатываемая в данной работе система является многопользовательской web-ориентированной. Реализация в виде web-приложения позволит легко добиться кроссплатформенности и мобильности, т.к. на сегодняшний день браузер предустановлен в большинстве операционных систем, либо имеется возможность дополнительной загрузки необходимого браузера. К тому же, пользователь имеет возможность получать доступ к приложению с различных устройств после регистрации аккаунта в системе без необходимости установки приложения на все устройства

Система состоит из клиентской и серверной части.

Клиентская часть представляет собой одностраничное web-приложение и отвечает за предоставление конечному пользователю запрашиваемого контента. Серверная часть приложения обеспечивает хранение и обработку данных о пользователях и проектах, и предоставление этих данных пользователю.

2.2 Архитектура представлений

В ходе проектирования пользовательского интерфейса были выделены следующие разделы приложения, присутствие которых необходимо для удобства работы пользователя с системой:

- Обзор – раздел, в котором можно посмотреть сводную информацию о проекте и список последних его событий;
- Задания – раздел, в котором отображаются задания проекта, в которых пользователь является либо исполнителем, либо руководителем;
- Календарь – раздел, в котором пользователи могут нагляднее просматривать информацию о сроках выполнения в виде календаря;
- Участники – раздел, в котором приводится список пользователей, состоящих в проекте;

- Настройки – раздел, в котором имеется возможность просмотра более детальной информации о проекте, такой как его описание, сроки, менеджер и т.п. с возможностью их изменения.

2.3 Диаграммы вариантов использования

Исходя из разработанной архитектуры представлений, были спроектированы варианты использования для каждого из них.

Диаграмма вариантов использования раздела «обзор» представлена на рисунке 2.1.

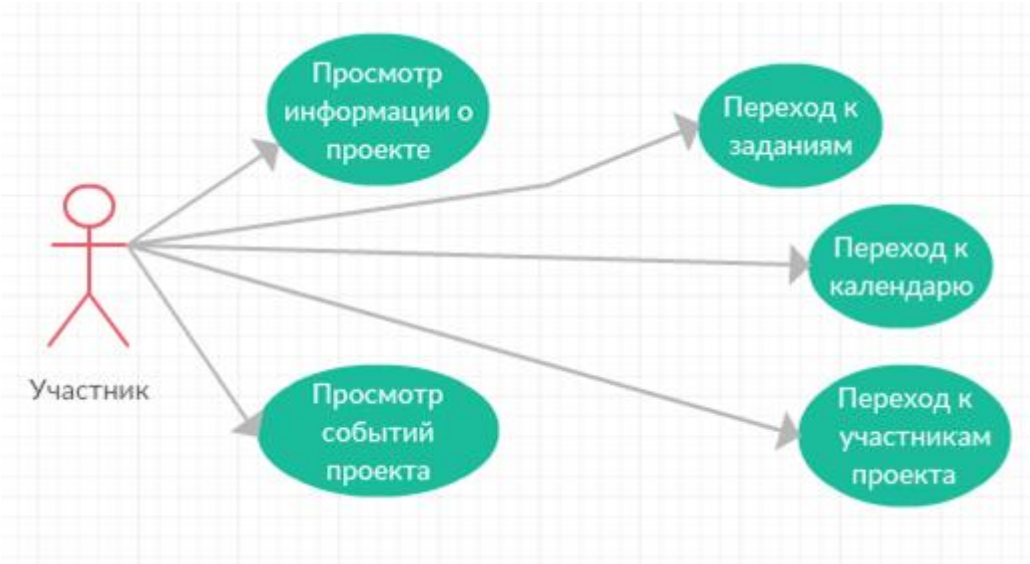

Рисунок 2.1. Диаграмма вариантов использования раздела «обзор»

Любой участник проекта имеет возможность просмотра информации о проекте, такой как срок выполнения, количество заданий, количество участников, а также последние события, произошедшие в проекте. Этим возможностям соответствуют варианты использования «просмотр информации о проекте» и «просмотр событий проекта».

Также предусматривается наличие быстрых ссылок для перехода в разделы «задания», «календарь» и «участники проекта», т.к. предполагается, что именно эти разделы являются наиболее посещаемые при работе с системой. Этому функционалу соответствуют варианты использования «переход к заданиям», «переход к календарю» и «переход к участникам».

Диаграмма вариантов использования раздела «задания» представлена на рисунке 2.2.

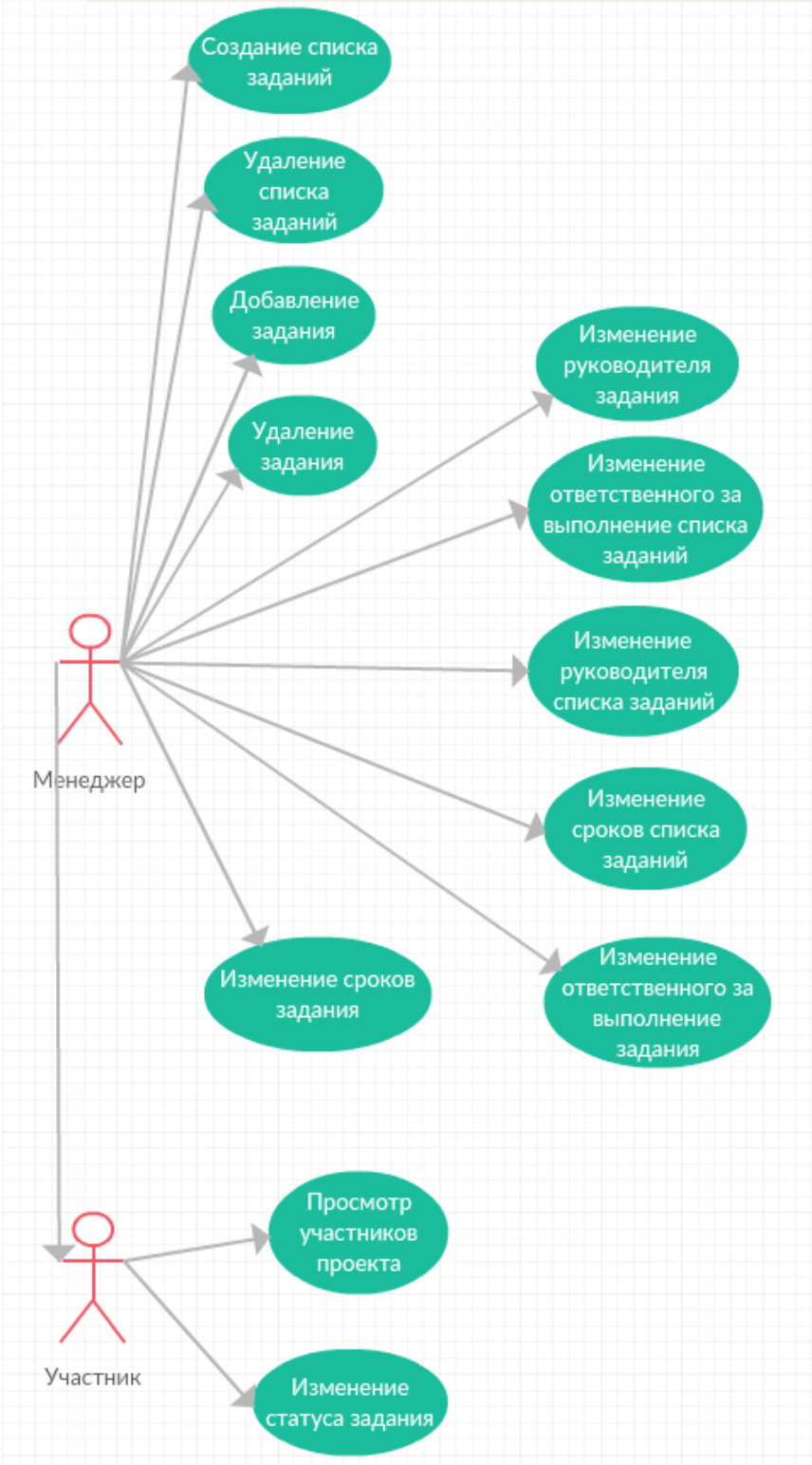

Рисунок 2.2. Диаграмма вариантов использования раздела «задания»

Любой участник проекта имеет возможность просмотра заданий, за выполнение которых он ответственен, и изменения их статуса, например, когда работник выполнил задачу, он изменяет его состояние и тем самым отправляет

его на проверку. На диаграмме (рисунок 2.2) этот функционал соответствует вариантам использования «просмотр заданий проекта» и «изменение статуса задания».

В проекте также присутствуют менеджеры, которые так же, как и обычные участники, обладают возможностями, описанными выше. Дополнительно, менеджеры создают задания для выполнения, назначают для них исполнителей, руководителей и сроки и могут изменять их в течении работы над проектом. Задания могут быть сгруппированы в списки, которые также могут иметь руководителей. Также, задания и списки заданий могут быть удалены. Эти функции соответствуют вариантам использования «создание списка заданий», «удаление списка заданий», «добавление задания», «удаление задания», «изменение руководителя задания», «изменение ответственного за выполнение задания», «изменение сроков задания», «изменение ответственного за выполнение списка заданий», «изменение руководителя списка заданий».

Диаграмма вариантов использования раздела «календарь» представлена на рисунке 2.3.

Рисунок 2.3. Диаграмма вариантов использования раздела «календарь»

Участник проекта может просматривать задания проекта в виде календаря, что позволяет в удобном виде оценивать их сроки. Присутствует возможность просмотра заданий, как текущего месяца, так и предшествующих и последующих месяцев. Описанный функционал соответствует вариантам

использования «просмотр заданий текущего месяца» и «изменение периода просмотра». Кроме того, менеджер проекта имеет возможность изменения сроков заданий.

Диаграмма вариантов использования раздела «участники» представлена на рисунке 2.4.

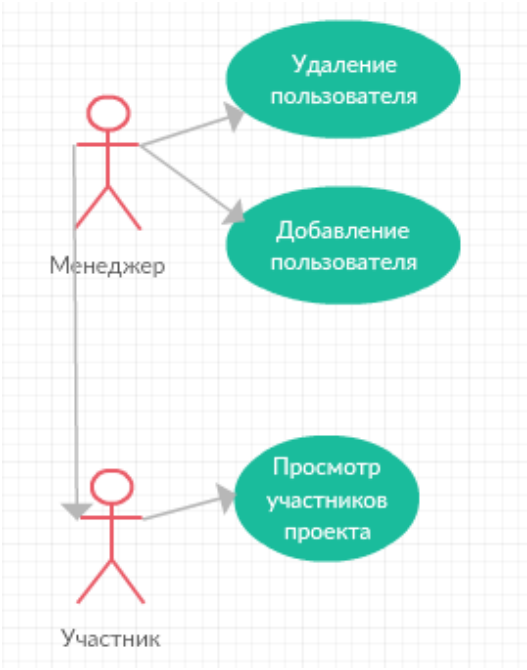

Рисунок 2.4. Диаграмма вариантов использования раздела «участники»

Участник проекта имеет возможность просматривать таблицу, отображающую информацию об участниках проекта. Это выражается в варианте использования «просмотр участников проекта». Менеджер проекта, в свою очередь, имеет возможность добавлять и удалять пользователей.

Диаграмма вариантов использования раздела «настройки» представлена на рисунке 2.5.

Рисунок 2.5. Диаграмма вариантов использования раздела «настройки»

В разделе «настройки» участник может посмотреть более детальную информацию о проекте, что соответствует варианту использования «просмотр информации о проекте». Менеджер проекта имеет возможность эту информацию изменить, а также сменить менеджера проекта и удалить проект.

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Реализация логики приложения осуществлялась на языке JavaScript стандарта ECMAScript 6. Реализация пользовательского интерфейса осуществлялась на языке JSX. Поддерживаются любые настольные операционные системы, на которые могут быть установлены браузеры Google Chrome, Mozilla Firefox, Opera и Internet Explorer 11. Также поддерживаются мобильные устройства под управлением iOS 7.0 и выше и Android 4.0 и выше.

3.1 Используемые при разработке инструменты и технологии

3.1.1 Фреймворк для реализации SPA

При реализации приложения используется технология SPA (single page application, одностраничное приложение). В этой технологии приложение фактически располагается в одном HTML документе, и навигация по приложению осуществляется динамическими манипуляциями с его содержимым средством JavaScript. На сегодняшний день существует несколько популярных средств, на основе которых можно реализовать подобных функционал. Такие средства: ReactJs + Flux, AngularJs, EmberJs. Последние два представляют собой полноценные фреймворки, которые содержат в себе большинство средств, необходимых для различных сценариев использования. ReactJs представляет собой библиотеку для построения пользовательских интерфейсов, а Flux, в свою очередь, архитектуру приложений, которая создана как дополнение к React. В ходе работы с React + Flux могут возникнуть потребности в средствах, которые не были в них заложены, однако подобные библиотеки на сегодняшний день обладают огромным сообществом, с помощью которого можно найти необходимые средства.

Далее представлены особенности данных средств разработки.

Angular:

- Возможность создания собственных DOM-элементов;
- Наличие двухсторонней и односторонней связи компонентов;

- Архитектура приложения состоит из Представлений (Views), Контроллеров (Controllers), Служб (Services), директив (Directives);
- Неравномерная кривая обучения;
- На текущий момент происходит переход с первой версии на вторую, которая находится в статусе бета;
- Использование языка TypeScript;
- Размер файла равен 622 КБ (включая только основной функционал) [6].

Ember:

- Низкий уровень повторяемости кода;
- Упор на производительность;
- Работа с шаблонизаторами;
- Рендеринг на стороне сервера;
- Двунаправленное связывание;
- Неравномерная кривая обучения;
- Размер файла равен 451 КБ [5].

React + Flux:

- Наличие огромного сообщества;
- Архитектура, полностью построенная на компонентах;
- Односторонний поток данных;
- Использование виртуальной DOM-модели для увеличения производительности;
- Использование языка JSX;
- Размер файла равен 144 КБ [1].

Для реализации разрабатываемого приложения был выбран React + Flux. Наличие огромного сообщества, отличной документации и прямой кривой обучения позволяет сосредоточиться на реализации самого приложения при использовании данных средств. Angular не был выбран из-за его нахождения в статусе beta и необходимости использования языка TypeScript, как основного,

изучение которого требует дополнительных временных затрат. Ember не был выбран ввиду его большого размера и нелинейной кривой обучения.

Для реализации маршрутизации на стороне клиента используется библиотека React Router, позволяющая в декларативном стиле определить возможные маршруты приложения.

3.1.2 Сборщик модулей

При разработке приложения использовался сборщик модулей (англ. module bundler) Webpack. Разрабатываемое приложение состоит из различных модулей, содержащих код на JavaScript, а также различные модули CSS стилей. Текущие браузеры не имеют поддержки модульности, а подключение каждого файла по отдельности не имеет смысла, т.к. в общем случае загрузка необходимых ресурсов одним HTTP-запросом происходит в несколько раз быстрее, чем загрузка посредством нескольких отдельных запросов. Поэтому возникает необходимость генерации статических ресурсов приложения.

К тому же, разработка приложения осуществляется с использованием стандарта ES6 языка JavaScript, который был выпущен в 2015 году. Этот стандарт значительно расширяет синтаксис языка и ускоряет разработку, однако на текущий момент не имеет полной поддержки среди браузеров, следовательно, необходима трансляция исходного кода в стандарт ES5, поддерживаемый большинством браузеров.

Webpack предоставляет все перечисленные ранее средства. По умолчанию webpack обеспечивает только генерацию единого модуля приложения, однако он обладает поддержкой расширений [7]. Все необходимые расширения для данного проекта уже были разработаны сообществом к моменту начала разработки.

В работе используются следующие библиотеки вместе с их расширениями для Webpack:

- Babel и Babel-loader для трансляции кода из стандарта ES6 в стандарт ES5 [6];

- Babel-preset-react для трансляции кода на языке JSX, используемом при описании пользовательских интерфейсов средствами ReactJS в код на JavaScript;
- Style-loader и Css-loader для загрузки модулей с CSS-стилями [9, 10].

3.1.3 CSS фреймворк

Клиентом данного приложения является браузер. Благодаря наличию браузеров для практически любой платформы, имеется возможность реализовать поддержку как ПК, так и мобильных устройств. В качестве CSS фреймворка был выбран Bootstrap 3. Bootstrap 3 – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов [11].

Основные преимущества Bootstrap 3:

- Экономия времени – Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы, и сконцентрироваться на других разработках;
- Высокая скорость – динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке;
- Простота в использовании;
- Совместимость с браузерами – Bootstrap совместим с Mozilla Firefox, Google Chrome, Safari, Internet Explorer и Opera;
- Открытое программное обеспечение – особенность Bootstrap, которая предполагает удобство использования посредством открытости исходных кодов и бесплатной загрузки.

Bootstrap предоставляет средства для создания адаптивного дизайна (англ. responsive design), который, в свою очередь, позволяет оптимизировать приложение под различные варианты ширины экрана устройства. Благодаря

этому одна и та же HTML-верстка будет отображаться по-разному в зависимости от устройства, с которого осуществляется просмотр.

3.2 Архитектура Flux

Flux – архитектура клиентской части web-приложений, основной упор в которой сделан на однонаправленный поток данных. Эта архитектура содержит три основных компонента – Dispatcher, Store, View, и один дополнительный – Action [2]. Смысл этих компонентов в следующем:

- Actions (Действия) – некоторые события, возникшие в результате взаимодействия пользователя с приложением. Содержат в себе некоторый набор данных (payload), который несет в себе информацию о самом действии, которое произошло;
- Dispatcher (Диспетчер) – некоторый центральный обработчик действий. Действия, возникшие в ходе работы системы вместе с данными об этом действии отправляются в диспетчер. Далее диспетчер пересылает эти данные различным хранилищам (Store), которые подписаны на это действие;
- Stores (хранилища) – контейнеры, которые хранят различные части состояния всего приложения и содержат в себе бизнес-логику приложения;
- Views (представления) – различные компоненты системы, которые реализуют пользовательский интерфейс приложения. Они используют данные из хранилищ для отображения актуальной версии пользовательского интерфейса приложения, и реагируют на их изменения.

Описанный процесс можно изобразить в виде следующей диаграммы (рисунок 3.1):

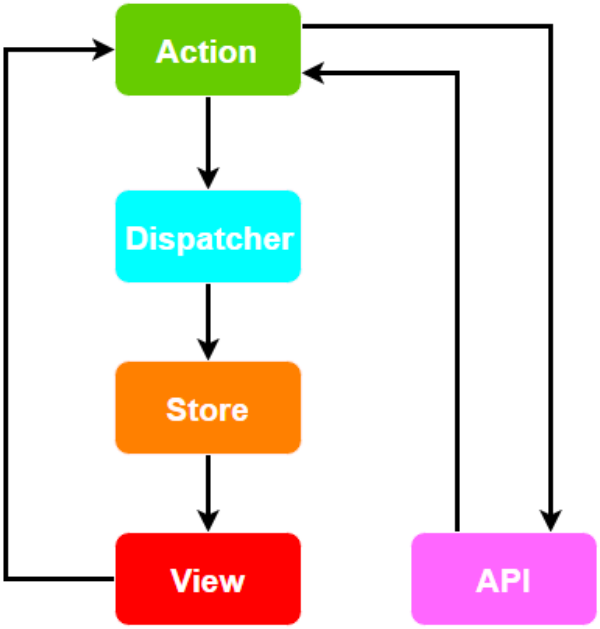

Рисунок 3.1. Диаграмма потока данных в архитектуре Flux

Далее приведено более подробное описание компонентов архитектуры Flux.

3.2.1 Dispatcher (Диспетчер)

По своей сути диспетчер является некоторой системой событий. Он регистрирует в себе различные события, которые рассылают действия, и подписывает Хранилища на изменения этих действий. При этом важно, чтобы Диспетчер был один на всё приложение [2]. Пример того, как это может выглядеть в программной реализации (на JavaScript):

```
var Dispatcher = require('flux').Dispatcher;
var AppDispatcher = new Dispatcher();
AppDispatcher.handleAction = function(action) {
     this.dispatch({
          type: "ACTION",
          action: action
     });
}
```

В примере выше создается Диспетчер и далее при возникновении Действия Диспетчер рассылает всем Хранилищам, подписанным на это Действие, информацию о событии функцией dispatch. Описанный процесс изображен на следующей диаграмме (рисунок 3.2):

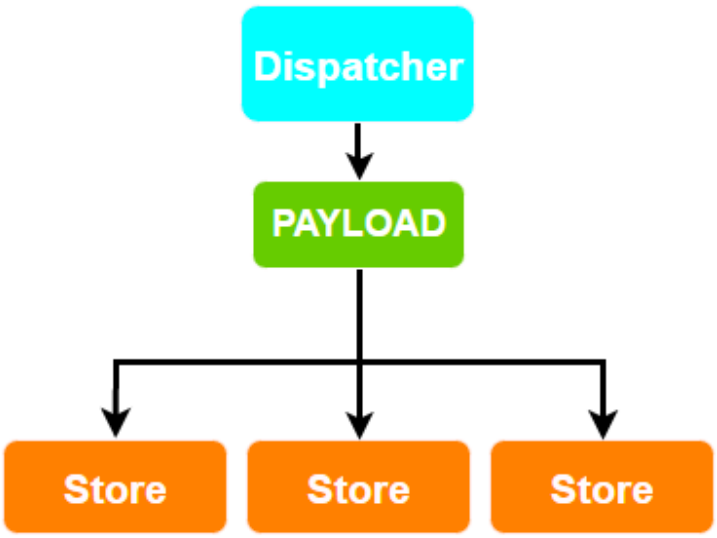

Рисунок 3.2. Диаграмма работы диспетчера

3.2.2 Stores (Хранилища)

Хранилище в архитектуре Flux хранит в себе данные определенной части приложения, методы получения и обработки этих данных, а также обработчики действий, зарегистрированные в Диспетчере [2]. Пример реализации представлен ниже.

```
var AppDispatcher = require('../dispatcher/AppDispatcher');
var ShoeConstants = require('../constants/ShoeConstants');
var EventEmitter = require('events').EventEmitter;
var merge = require('react/lib/merge');

// Внутренний объект для хранения shoes
var _shoes = {};

// Метод для загрузки shoes из данных Действия
function loadShoes(data) {
  _shoes = data.shoes;
}

// Добавить возможности Event Emitter из Node
var ShoeStore = merge(EventEmitter.prototype, {

  // Вернуть все shoes
  getShoes: function() {
    return _shoes;
  },

  emitChange: function() {
    this.emit('change');
  },
```

```
  addChangeListener: function(callback) {
    this.on('change', callback);
  },

  removeChangeListener: function(callback) {
    this.removeListener('change', callback);
  }

});

// Зарегистрировать обработчик в Диспетчере
AppDispatcher.register(function(payload) {
  var action = payload.action;
  var text;
  // Обработать Действие в зависимости от его типа
  switch(action.actionType) {
    case ShoeConstants.LOAD_SHOES:
      //  Вызвать  внутренний  метод  на  основании  полученного
Действия
      loadShoes(action.data);
      break;

    default:
      return true;
  }

  // Если Действие было обработано, создать событие "change"
  ShoeStore.emitChange();

  return true;

});

     module.exports = ShoeStore;
```

Как упоминалось выше, хранилища слушают и рассылают различные события, например, чтобы оповестить о своем обновлении. В примере выше это достигается наследованием хранилища от EventEmitter – класса системы NodeJs, позволяющего работать с событиями. Для этой задачи в целом может подойти любая библиотека для работы с событиями.

Также в коде выше можно увидеть ранее описанную схему взаимодействия хранилища с Диспетчером. Хранилище подписывается на оповещения от Диспетчера с помощью метода register, которому указывается метод, который будет вызываться после прихода оповещения. Этот метод определяет, будет ли произошедшее Действие обрабатываться в этом

Хранилище. Если да, то после обработки вызывается метод onchange данного хранилища, чтобы оповестить Представления об обновлении.

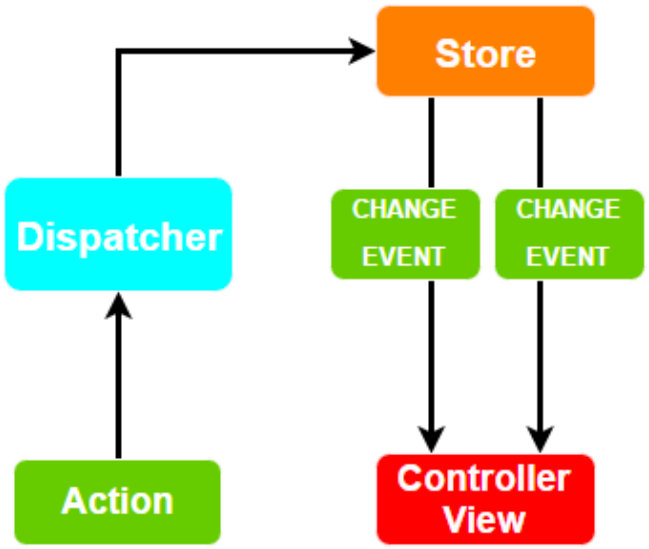

Рисунок 3.3. Диаграмма взаимодействия с хранилищами

3.2.3 Views (Представления)

Представлением в архитектуре Flux может служить практически все что угодно, что может создавать DOM-элементы на странице. Представление должно реагировать на события Хранилища и изменяться соответственно с новым состоянием Хранилища [2]. Одной из библиотек, представляющих необходимый функционал является ReactJs.

ReactJs – JavaScript библиотека с открытым исходным кодом для создания пользовательского интерфейса.

В основе React лежит понятие компонента как некоторой составляющей интерфейса, которая может содержать в себе вложенные компоненты. Например, компонент «Навигация», который может содержать в себе компонент «Список ссылок» и т.п. Таким образом весь пользовательский интерфейс разбивается на компоненты с некоторым корневым компонентом. При этом компоненты имеют Состояние и правило рендеринга самого компонента на HTML страницу [1].

ReactJs, так же, как и Flux, придерживается однонаправленному потоку данных. Также в нем используется концепция Virtual DOM, которая позволяет

определять изменения в компоненте и затрачивать наименьшее количество операций для приведения фактического DOM HTML-документа в соответствие.

При взаимодействии пользователя с пользовательским интерфейсом состояние компонентов может измениться, следовательно, необходимо произвести ре-рендеринг, чтобы фактический DOM страницы соответствовал состоянию компонентов. Проблема DOM в том, что он изначально не был рассчитан на большое количество манипуляций и данные манипуляции имеют не очень большое быстродействие.

Для быстрого нахождения разницы DOM и состояния компонентов используется концепция Virtual DOM. React хранит в памяти легковесную копию реального DOM. Далее при изменении состояния компонентов сначала происходит их рендеринг в Virtual DOM, обладающего большей производительностью. После чего происходит сравнение виртуального и реального DOM и определяется минимальный набор изменений, необходимых для актуализации реального DOM. Таким образом количество DOM-манипуляций сводится к минимуму.

При разработке с помощью ReactJs используется язык JSX – JavaScript with Extensions. Этот язык позволяет использовать в коде на JavaScript конструкции языков разметки, например, XML. Это используется для удобного представления HTML-элементов прямо в коде.

3.3 Реализация компонентов Flux

В работе согласно архитектуре Flux были реализованы следующие ее следующие компоненты:

Stores:

- OverviewStore – хранит информацию о сводной информации проекта, такую как такую как количество заданий в проекте, срок, количество участников и текущий прогресс выполнения. Также хранится список последних событий проекта;

- EventsStore – хранит информацию о событиях проекта, отображаемых в календаре приложения;
- MembersStote – используется для учета информации о пользователях проекта;
- TasksStore – хранит информацию о заданиях проекта. Используется для отображения сводной таблицы заданий;
- UserStore – используется для хранения информации о пользователе и его настроек.

Actions:

- updateProjectInfo – обновление информации о проекте;
- updateRecentEvents – обновление последних событий проекта;
- updateTasks – обновление заданий;
- changeTaskStatus – изменение статуса заданий;
- addTask – добавление задания;
- deleteTask – удаление задания;
- addTasklist – добавление списка заданий;
- deleteTasklist – удаление списка заданий;
- updateEvents – обновление событий календаря;
- updateMembers – обновление участников проекта;
- addMember – добавление участника проекта;
- deleteMember – удаление участника проекта;
- updateSettings – обновление настроек проекта;
- changeSettings – изменение настроек проекта;

Views:

В работе была реализована следующая иерархия компонентов: Корневым компонентом является компонент App (выделен красным на рисунке 3.4). Он представляет собой контейнер для размещения в нем всех остальных компонентов приложения и не имеет своей логики функционирования.

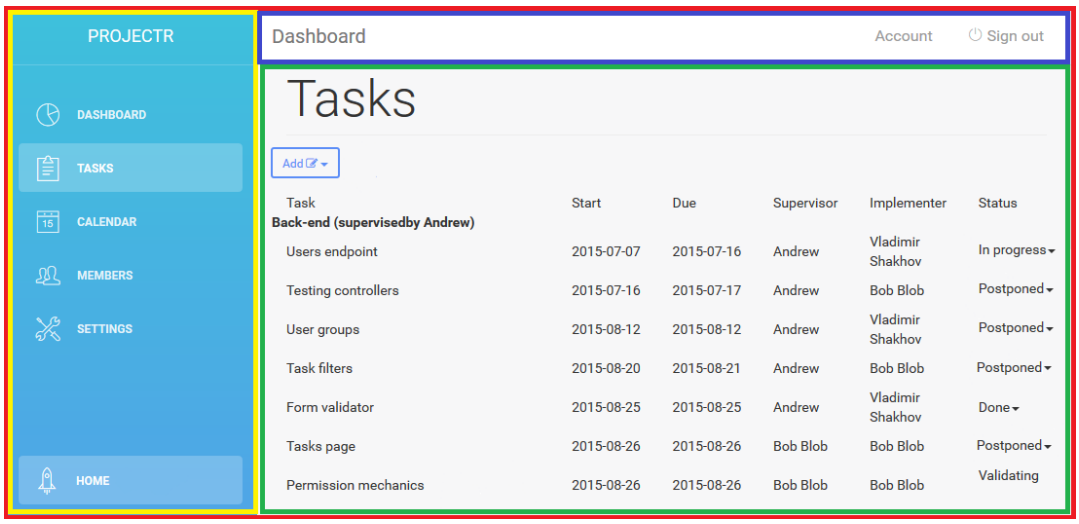

Рисунок 3.4. Иерархия компонентов приложения

В корневом компоненте постоянно содержатся компоненты Sidebar (выделен желтым) и Navbar (выделен синим), а также вариативный компонент, который зависит от того, в каком месте приложения находится пользователь (выделен зеленым). Navbar представляет собой блок, в котором пользователь имеет возможность перехода в свой профиль, а также выхода из системы. Sidebar представляет собой блок боковой навигации, с помощью которой пользователь осуществляет навигацию по приложению. При этом, в зависимости от того, в какой части приложения находится пользователь, центральная область приложения отображает один из следующих компонентов:

- Overview;
- Tasks;
- Calendar;
- Members;
- Settings.

Компонент Overview предоставляет пользователю сводную информацию о проекте, такую как количество заданий в проекте, срок, количество участников и текущий прогресс выполнения.

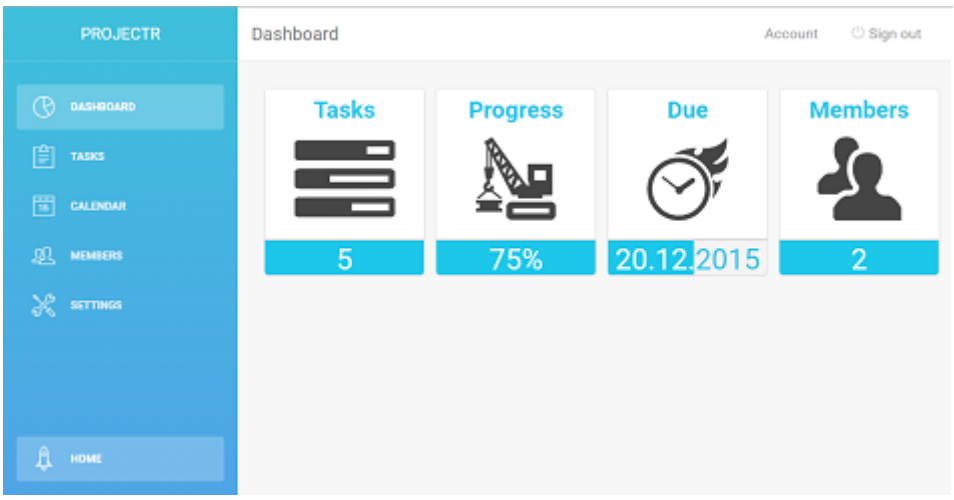

Рисунок 3.5. Компонент Overview

Компонент Tasks является одним из основных и содержит в себе списки заданий для проекта. Пользователи имеют возможность создавать списки заданий и добавлять в них сами задания. Для каждого задания устанавливаются его сроки, ответственный за исполнение и за надзор. Также задание имеет его текущий статус.

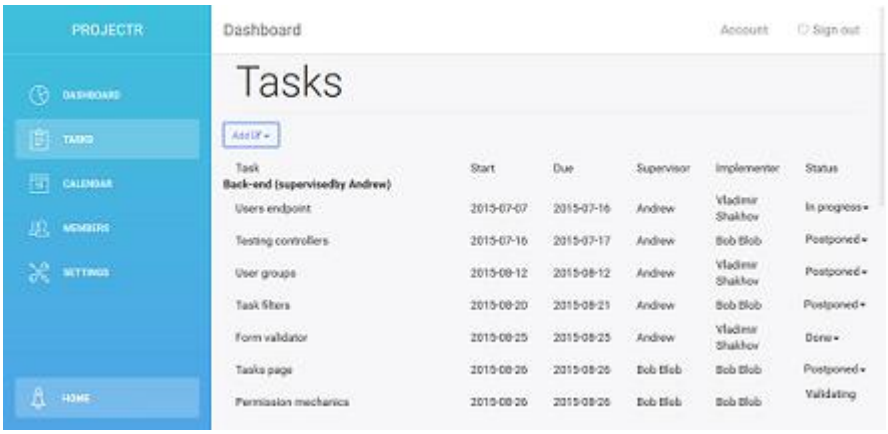

Рисунок 3.6. Компонент Tasks

Компонент Calendar представляет собой календарь, на котором отображаются задания для каждого пользователя в удобном виде, чтобы можно было визуально оценить сроки выполнения заданий.

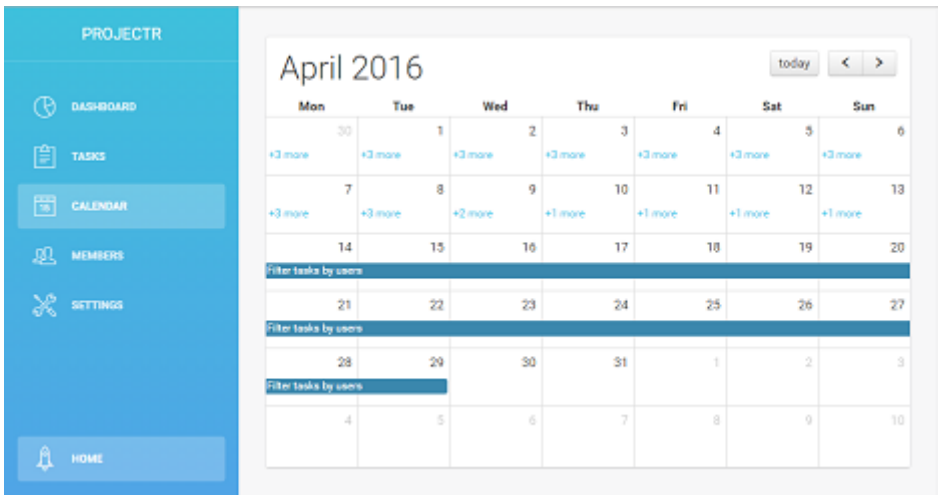

Рисунок 3.7. Компонент Calendar

Компонент Members содержит в себе таблицу участников проекта. При наличии соответствующих прав пользователи могут добавлять в проект участников, удалять участников, а также изменять их статус в проекте.

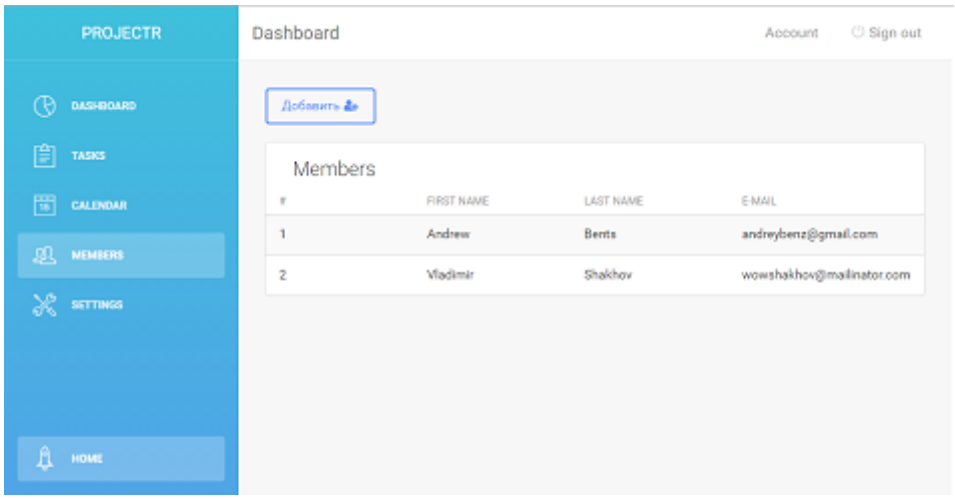

Рисунок 3.8. Компонент Members

Компонент Settings предоставляет возможность изменять настройки проекта, такие как: название проекта, руководитель, сроки выполнения, описание. Также есть возможность удалить проект или покинуть его.

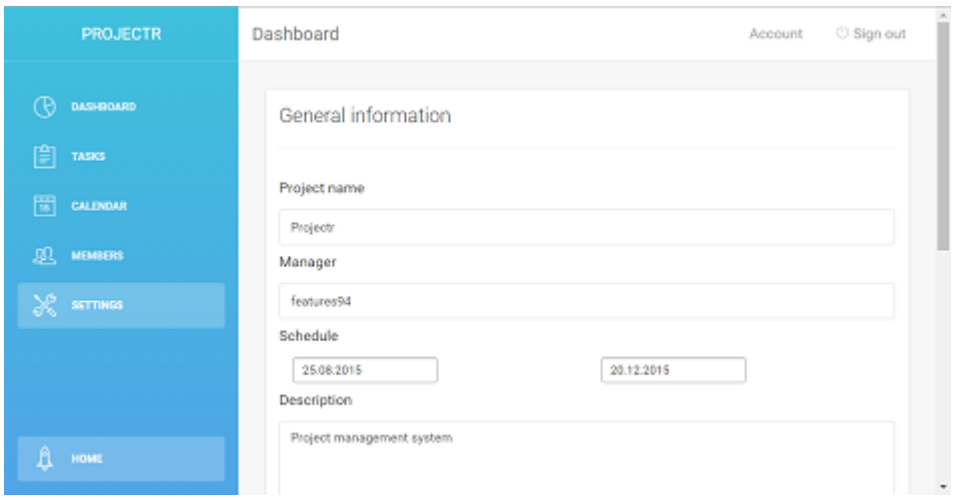

Рисунок 3.9. Иерархия компонентов приложения

3.4 Аутентификация пользователей на основе JSON Web Token

В работе большинства web-приложений возникает необходимость аутентификации пользователей – пользователь после регистрации вводит свои данные и входит в систему, получая доступ к различным ресурсам, доступным зарегистрированным пользователям. Обычно подобное поведение достигается с помощью сессий на сервере. Сервер позволяет запоминать, аутентифицирован ли пользователь, совершающий запрос, и отправлять соответствующий ответ. Однако это подразумевает хранение некоторого состояния о совершающем запрос. Используемая в работе REST-архитектура подразумевает отсутствие состояний сервера. Другими словами, все данные о запросе содержатся в самом запросе, и ответ от сервера не должен меняться в зависимости от его состояния. Это приводит к необходимости использования других методов аутентификации. Один из таких методов – JSON Web Tokens.

Механизм JWT (JSON Web Tokens, веб-токены с использованием JSON) – открытый стандарт (RFC 7519) для передачи так называемых «запросов» (англ. claims) между двумя сторонами, взаимодействующими с помощью веб-приложений. JWT предоставляет пользователям способ аутентифицировать каждый запрос без необходимости сервера поддерживать сессию или клиента отсылать логин и пароль на сервер при каждом запросе [12]. Некоторые особенности JWT:

- Токен содержит в себе время истечения, после которого запросы с участием данного токена будут откланяться;
- Сам по себе токен не зашифрован; любой, кто получил к нему, доступ может его прочесть;
- Однако токен подписан сервером; если его значения изменить, сервер узнает о подмене и отклонит запрос.

JWT состоит из трех главных частей: заголовок, блок полезной нагрузки и подпись. Все три части кодируются с помощью кодировки base64 и соединяются точкой [12].

Далее представлена более подробная информация о содержимом токена.

3.4.1 Заголовок

Заголовок токена состоит из двух полей:

- Тип (typ, на текущий момент поддерживается только значение JWT);
- Алгоритм шифрования (например, HS256).

В результате заголовок будет выглядеть следующим образом:

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

После кодирования base64:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

3.4.2 Полезная нагрузка

В разделе «полезная нагрузка» содержится описание токена и информация, которую необходимо передать. Эти данные располагаются по подразделам [13].

В разделе Registered claims содержится описание токена в виде различных полей с предопределенными именами:

- iss: эмитент токена;
- sub: тема токена;

- aud: получатели токена;
- exp: наиболее часто используемая заявка. Exp определяет время окончания действия токена. Обязательно должно быть больше текущего времени;
- nbf: время, до наступления которого токен недействителен;
- iat: время создания токена. Используется для определения времени существования токена;
- jti: уникальный идентификатор, используется для предотвращения повторного использования токена.

В разделе public claims находятся заявки, создаваемые самими пользователями JWT, например, имя пользователя системы, информация о нем и так далее [13].

Пример блока полезной нагрузки, содержащего два registered claims (iss и exp) и два public claims (name и admin):

```
{
  "iss": "scotch.io",
  "exp": 1300819380,
  "name": "John Smith",
  "admin": true
}
```

3.4.3 Подпись

Последним блоком JWT является подпись, которая формируется в виде хэша от заголовка, полезной нагрузки и некоторого секрета, хранимого на сервере [13]. Пример вычисления на языке JavaScript:

```
var encodedString = base64UrlEncode(header) + "." + base64UrlEncode(payload);

HMACSHA256(encodedString, 'secret');
```

Секрет хранится только на сервере. С его помощью сервер в состоянии подтвердить подлинность токенов и подписать новые.

3.4.4 Процесс аутентификации

При использовании JSON Web Tokens процесс аутентификации выглядит следующим образом:

- Пользователь отсылает логин и пароль на сервер;
- Сервер проверяет данные на правильность, подписывает и отсылает новый токен;
- Токен запоминается на стороне клиента и прилагается к каждому запросу, требующему аутентификацию;
- Сервер в свою очередь проверяет подпись токена и отсылает требуемые данные.

Далее, для всех запросов, требующих аутентификации, клиент должен пересылать полученный токен [13]. Один из распространённых методов пересылки токена заключается в прикреплении его в заголовки HTTP-запроса следующим образом:

```
Authorization: Bearer
```

При этом сервер перед выдачей данных считывает токен из заголовков запроса, проверяет его на подлинность. Если он проходит проверку, требуемые данные отправляются в ответ за запрос.

3.5 Архитектура REST

Серверная часть приложения реализована с использованием архитектуры REST. При использовании этой архитектуры клиент совершает HTTP-запрос на определенный адрес, называемый конечной точкой (endpoint) и сервер, в зависимости от используемого метода, возвращает данные или создает новую запись [16][17]. Например, GET /users/ вернет список всех пользователей, GET /users/1 вернет информацию о пользователе с id равным 1, POST /users/ создаст нового пользователя. При этом данные, возвращаемые сервером, представляются в виде JSON или XML, и на клиентскую часть возлагается отображение этих данных.

В системе поддерживаются следующие конечные точки:

- GET /project/:id – получение информации о проекте с указанным id;
- POST /project – создание нового проекта;
- GET /tasks/:id – получение информации о заданиях проекта;

- POST /tasks/:id – создание нового задания для проекта;
- GET /members/:id – добавление нового участника в проект;
- POST /members/:id – добавление нового участника в проект;
- GET /settings/:id – получение настроек проекта;
- POST /settings/:id – изменение настроек проекта.

Обращение к данным конечным точкам REST-сервиса происходит по мере взаимодействия пользователя с системой.

4 ТЕСТИРОВАНИЕ

В ходе работы производились следующие виды тестирований:

- Unit-тестирование модулей исходного кода приложения, реализующих логику его работы;
- Поведенческое тестирование React-компонентов системы на соответствие различным ситуациям;
- Тестирование производительности приложения с помощью средств разработчика в браузерах.

4.1 Unit-тестирование

При использовании метода unit-тестирования под модулем (англ. unit) понимается наименьший тестируемый участок приложения. В объектно-ориентированном программировании под таким участком обычно понимается некий интерфейс (например, класс). Для каждого такого модуля создаются тесты, которые проверяют его функционал на отсутствие ошибок и на соответствие спецификации. Далее происходит запуск данных тестов. Если модуль успешно прошел тесты, он считается верно реализованным (до следующего внесения в него изменений). Если тесты не были пройдены, модуль отправляется на доработку или отладку [14].

В качестве библиотеки выполнения тестов используется Mocha JS совместно с библиотекой Chai JS.

Mocha содержит средства для организации самого процесса тестирования, а Chai содержит функции, необходимые для проверки различных утверждений [14][15].

Пример теста:

```
describe('Utils', () => {
    it('ignores all props which are not a function', () => {
      const reducer = combineReducers({
        fake: true,
        broken: 'string',
        another: { nested: 'object' },
        stack: (state = []) => state
```

```
        })

        expect(
          Object.keys(reducer({ }, { type: 'push' }))
        ).toEqual([ 'stack' ])
      })
  });
```

4.2 Тестирование React-компонентов

ReactJS предоставляет средства для тестирования, собранные в библиотеке React Test Utilities. С помощью них имеется возможность проверить правильность рендеринга компонентов, его соответствие ожидаемой структуре. Это может подразумевать проверку количества дочерних компонентов, проверку правильности структуры получаемой HTML-разметки, симуляции пользовательских событий [18].

Также предоставляется возможность тестирования посредством метода «shallow rendering», которой использовался при тестировании компонентов в разработанной системе.

4.2.1 Тестирование с помощью Shallow rendering

«Shallow rendering» – подход тестирования React компонентов.

В то время как другие методы воспроизводят полный рендеринг всего компонента, в этом способе используется специальный рендерер, который осуществляет отображение только компонента самого верхнего уровня. Остальная часть иерархии не участвует в рендеринге и так и остается React-компонентом. Однако при этом все еще есть доступ к свойствам таких элементов. Таким образом, имеется возможность проверить поведение компонента [18].

Пример тестирования компонента:

```
import React from 'react/addons';
import { InvitedPerson, InvitationList } from 'components/InvitationList';

describe("Invitation List - testing with shallow rendering", () => {
  beforeEach(function() {
    this.examplePeople = [
      { id: 1, name: "Waldo" },
```

```
      { id: 2, name: "Hercules" }
    ];

    let { TestUtils } = React.addons;

    this.TestUtils = TestUtils;
    this.renderer = TestUtils.createRenderer();
    this.renderer.render(<InvitationList initiallyInvited={this.examplePeople}
/>);
  });

  it("renders a list in a box with proper CSS classes and people within it",
function() {
    let result = this.renderer.getRenderOutput(),
        personOneID = `person_${this.examplePeople[0].id}`,
        personTwoID = `person_${this.examplePeople[1].id}`;

    expect(result.type).toEqual('div');
    expect(result.props.className).toEqual("invitation-list-container");

    expect(result.props.children).toEqual(
      <ul className="invitation-list" ref='list'>
        <InvitedPerson key={personOneID} ref={personOneID}
person={this.examplePeople[0]} />
        <InvitedPerson key={personTwoID} ref={personTwoID}
person={this.examplePeople[1]} />
      </ul>
    );
  });
});
```

Описанный тест проверяет, что корневой элемент является div-элементом, что он имеет имя класса «invitation-list-container», а также проверяет дочерние элементы на соответствие ожидаемым.

Достоинства метода:

- Тесты, написанные таким образом, как правило, более краткие, чем тесты, написанные на основе других подходов;
- Компоненты нижнего уровня не могут повлиять на результаты теста (кроме самого верхнего), что делает его отличным инструментом для модульного тестирования;
- Вам не нужен реальный DOM, чтобы проверить свои компоненты. Это делает тесты, написанные таким образом очень быстрыми;

Недостатки:

- Так как компоненты низкого уровня не инстанциируются, трудно делать предположения в тесте об их состоянии.

4.3 Тестирование с помощью средств разработчика в браузере

Браузер Google Chrome предоставляет в панели разработчика средства, анализирующие загрузку страницы. Это осуществляется во вкладке Audits. После анализа разработчику предлагаются предложения по оптимизации и уменьшении времени загрузки страницы и способы улучшения воспринимаемой пользователем первоначальной отзывчивости приложения. Например, страница может содержать неиспользуемые CSS-правила, удаление которых уменьшит размер загружаемого файла и работу браузера по его лексическому разбору. Пример анализа представлен на рисунке 4.1.

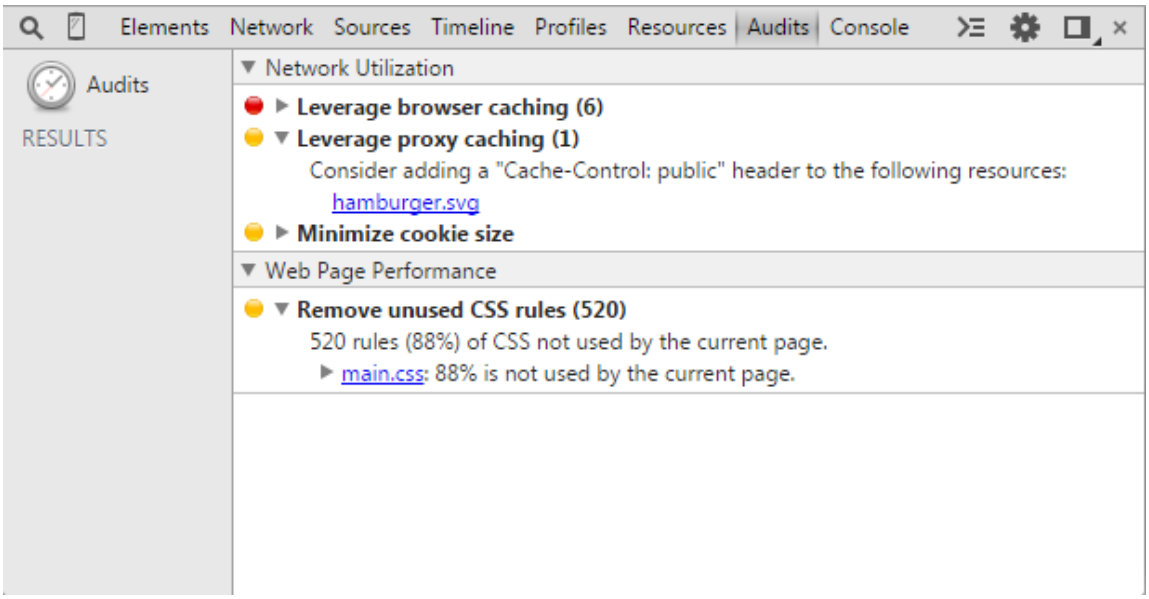

Рисунок 4.1. Анализ загрузки страницы

Во вкладке Timeline показывается полный обзор времени, затрачиваемого на обработку различных частей страницы при ее загрузке и использовании. Для анализа доступна информация о загрузке ресурсов, разбора JavaScript файлов, расчета стилей и перерисовки элементов страницы. Вся информация отображается на графике. Пример представлен на рисунке

Рисунок 4.2. Анализ работы со страницей

4.4 Результаты тестирования

4.4.1 Общее описание условий тестирования

В ходе работы было произведено тестирование производительности приложения с помощью средств разработчика браузера Google Chrome, а также unit-тестирование и поведенческое тестирование пользовательского интерфейса.

Тестирование производилось на ПК следующей конфигурации:

- Процессор: Inter Core i5-2450M CPU @2.50Ghz;
- Оперативная память: 4Гб;
- Операционная система: Microsoft Windows 10;
- Браузер: Google Chrome версии 51.0.2704.84 m.

4.4.2 Тестирование производительности

В ходе тестирования производительности средствами разработчика браузера Google Chrome было выявлено нарушение порядка загрузки на странице JavaScript и CSS-файлов, что потенциально может привести к увеличению времени ее загрузки. Также было обнаружено 500 неиспользуемых CSS-правил, наличие которых увеличивает размер CSS-файлов и, соответственно, увеличивает время загрузки страницы. Обнаруженные недостатки были устранены.

Также, в ходе тестирования производительности было выявлено, что время загрузки первоначальных ресурсов не превышает 350 мс, время

обработки ресурсов браузером не превышает 250 мс. Соответственно, общее время полной загрузки первоначальной страницы находится в пределах 600 мс.

В процессе работы с приложением время обработки пользовательских событий не превышает 250мс, время рендеринга содержимого одного кадра страницы браузером не превышает 10 мс.

Полученные результаты свидетельствуют о высокой производительности разработанного приложения.

4.4.3 Результаты unit-тестирования и поведенческих тестов

Приложение содержит 18 модулей. Для каждого из модулей были написаны unit-тесты. Покрытие тестов составляет 100%, что означает соответствие результата работы модулей ожидаемым.

Пользовательский интерфейс состоит из 7 модулей. Для каждого модуля написаны поведенческие тесты. Покрытие тестов составляет 100%, что означает, что поведение реализованного пользовательского интерфейса соответствует ожидаемому.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8В2А	Бенц Андрей Сергеевич

Институт	Кибернетики	Кафедра	Вычислительной техники
Уровень образования	Бакалавр	Направление/специальность	09.03.01

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих* 2. *Нормы и нормативы расходования ресурсов* 3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*На основании информации, представленной в научных статьях и публикациях, аналитических материалах, статистических бюллетенях и изданиях, нормативно-правовых документах, определить методику расчета экономической эффективности.*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения* 2. *Формирование плана и графика разработки и внедрения инженерного решения* 3. *Формирование организационной структуры управления инженерным проектом* 4. *Планирование потребности в человеческих ресурсах*	*Оценка ресурсной, социальной эффективности НИ и потенциальных рисков.*
Перечень графического материала *(с точным указанием обязательных чертежей)***:**	
1. *Оценка плана проведения НИ* 2. *QuaD-технология* 3. *Матрица SWOT* 4. *Морфологический анализ*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Николаенко Валентин Сергеевич	-		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В2А	Бенц Андрей Сергеевич		

5 МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

5.1 Введение

Управление проектами занимает большую нишу в практически любом начинании отдельного человека или команды людей и в большой степени определяет успех проекта и будет ли этот проект завершен.

В работе разрабатывается web-ориентированное приложение для управления проектами, представляющее собой RESTful-сервис на серверной части приложения, и одностраничное web приложение на клиентской части. Приложение разработано с использованием адаптивной верстки, что позволяет ему быть оптимизированным под различные разрешения экранов, то есть иметь удобный пользовательский интерфейс как на персональных компьютерах, так и на мобильных устройствах.

Целью данного раздела выпускной квалификационной работы является оценка ее параметров, таких как перспективность, ресурсоэффективность и коммерческого потенциала.

Для достижения целей производится оценка с помощью Quad- и SWOT-анализов. Для проведения Quad-анализа необходимо выполнение следующих задач:

- Выделение основных показателей оценки качества разработки и показателей коммерческого потенциала разработки;
- Определение весов выделенных показателей и вычислить средневзвешенное значение показателя качества и перспективности;
- Проанализировать полученные результаты.

Для проведения SWOT-анализа необходимо выполнение следующих задач:

- Выявить сильные и слабые стороны, а также угроз и возможностей проекта;
- Установить связи между ними;
- Проанализировать результаты.

5.2 Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

5.2.1 Потенциальные потребители результатов исследования

Произведем анализ рынка потенциальных потребителей. Разрабатываемый продукт представляет собой систему управления проектами. В качестве проекта может выступать любое намерение исполнителя выполнить некоторую задачу. Далее исполнитель декомпозирует свой проект на различные задания. Соответственно, разрабатываемая система может использоваться как отдельными пользователями, которые решили привнести четкую организацию выполнения своего проекта, а также небольшими командами (например, студентов, выполняющих различные учебно-исследовательские работы) или же различными организациями, которые управляют через подобные системы процессом выполнение своих проектов.

5.2.2 Технология QuaD

Технология QuaD инструмент для количественной оценки характеристик, описывающих качество разработки и ее перспективность на рынке, используя такие характеристики, как эффективность, конкурентоспособность и другие.

В ходе применения QuaD анализа были выделены следующие критерии оценки:

1. Потребность в ресурсах – необходимое для приемлемой работы системы конфигурация оборудования;

2. Кроссплатформенность – возможность использования разработанной системы на различных устройствах под управлением различных операционных систем, в том числе и мобильных устройств;

3. Функциональная мощность – набор возможностей, которые предоставляет разработанное приложение и качество реализации данных возможностей;

4. Простота эксплуатации – понятность представления системой информации, удобство использования пользовательского интерфейса.

5. Устойчивость к сбоям – способность системы к предотвращению ситуаций, в которых наблюдается отказ в обслуживании, способность системы в кратчайшие сроки восстанавливаться после возникновения подобных ситуаций.

6. Оказываемая нагрузка на сеть – минимальный объем передаваемых данных по сети, необходимый для обеспечения оптимальной работы приложения, без ощутимых задержек и нарушений производительности.

Оценочная карта для сравнения конкурентных технических решений приведена в таблице 5.1:

Таблица 5.1 – Оценочная карта

Критерии оценки	Вес	Баллы	Макс. балл	Отн. знач.	Ср.-взвеш. знач.
Показатели оценки качества разработки					
1. Потребность в ресурсах	0,10	70	100	0,7	0,07
2. Кроссплатформенность	0,10	80	100	0,8	0,08
3. Функциональная мощность	0,15	70	100	0,7	0,105
4. Простота эксплуатации	0,1	80	100	0,8	0,08
5. Устойчивость к сбоям	0,15	90	100	0,9	0,135
6. Оказываемая нагрузка на сеть	0,1	70	100	0,7	0,07
Показатели оценки коммерческого потенциала разработки					
7. Доступность	0,05	80	100	0,8	0,04
8. Перспективность рынка	0,2	80	100	0,8	0,16
9. Законченность работы	0,05	70	100	0,7	0,035
				Итог:	0,755

Оценка качества и перспективности по технологии QuaD определяется по формуле:

$$\Pi_{\text{ср}} = \sum B_i \cdot \text{Б}_i ,$$

где Пср – средневзвешенное значение показателя качества и перспективности научной разработки; Bi – вес показателя (в долях единицы); Бi – средневзвешенное значение i-го показателя.

Можно видеть, что интегральный показатель конкурентоспособности разработки составляет 0,755, что является достаточно благоприятным для продолжения разработки.

5.2.3 SWOT-анализ

SWOT-анализ предполагает выявление сильных и слабых сторон разработки, выявление угроз и возможностей. После выявления данных факторов необходимо произвести установление связей между ними.

Результаты анализа представлены в таблице 5.2:

Таблица 5.2 – Результаты SWOT-анализа

	Сильные стороны 1. Использование современных web-технологий 2. Скорость работы 3. Отсутствие встроенных покупок 4. Возможность использования на различных платформах 5. Востребованность в системах управления проектами	**Слабые стороны** 1. Незавершенность проекта 2. Отсутствие поддержки старых браузеров 3. Наличие конкурентов
Возможности 1. Потенциал к	B1C1: использование современных технологий облегчит дальнейшее	B1Сл1: при развитии системы будет устранена

развитию системы 2. Продажа готовой версии продукта	совершенствование системы	незавершенность проекта
Угрозы 1. Отсутствие финансирования и привлеченной рабочей силы 2. Ошибки на стадиях проектирования и реализации 3. Ограниченный функционал	У2С2: скорость работы системы может быть значительно снижена как следствие ошибок проектирования и реализации	У3Сл3: из-за недостаточной функциональной мощности, варианты конкурентов могут пользоваться большим спросом

Таким образом можно прийти к выводу, что основными рисками при дальнейшей разработке и продвижении системы является отсутствие ресурсов, а также большое количество конкурентов, продукты которых обладают схожим или превосходящим функционалом.

5.3 Определение возможных альтернатив проведения научных исследований

В виде альтернативы был произведен морфологический анализ. Его сущность заключается в выделении характеристик рассматриваемого объекта и рассмотрении различных комбинаций найденных характеристик. Результаты анализа представлены в таблице.

Таблица 5.3 – Результаты морфологического анализа

	1	2	3
А. Платформа	Десктоп-приложение	Web-приложение	Комплекс приложений (десктоп, мобильные приложение, веб-клиент и др.)
А. Архитектура серверной части приложения	MVC	MVP	MVVM
В. Используемая СУБД	MySQL	MongoDB	PostreSQL

Б. Способ клиент-серверного взаимодействия	REST	GraphQL	PJAX
В. Платформа реализации серверной части приложения	PHP	NodeJS	Python (Django)
Г. Тип клиентской части приложения	Классическое приложение с элементами интерактивности	Одностраниченое приложение (SPA)	Смешанное приложение
Е. Средства реализации клиентской части	ReactJS	AngularJS	Jquery
И. Тип приложения по количеству пользователей	Однопользовательское	Многопользовательское	
К. Модель распространения	Платная	Бесплатная	Со встроенными покупками

5.4 Формирование организационной структуры управления инженерным проектом

В разработке проекта на задействовано три исполнителя – научный руководитель проекта и два программиста.

Задачами одного из программиста являются: проектирование базы данных разрабатываемой системы, обеспечения взаимодействия серверной части приложения с базой данных, проектирование и реализация RESTful-сервиса, обеспечивающего взаимодействия клиентской части приложения с серверной.

Задачами одного из программиста являются: проектирование базы данных разрабатываемой системы, обеспечения взаимодействия серверной части приложения с базой данных, проектирование и реализация RESTful-сервиса, обеспечивающего взаимодействия клиентской части приложения с серверной.

Задачами другого программиста являются: проектирование и реализация пользовательского интерфейса приложения, реализация одностраничного web-приложения, реализация взаимодействия клиентской части с RESTful сервисом посредством асинхронных запросов.

Организационная структура проекта является линейно-функциональной, как следствие задействования на начальном этапе разработки трех исполнителей. В дальнейшем организационная структура может быть изменена, например, для увеличения эффективности выполнения работы, а также расширения списка работ, требующих выполнения.

Изменение организационной структуры может проявляться, например, в создании функциональных подразделений, ответственных за реализацию продукта, техническую поддержку пользователей и так далее.

5.5 Планирование потребности в человеческих ресурсах

При разработке данного проекта не возникло необходимости в увеличении количества задействованных в работе над проектом исполнителей. Описанная выше организационная структура оказалась оптимальной для достижения поставленных целей, однако при росте и развитии системы одновременно с этим увеличивается потребность в человеческих ресурсах и эффективной организации их работы. При дальнейшей работе над проектом возможно изменение организационной структуры и расширение числа задействованных работников для достижения оптимальной степени разделения труда и минимизации временных затрат на выполнение работ над проектом.

5.6 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

В данном разделе была произведена оценка таких параметров выпускной квалификационной работы, как перспективность, ресурсоэффективность и коммерческого потенциала

Проведенный SWOT-анализ позволил обнаружить слабые и сильные стороны проекта, позволил выделить возможности и угрозы проекта. С учетом выделенных сторон проекта были установлены связи между ними. На основании проведенного анализа появилась возможность сформулировать дальнейшее направление действий. В целом результаты SWOT-анализа показали целесообразность работы над проектом.

Анализ, проведенный по технологии Quad, показал довольно высокую перспективность проекта, что выражается в довольно высоком значении средневзвешенного показателя качества и перспективности научной разработки, равного 0,755.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8В2А	Бенц Андрей Сергеевич

Институт	ИК	Кафедра	ВТ
Уровень образования	бакалавриат	Направление/специальность	09.03.01 «Информатика и вычислительная техника» профиль «Вычислительная техника»

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Web-ориентированная система управления проектами. Предназначена для организации и систематизации работ над проектами.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Некорректная работа серверной части программы	– Некорректное взаимодействие серверной части приложения с базой данных – Некорректно спроектированная база данных, как следствие возможность нарушения выполнения некоторых запросов – Затруднение при извлечении информации для отображения – Использование Unit-тестирования для автоматизированного тестирования модулей приложения
2. Неправильно спроектированный интерфейс	– Замедление или невозможность работы с системой как следствие ошибок в интерфейсе системы – Замедление процесса работы и склонность пользователей к совершению из-за ошибок проектирования пользовательского интерфейса – Необходимость тщательного проектирования интерфейса – Использование поведенческого подхода к тестированию компонентов.
3. Утечка данных сотрудников	– Подверженность системы к утечке данных при ошибках в реализации защиты от несанкционированного доступа

	– Последствия утечки данных сотрудников – Рассмотрение других каналов утечек, таких как акустических и оптических, защита от которых не может быть предусмотрена самой системой
4. Утечка данных об организации	– Негативные последствия утечки данных об организации – Негативные последствия утечки данных о выполняемых организацией проектов. – Системы защиты от несанкционированного доступа к конфиденциальным ресурсам – Невозможность предотвращения утечки информации по акустическим и оптическим каналам
5. Потеря данных	– Последствия потери данных о пользователях и организациях – Потеря данных из-за некорректной работы базы данных – Потеря данных из-за некорректной работы серверной части, взаимодействующей с базой данных – Потеря данных из-за ошибочных действий пользователя – Тестирование взаимодействия с базой данных

Дата выдачи задания для раздела по линейному графику	

–

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Невский Егор Сергеевич			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В2А	Бенц Андрей Сергеевич		

6 СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

6.1 Введение

Конечный продукт, полученный в ходе выполнения, представляет собой web-ориентированную систему управления проектами. Предполагается использование системы большим количеством людей.

Так как разрабатываемая система является программным продуктом, влияние ее на окружающую среду отсутствует, однако все же присутствует определенное влияние на людей, взаимодействующих с разрабатываемой системой. Пользователи ожидают безупречную и безотказную работу приложения, однако в определённых обстоятельствах в его реализации и сопровождении могут быть допущены различные ошибки, которые в свою очередь могут повлечь негативные последствия, отражающиеся в различных сферах деятельности человека.

Большая часть рисков заключается в том, что система хранит и обрабатывает различные данные, которые нуждаются в обеспечении целостности, конфиденциальности и доступности. Нарушение какого-либо из этих требований может повлечь за собой негативные последствия.

В разделе «социальная ответственность» выпускной квалификационной работы рассматриваются различные факторы, которые могут привести к нарушению целостности, конфиденциальности и доступности данных, хранимых в системе, а также различные последствия нарушения этих требований и некоторые варианты их предупреждения.

Как и в любой другой информационной системе, в разработанной системе присутствуют определенные факторы, которые могут повлечь за собой нарушение нормальной работы системы и принести негативные последствия как для отдельных пользователей системы, так и для всей организации в целом. Такие факторы возникают потому, что разработанная система хранит и обрабатывает данные о пользователях, об организациях, а также непосредственно о проектах, которые выполняют участники системы. Следовательно, в случае неудачной реализации какого-либо компонента

системы, возникает вероятность нарушения целостности, конфиденциальности и доступности данных пользователей.

Под целостностью информации подразумевается гарантия того, что информация остается неизменной и корректной в том смысле, что в процессе передачи и хранения информации она остается неизменна. Как было упомянуто выше, разработанная система хранит данные о состоянии различных проектов пользователей. Повреждение этой информации может серьезно замедлить или даже приостановить процесс работы организации над проектом. Соответственно возникает необходимость обеспечения надежного хранения информации и безошибочного выполнения операций над хранимой информацией.

Под доступностью информации понимается гарантия того, что авторизованные пользователи могут получить доступ и работать с информацией по требованию. Возможны случаи, в которых скорость получения доступа к информации будет неприемлемой для данного вида работ, или даже полное отсутствие доступа к необходимой информации. Это очевидно замедляет выполнение проекта.

Понятие конфиденциальности гарантирует, что информация может быть получена и обработана только теми лицами, которые на это уполномочены. Другими словами, осуществляется защита информации от несанкционированного доступа. Следствием нарушения данного понятия в контексте разработанной системы является утечка данных о проекте, что несет за собой довольно негативные последствия, если проект является коммерческим и за счет произошедшей утечки данных конкуренты организации могут получить выгоду. Также может произойти утечка персональных данных участников проектов, что также не исключает возможность негативного эффекта. Например, была необходимость скрыть нахождение в проекте каких-либо участников, или же, сопоставив качество и скорость выполнения работы какого-либо сотрудника, специалисты по подбору

персонала из конкурентных организаций могут предпринять попытку перехвата работника.

Таким образом, разработанная система содержит в себе компоненты, недоработка или неправильная реализация которых может крайне негативно отразиться на пользователях. Если рассмотреть данные компоненты более детально, можно выделить следующие возможные сценарии возникновения ошибок:

- Некорректная работа серверной части программы;
- Неправильно спроектированный интерфейс;
- Утечка данных сотрудников;
- Утечка данных об организации;
- Потеря данных.

6.2 Некорректная работа серверной части программы

Под ошибками в логике работы системы понимается либо отличное от задуманного функционирование различных компонентов системы, либо же полное отсутствие требуемого функционала как следствие ошибок в реализации приложения.

Подобные проблемы могут проявляться из-за ошибок в коде серверной части приложения, которая взаимодействует с базой данных, или же ошибочная структура базы данных сама по себе.

В результате, например, новые задания в проекте не будут отображаться, или же будет отсутствовать возможность изменения статуса задания, или любые другие операции над проектом.

Очевидны негативные последствия подобного рода ошибок: пользователи, сами того не желая, могут нарушить целостность своего проекта, либо же пользователи просто не смогут продолжать выполнение проекта, что в конечном итоге ведет к простоям в работе организации и отодвиганию сроков выполнения проекта.

Как способ предотвращения появления подобных ошибок во время реализации приложения производится автоматизированное тестирование каждого его компонента и модуля. Это можно производит с помощью Unit-тестирования – процесса, позволяющего проверить на корректность отдельные модули исходного кода программы. Идея данного тестирования состоит в том, что составляются тесты для каждой функции или метода, что в свою очередь позволяет проверить правильность уже написанных составляющих системы, а также проверить, не привел ли новый код в регрессии системы, то есть к появлению ошибок в уже оттестированных местах программы. Соответственно, в ситуациях, когда код приложения не проходит тесты, производится либо уточнения тестов, либо устранение ошибок в реализации.

6.3 Неправильно спроектированный интерфейс

Пользовательский интерфейс, по сути, является единственным средством взаимодействия пользователя с приложением. Чем лучше спроектирован интерфейс, тем проще пользователю выполнять свои задачи и тем быстрее он с ними справляется. Однако, различные ошибки в проектировании и реализации пользовательского интерфейса могут привести к существенному замедлению или даже невозможности выполнения пользователями их задач или даже к негативным последствиям, таким как повреждение структуры проектов. Даже при наличии незначительных ошибок в пользовательском интерфейсе ухудшается опыт взаимодействия пользователей с приложением, что приводит к появлению негативных эмоций, необходимости заново воспроизводить некоторые действия, и к увеличению количества совершаемых ошибок.

Во избежание негативного пользовательского опыта использования приложения необходимо уделить время проектированию пользовательскому интерфейсу. Так, например, не стоит придумывать что-то кардинально новое и непонятное, так как существуют некоторые базовые элементы интерфейсов, которые очень распространены и используются во многих приложениях. Пользователи привыкли к их использованию и быстрее смогут начать работать с системой. Также стоит расположить часто используемые элементы в одном очевидном месте, чтобы сократить время доступа к этим элементам. К тому же, необходимо наличие различных окон с подтверждением совершения действий, либо с информацией о том, что действие не было завершено. Примеры подобных сообщений:

- «Задание успешно добавлено»;
- «Добавление задания не удалось»;
- «Вы уверены, что хотите удалить задание?».

Для предотвращения ошибок в реализации пользовательского интерфейса используется поведенческое тестирование, основанное на Unit-тестировании, описанном выше.

Описываются тесты, заключающиеся в различных состояниях, в которых может находиться компонент и его ожидаемое содержимое. Также присутствует возможность изменение состояния компонента с помощью эмулирования различных действий пользователя, таких как нажатие мышью.

6.4 Утечка данных сотрудников

Как уже было упомянуто выше, разработанное приложение хранит данные о пользователях. Эти данные могут носить конфиденциальный характер, и при их утечке может нанести вред организации или тем лицам, чьи данные содержатся в утечке. Кроме базовой информации о самом человеке может быть похищена информация о его номере телефона, адресе электронной почты и другая информация.

Очевиден негативный эффект утечки таких данных. Соответственно необходимо тщательное продумывание всех путей запрашивания данных разными пользователями с разными типами привилегий в системе.

В разработанной системе весьма ограниченный набор подобных данных, которые можно хранить о пользователе. Отсутствует хранение информации, на которое требуется специальное разрешение. Однако, все же необходимо исключить вероятность любого несанкционированного доступа к данным участников системы.

Тем не менее, подобная защита не исключает все каналы утечки информации, такие как оптические, когда потенциальные злоумышленники могут увидеть конфиденциальную информацию на мониторе у пользователя, и акустические, когда злоумышленник может различными способами подслушать интересующие его данные.

6.5 Утечка данных об организации

В разработанной системе предполагается хранение данных об организации, которая выполняет сам проект. Утечка данных может повлечь за собой негативные последствия, если проект является коммерческим и за счет

произошедшей утечки данных конкуренты организации могут получить выгоду.

К данным, утечка которых может представлять угрозу можно отнести информацию о ходе выполнения проекта. При утечке этих данных конкуренты могут получить преимущество, например, зная, когда планируется финальный выпуск продукта, конкуренты могут запланировать выпуск своего продукта раньше.

Также может быть получена информация о составе самого проекта, что может позволить конкурентной организации составить представление о том, из каких частей состоит разрабатываемый продукт, что в свою очередь позволит им реализовать аналогичный продукт или улучшить свой существующий.

Соответственно, в разрабатываемой системе должны присутствовать способы устранения попыток несанкционированного доступа к конфиденциальным ресурсам неуполномоченным на то пользователям. Это предполагает наличие:

- Системы авторизации пользователей;
- Отсутствие доступа к информации в системе для неавторизованных пользователей;
- Проверка каждого запроса на наличие прав на его совершение.

Однако, как было упомянуто выше, подобные методы не могут защитить от акустического и оптического каналов утечки.

6.6 Потеря данных

Разработанная система хранит данные о состоянии различных проектов пользователей. Повреждение этой информации или отсутствие оперативного доступа к этой информации может серьезно замедлить или даже приостановить процесс работы организации над проектом. В лучшем случае придется потратить много времени, чтобы восстановить потерянные данные, в худшем случае потерянные данные невозможно будет восстановить.

При потере информации о заданиях какого-либо участника, будет невозможно приступить к выполнению этих заданий, что может привести к простоям выполнения всего проекта.

Потеря данных возможна как следствие различных событий. Например, пользователь может случайно совершить непреднамеренную операцию и удалить часть информации. Как было упомянуто выше, для решения подобных ситуаций используются различные окна подтверждения совершения действий.

Возможны ошибки в запросах, посылаемых базе данных и логике компонентов, которые непосредственно обмениваются с ней информацией. Некорректно составленные запросы могут препятствовать извлечению необходимой информации из базы данных. Также могут выполняться запросы, приводящие к удалению существующих данных, изначально не предназначавшиеся для удаления.

Все это приводит к тому, что необходимо проводить тщательное тестирование всех запросов к базе данных, совершаемых системой, исследования их влияния на ее содержимое и определение возможных случаев их некорректной работы.

6.7 Вывод

В данном разделе выпускной квалификационной работы было рассмотрено влияние разрабатываемого продукта на общество.

Были выявлены основные факторы, которые могут нанести вред пользователям приложения, такие как:

- Некорректная работа серверной части программы;
- Неправильно спроектированный интерфейс;
- Утечка данных сотрудников;
- Утечка данных об организации;
- Потеря данных.

Перечисленные факторы были подробно рассмотрены с целью выявления различных причин их возникновения наряду с возможными способами предотвращения их появления.

ЗАКЛЮЧЕНИЕ

В работе было реализовано клиентское одностраничное web-приложение, использующее RESTful-сервис в качестве серверной части.

Разработанное приложение позволяет пользователям создавать проекты, добавлять в него участников, становиться участником других проектов. В проекте присутствует возможность добавления заданий и назначения исполнителей и руководителей. Руководитель может следить за статусом выполнения заданий. Также для оценивания пользователями сроков заданий, в системе присутствует календарь.

В ходе сравнения разработанной системы с существующими аналогами были выявлены недостатки по сравнению с Microsoft Project и Wrike:

- Отсутствие возможности нахождения критического пути проекта;
- Отсутствие возможности отображения состояния проекта с помощью различных графиков;
- Уведомление пользователей по e-mail и с помощью push-уведомлений.

По результатам тестирования исходный код проходит все написанные тесты, приложение имеет высокую производительность.

Дальнейшая разработка включает в себя расширение функционала системы, в том числе, с целью устранения недостатков, найденных при сравнении системы с существующими аналогами, и оптимизацию существующего с целью увеличения производительности приложения и увеличения эффективности работы пользователя с ним.

СПИСОК ПУБЛИКАЦИЙ

1. Бенц, А. С. Устранение дефектов на оцифрованных изображениях [Электронный ресурс] / А. С. Бенц, В. С. Шахов, П. А. Хаустов // Молодежь и современные информационные технологии: сборник трудов XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых, г. Томск, 12-14 ноября 2014 г.в 2 т. / Национальный исследовательский Томский политехнический университет (ТПУ), Институт кибернетики (ИК) ; ред. кол. Е. А. Сикора [и др.]. — Томск: Изд-во ТПУ, 2014. — Т. 2. — [С. 72-73]. — Заглавие с титульного экрана. — Свободный доступ из сети Интернет. — Adobe Reader. Режим доступа: http://www.lib.tpu.ru/fulltext/c/2014/C04/V2/029.pdf

2. Бенц, А. С. Обнаружение автомобилей на аэрофотоснимках [Электронный ресурс] / А. С. Бенц, В. С. Шахов, П. А. Хаустов // Молодежь и современные информационные технологии: сборник трудов XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых, г. Томск, 12-14 ноября 2014 г.в 2 т. / Национальный исследовательский Томский политехнический университет (ТПУ), Институт кибернетики (ИК) ; ред. кол. Е. А. Сикора [и др.]. — Томск: Изд-во ТПУ, 2014. — Т. 2. — [С. 74-75]. — Заглавие с титульного экрана. — Свободный доступ из сети Интернет. — Adobe Reader. Режим доступа: http://www.lib.tpu.ru/fulltext/c/2014/C04/V2/030.pdf

3. Бенц, А. С. Исследование зависимости качества классификации от различных параметров ИНС [Электронный ресурс] / А. С. Бенц, А. Я. Браневский, П. А. Хаустов // Молодежь и современные информационные технологии : сборник трудов XI Международной научно-практической конференции студентов, аспирантов и молодых ученых, г. Томск, 13-16 ноября 2013 г. / Национальный исследовательский Томский политехнический университет (ТПУ) ; под ред. Е. А. Сикоры и др.. — Томск: Изд-во ТПУ, 2013. — [С. 139-141]. — Заглавие с титульного экрана. — Свободный доступ из сети

Интернет. — Adobe Reader. Режим доступа: http://www.lib.tpu.ru/fulltext/c/2013/C04/060.pdf

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Facebook Code [Электронный ресурс]: ReactJs. URL: https://facebook.github.io/react/. – Свободный доступ. (дата обращения: 08.06.2016).

2. Facebook Code [Электронный ресурс]: Flux Architecture. URL: https://facebook.github.io/flux/. – Свободный доступ. (дата обращения: 08.06.2016).

3. Wrike [Электронный ресурс]: Описание Wrike. URL: https://www.wrike.com/ru/ru/. – Свободный доступ. (дата обращения: 08.06.2016).

4. Microsoft Project [Электронный ресурс]: Описание Microsoft Project. URL: https://products.office.com/ru-ru/project/project-and-portfolio-management-software. – Свободный доступ. (дата обращения: 08.06.2016).

5. Ember JS [Электронный ресурс]: Описание и документация к Ember JS. URL: http://emberjs.com/. – Свободный доступ. (дата обращения: 08.06.2016).

6. Angular JS 2 [Электронный ресурс]: Описание и документация к Angular JS 2. URL: https://angular.io/. – Свободный доступ. (дата обращения: 08.06.2016).

7. Webpack github page [Электронный ресурс]: Описание и документация к сборщику модулей Webpack. URL: https://webpack.github.io/. – Свободный доступ. (дата обращения: 08.06.2016).

8. Babel loader github page [Электронный ресурс]: Описание и документация расширения Babel loader к сборщику модулей Webpack. URL: https://github.com/babel/babel-loader. – Свободный доступ. (дата обращения: 08.06.2016).

9. Style loader github page [Электронный ресурс]: Описание и документация расширения Style loader к сборщику модулей Webpack. URL: https://github.com/webpack /style-loader. – Свободный доступ. (дата обращения: 08.06.2016).

10. CSS loader github page [Электронный ресурс]: Описание и документация расширения CSS loader к сборщику модулей Webpack. URL: https://github.com/webpack /css-loader. – Свободный доступ. (дата обращения: 08.06.2016).

11. Bootstrap [Электронный ресурс]: Bootstrap Components. URL: http://getbootstrap.com/components/. – Свободный доступ. (дата обращения: 08.06.2016).

12. JWT [Электронный ресурс]: JSON Web Tokens. URL: https://jwt.io/. – Свободный доступ. (дата обращения: 08.06.2016).

13. Auth0 [Электронный ресурс]: Get Started with JSON Web Tokens. URL: https://auth0.com/learn/json-web-tokens/. – Свободный доступ. (дата обращения: 08.06.2016).

14. Mocha JS [Электронный ресурс]: Документация к фреймворку тестирования Mocha JS. URL: https://mochajs.org/. – Свободный доступ. (дата обращения: 08.06.2016).

15. Chai JS[Электронный ресурс]: Документация к библиотеке средств сравнения Chai JS . URL: http://chaijs.com/. – Свободный доступ. (дата обращения: 08.06.2016).

16. Elkstein [Электронный ресурс]: Learn REST: A Tutorial. URL: http://rest.elkstein.org/. – Свободный доступ. (дата обращения: 08.06.2016).

17. UCI School of Information and Computer Sciences [Электронный ресурс]: Representational State Transfer (REST). URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. – Свободный доступ. (дата обращения: 08.06.2016).

18. Facebook Code [Электронный ресурс]: Тестирование React-компонентов. URL: https://facebook.github.io/react/docs/test-utils.html. – Свободный доступ. (дата обращения: 08.06.2016).

ПРИЛОЖЕНИЕ А

CSS-ПРАВИЛА

Файл *styles.css*

```
body,
h1, .h1,
h2, .h2,
h3, .h3,
h4, .h4,
h5, .h5,
h6, .h6,
p,
.navbar,
.brand,
.btn-simple,
.alert,
a,
.td-name,
td,
button.close {
  -moz-osx-font-smoothing: grayscale;
  -webkit-font-smoothing: antialiased;
  font-family: "Roboto","Helvetica Neue",Arial,sans-serif;
  font-weight: 400;
}

h1, .h1, h2, .h2, h3, .h3, h4, .h4 {
  font-weight: 300;
  margin: 30px 0 15px;
}

h1, .h1 {
  font-size: 52px;
}

h2, .h2 {
  font-size: 36px;
}

h3, .h3 {
  font-size: 28px;
  margin: 20px 0 10px;
}

h4, .h4 {
  font-size: 22px;
  line-height: 30px;
}

h5, .h5 {
  font-size: 16px;
  margin-bottom: 15px;
}

h6, .h6 {
  font-size: 14px;
  font-weight: 600;
  text-transform: uppercase;
}
```

Файл sidebar.css

```
.sidebar {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  width: 260px;
  display: block;
  z-index: 1;
  color: #fff;
  font-weight: 200;
}
.sidebar .sidebar-wrapper {
  position: relative;
  max-height: none;
  min-height: 100%;
  overflow: hidden;
  width: 260px;
}
.sidebar .sidebar-background {
  position: absolute;
  z-index: 1;
  height: 100%;
  width: 100%;
  display: block;
  top: 0;
  left: 0;
}
.sidebar .logo {
  padding: 10px 15px;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
}
.sidebar .logo p {
  float: left;
  font-size: 20px;
  margin: 10px 10px;
  color: #FFFFFF;
}
.sidebar .nav {
  margin-top: 20px;
}
.sidebar .nav li > a {
  color: #FFFFFF;
  margin: 5px 15px;
  opacity: .86;
  border-radius: 4px;
}
.sidebar .nav li:hover > a {
  background: rgba(255, 255, 255, 0.13);
  opacity: 1;
}
.sidebar .nav li.active > a {
  color: #FFFFFF;
  opacity: 1;
  background: rgba(255, 255, 255, 0.23);
}
.sidebar .nav p {
  margin: 0;
  line-height: 30px;
  font-size: 12px;
  font-weight: 600;
  text-transform: uppercase;
}
.sidebar .nav i {
  font-size: 28px;
  float: left;
```

```
  margin-right: 15px;
  line-height: 30px;
  width: 30px;
  text-align: center;
}

.sidebar .logo,
body > .navbar-collapse .logo {
  padding: 10px 15px;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
}
.sidebar .logo p,
body > .navbar-collapse .logo p {
  float: left;
  font-size: 20px;
  margin: 10px 10px;
  color: #FFFFFF;
  line-height: 20px;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
}
.sidebar .logo .simple-text,
body > .navbar-collapse .logo .simple-text {
  text-transform: uppercase;
  padding: 5px 0px;
  display: block;
  font-size: 18px;
  color: #FFFFFF;
  text-align: center;
  font-weight: 400;
  line-height: 30px;
}
```

Файл main.css

```
.wrapper {
  position: relative;
  top: 0;
  height: 100vh;
}


.main-panel {
  background: rgba(203, 203, 210, 0.15);
  position: relative;
  z-index: 2;
  float: right;
  width: calc(100% - 260px);
  min-height: 100%;
}
.main-panel > .content {
  padding: 30px 15px;
  min-height: calc(100% - 123px);
}
.main-panel > .footer {
  border-top: 1px solid #e7e7e7;
}
.main-panel .navbar {
  margin-bottom: 0;
}

.sidebar,
.main-panel {
```

```
  overflow: auto;
  max-height: 100%;
  height: 100%;
  -webkit-transition-property: top,bottom;
  transition-property: top,bottom;
  -webkit-transition-duration: .2s,.2s;
  transition-duration: .2s,.2s;
  -webkit-transition-timing-function: linear,linear;
  transition-timing-function: linear,linear;
  -webkit-overflow-scrolling: touch;
}
```

Файл btn.css

```
.btn {
  border-width: 2px;
  background-color: transparent;
  font-weight: 400;
  opacity: 0.8;
  filter: alpha(opacity=80);
  padding: 8px 16px;
  border-color: #888888;
  color: #888888;
}
.btn:hover, .btn:focus, .btn:active, .btn.active, .open > .btn.dropdown-toggle {
  background-color: transparent;
  color: #777777;
  border-color: #777777;
}

.btn.btn-fill {
  color: #FFFFFF;
  background-color: #888888;
  opacity: 1;
  filter: alpha(opacity=100);
}
.btn.btn-fill:hover, .btn.btn-fill:focus, .btn.btn-fill:active, .btn.btn-fill.active,
.open > .btn.btn-fill.dropdown-toggle {
  background-color: #777777;
  color: #FFFFFF;
}
.btn.btn-fill .caret {
  border-top-color: #FFFFFF;
}
.btn .caret {
  border-top-color: #888888;
}
.btn:hover, .btn:focus {
  opacity: 1;
  filter: alpha(opacity=100);
  outline: 0 !important;
}
.btn:active, .btn.active, .open > .btn.dropdown-toggle {
  -webkit-box-shadow: none;
  box-shadow: none;
  outline: 0 !important;
}
.btn.btn-icon {
  padding: 8px;
}

.btn-primary {
  border-color: #3472F7;
```

```
  color: #3472F7;
}
.btn-primary:hover, .btn-primary:focus, .btn-primary:active, .btn-primary.active, .open >
.btn-primary.dropdown-toggle {
  background-color: transparent;
  color: #1D62F0;
  border-color: #1D62F0;
}

.btn-primary.btn-fill {
  color: #FFFFFF;
  background-color: #3472F7;
  opacity: 1;
  filter: alpha(opacity=100);
}
.btn-primary.btn-fill:hover, .btn-primary.btn-fill:focus, .btn-primary.btn-fill:active,
.btn-primary.btn-fill.active, .open > .btn-primary.btn-fill.dropdown-toggle {
  background-color: #1D62F0;
  color: #FFFFFF;
}
.btn-primary.btn-fill .caret {
  border-top-color: #FFFFFF;
}
.btn-primary .caret {
  border-top-color: #3472F7;
}

.btn-success {
  border-color: #87CB16;
  color: #87CB16;
}
.btn-success:hover, .btn-success:focus, .btn-success:active, .btn-success.active, .open >
.btn-success.dropdown-toggle {
  background-color: transparent;
  color: #049F0C;
  border-color: #049F0C;
}

.btn-success.btn-fill {
  color: #FFFFFF;
  background-color: #87CB16;
  opacity: 1;
  filter: alpha(opacity=100);
}
.btn-success.btn-fill:hover, .btn-success.btn-fill:focus, .btn-success.btn-fill:active,
.btn-success.btn-fill.active, .open > .btn-success.btn-fill.dropdown-toggle {
  background-color: #049F0C;
  color: #FFFFFF;
}
.btn-success.btn-fill .caret {
  border-top-color: #FFFFFF;
}
.btn-success .caret {
  border-top-color: #87CB16;
}

.btn-info {
  border-color: #1DC7EA;
  color: #1DC7EA;
}
.btn-info:hover, .btn-info:focus, .btn-info:active, .btn-info.active, .open > .btn-
info.dropdown-toggle {
  background-color: transparent;
  color: #42d0ed;
  border-color: #42d0ed;
```

```
}

.btn-info.btn-fill {
  color: #FFFFFF;
  background-color: #1DC7EA;
  opacity: 1;
  filter: alpha(opacity=100);
}
.btn-info.btn-fill:hover, .btn-info.btn-fill:focus, .btn-info.btn-fill:active, .btn-
info.btn-fill.active, .open > .btn-info.btn-fill.dropdown-toggle {
  background-color: #42d0ed;
  color: #FFFFFF;
}
.btn-info.btn-fill .caret {
  border-top-color: #FFFFFF;
}
.btn-info .caret {
  border-top-color: #1DC7EA;
}
```

Файл table.css

```
.table .radio,
.table .checkbox {
  position: relative;
  height: 20px;
  display: block;
  width: 20px;
  padding: 0px 0px;
  margin: 0px 5px;
  text-align: center;
}
.table .radio .icons,
.table .checkbox .icons {
  left: 5px;
}
.table > thead > tr > th,
.table > tbody > tr > th,
.table > tfoot > tr > th,
.table > thead > tr > td,
.table > tbody > tr > td,
.table > tfoot > tr > td {
  padding: 12px 8px;
  vertical-align: middle;
}
.table > thead > tr > th {
  border-bottom-width: 1px;
  font-size: 12px;
  text-transform: uppercase;
  color: #9A9A9A;
  font-weight: 400;
  padding-bottom: 5px;
}
.table .td-actions .btn {
  opacity: 0.36;
  filter: alpha(opacity=36);
}
.table .td-actions .btn.btn-xs {
  padding-left: 3px;
  padding-right: 3px;
}
.table .td-actions {
  min-width: 90px;
```

```
}
.table > tbody > tr {
  position: relative;
}
.table > tbody > tr:hover .td-actions .btn {
  opacity: 1;
  filter: alpha(opacity=100);
}
```

Файл navbar.css

```
.nav > li > a:hover,
.nav > li > a:focus {
  background-color: transparent;
}

.navbar {
  border: 0;
  font-size: 16px;
  border-radius: 0;
}
.navbar .navbar-brand {
  font-weight: 400;
  margin: 5px 0px;
  padding: 15px 15px;
  font-size: 20px;
}
.navbar .navbar-nav > li > a {
  padding: 10px 15px;
  margin: 10px 3px;
  position: relative;
}
.navbar .navbar-nav > li > a.btn {
  margin: 15px 3px;
  padding: 8px 16px;
}
.navbar .navbar-nav > li > a.btn-round {
  margin: 16px 3px;
}
.navbar .navbar-nav .notification {
  position: absolute;
  background-color: #FB404B;
  text-align: center;
  border-radius: 10px;
  min-width: 18px;
  padding: 0 5px;
  height: 18px;
  font-size: 12px;
  color: #FFFFFF;
  font-weight: bold;
  line-height: 18px;
  top: 0px;
  left: 7px;
}
.navbar .btn {
  margin: 15px 3px;
  font-size: 14px;
}
.navbar .btn-simple {
  font-size: 16px;
}
.navbar.fixed {
  width: calc(100% - $sidebar-width);
```

```
  right: 0;
  left: auto;
  border-radius: 0;
}


.navbar-nav > li > .dropdown-menu {
  border-radius: 10px;
  margin-top: -5px;
}
```

Файл footer.css

```
.footer {
  background-color: #FFFFFF;
  line-height: 20px;
}
.footer nav > ul {
  list-style: none;
  margin: 0;
  padding: 0;
  font-weight: normal;
}
.footer nav > ul a:not(.btn) {
  color: #9A9A9A;
  display: block;
  margin-bottom: 3px;
}
.footer nav > ul a:not(.btn):hover, .footer nav > ul a:not(.btn):focus {
  color: #777777;
}
.footer hr {
  border-color: #DDDDDD;
}
.footer .title {
  color: #777777;
}

.footer-default {
  background-color: #F5F5F5;
}
```

Файл card.css

```
.card {
  border-radius: 4px;
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.05), 0 0 0 1px rgba(63, 63, 68, 0.1);
  background-color: #FFFFFF;
  margin-bottom: 30px;
}
.card .image img {
  width: 100%;
}
.card .filter {
  position: absolute;
  background-color: rgba(0, 0, 0, 0.68);
  top: 0;
  left: 0;
  width: 100%;
```

```
  height: 100%;
  text-align: center;
  opacity: 0;
}
.card .btn-hover {
  opacity: 0;
  filter: alpha(opacity=0);
}
.card:hover .btn-hover {
  opacity: 1;
  filter: alpha(opacity=100);
}
.card .content {
  padding: 15px 15px 10px 15px;
}
.card .header {
  padding: 15px 15px 0;
}
.card .category,
.card label {
  font-size: 14px;
  font-weight: 400;
  color: #9A9A9A;
  margin-bottom: 0px;
}
.card .category i,
.card label i {
  font-size: 16px;
}
.card label {
  font-size: 12px;
  margin-bottom: 5px;
}
.card .title {
  margin: 0;
  color: #333333;
  font-weight: 300;
}
.card .avatar {
  width: 30px;
  height: 30px;
  overflow: hidden;
  border-radius: 50%;
  margin-right: 5px;
}
.card .description {
  font-size: 14px;
  color: #333;
}
.card .footer {
  padding: 0;
  background-color: transparent;
  line-height: 30px;
}
.card .footer .legend {
  padding: 5px 0;
}
.card .footer hr {
  margin-top: 5px;
  margin-bottom: 5px;
}
.card .stats {
  color: #a9a9a9;
}
.card .footer div {
```

```
  display: inline-block;
}
.card .author {
  font-size: 12px;
  font-weight: 600;
  text-transform: uppercase;
}
.card .author i {
  font-size: 14px;
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРИЛОЖЕНИЯ

Файл index.js

```
import React from 'react';
import { render, findDOMNode } from 'react-dom';
import { Router, Route, Link } from 'react-router';
import { createHistory, createHashHistory, useBasename } from 'history';

import Sidebar from './components/sidebar';
import Navbar from './components/navbar';
import Overview from './components/overview';
import Tasks from './components/tasks';
import Calendar from './components/calendar';
import Members from './components/members';
import Settings from './components/settings';

const App = React.createClass({
	getInitialState() {
		return {user:{firstname: '', lastname: ''}, navOpen: false};
	},

	componentDidMount() {
		$.ajax({
			url: '/projman-git/auth/get_user',
			dataType: "json",
			type: "POST",
			success: (data, textStatus) => {
				this.setState({user: data});
			}
		});
	},

	handleToggleClicked(e) {
		this.setState({navOpen: !this.state.navOpen});
	},

	handleToggleClickedOutside(e) {
		if (this.state.navOpen) {this.setState({navOpen: false})}
	},

	render() {
		const { content } = this.props;
		let toggleClass = this.state.navOpen ? ' nav-open' : '';
		let className = "wrapper" + toggleClass;
		return (
			<div className={className}>
				<Sidebar projectId={this.props.params.id}
user={this.state.user} navOpen={this.state.navOpen}/>
				{/*<Breadcrumb/>*/}
				<div className="main-panel">
					<Navbar handleToggleClicked={this.handleToggleClicked}
handleToggleClickedOutside={this.handleToggleClickedOutside}/>
					<div className="content">
						<div className="container-fluid">
							{content}
						</div>
					</div>
				</div>
			</div>
		)
	}
```

```
});

const history = useBasename(createHistory)({
	basename: '/projman-git/project/'
})

render((
	<Router history={history}>
		<Route path="/" component={App}>
			<Route path="/view/:id" components={{ content: Overview }} />
			<Route path="/tasks/:id" components={{ content: Tasks }} />
			<Route path="/calendar/:id" components={{ content: Calendar }} />f
			<Route path="/members/:id" components={{ content: Members }} />
			<Route path="/settings/:id" components={{ content: Settings }} />
		</Route>
	</Router>
), document.getElementById('app'))
```

Файл calendar.js

```
import React from 'react';
import { findDOMNode } from 'react-dom';

const CalendarWrap = React.createClass({
	render: function() {
		// don't render anything, this is where we open the portal
		return <div/>;
	},

	componentDidMount: function() {
		var projectId = this.props.projectId;
		// do the old-school stuff
		var dialog = $(findDOMNode(this)).fullCalendar({
			aspectRatio: 1.78,
			firstDay: 1,
			eventLimit: true,
			culture: 'uk',
			format: 'L',
			events: function (start, end, tiemzone, callback) {
				$.ajax({
					url: '/projman-git/project/getEvents/' + projectId,
					dataType: "json",
					type: "POST",
					success: function (data, textStatus) {
						var events = [];
						for (var i = 0; i < data.length; i++) {
							events.push({
								title: data[i].description,
								start: data[i].timestart,
								end: data[i].timedue
							});
						}
						callback(events);
					}
				});
			},
			eventClick: function(calEvent, jsEvent, view) {
			}
		});
	}
});
```

```
export default class Calendar extends React.Component {
	render() {
		return (
			<div className="row">
				<div className="col-md-12">
					<div className="card card-calendar" style={{maxWidth:
1420, margin: '0 auto'}}>
						<CalendarWrap projectId={this.props.params.id}/>
					</div>
				</div>
			</div>
		);
	}
}
```

Файл overview.actions.js

```
import alt from '../alt';

class OverviewActions {
	updateProjectInfo(projectinfo) {
		if (!projectinfo && !projectinfo.length)
			projectinfo = [];
			return projectinfo;

	}

	updateRecentEvents(events) {
		return events;
	}

	setLoading() {
		return true;
	}

	unsetLoading() {
		return true;
	}

	reset() {
		return true;
	}

	fetchProjectInfo(id) {
		return (dispatch) => {
			getProjectInfo(id);
		}
	}

	fetchRecentEvents(id) {
		return (dispatch) => {
			getProjectEvents(id);
		}
	}
}

export default alt.createActions(OverviewActions);
```

Файл overview.store.js

```
import alt from '../alt';
```

```
import OverviewActions from '../actions/cart.actions';

class OverviewStore {
	constructor() {
		this.projectInfo = {};
		this.events = {};
		this.loading = true;
		this.bindListeners({
			handleUpdateProjectInfo: OverviewActions.HANDLE_UPDATE_PROJECT_INFO,
			handleUpdateRecentEvents:
OverviewActions.HANDLE_UPDATE_RECENT_EVENTS,
			handleSetLoading: OverviewActions.HANDLE_SET_LOADING,
			handleUnsetLoading: OverviewActions.HANDLE_UNSET_LOADING
		});
		storage.getMoneyDeposited( (err, docs) => {
			this.moneyDeposited = docs[0].money_deposited;
		});
	}

	handleUpdateRecentEvents(events) {
		if (checkIfDefined) {
			this.events = events;
		}
	}

	handleUpdateProjectInfo(info) {
		if (checkIfDefined) {
			this.info = info;
		}
	}

}

export default alt.createStore(OverviewStore, 'OverviewStore');
```

Файл tasks.actions.js

```
import alt from '../alt';

class TasksActions {
	updateTasks(tasks) {
		return projectinfo;

	}

	changeTaskStatus(status) {
		return status;
	}

	setLoading() {
		return true;
	}

	unsetLoading() {
		return true;
	}

	addTask(id, taskInfo) {
		return taskInfo;
	}

	deleteTask(id) {
		return true;
```

```
	}

	addTaskList(id, tasklistInfo) {

	}

	deleteTaskList(id) {
		return true;
	}


	fetchTasks(id) {
		return (dispatch) => {
			getProjectInfo(id);
		}
	}

	fetchTaskslists(id) {
		return (dispatch) => {
			getProjectEvents(id);
		}
	}
}

export default alt.createActions(TasksActions);
```

Файл member.actions.js

```
import alt from '../alt';

class MemberActions {
	updateMembers(members) {
		return members;

	}

	changeMemberStatus(id, status) {
		return {
			id,
			status
		};
	}

	setLoading() {
		return true;
	}

	unsetLoading() {
		return true;
	}



	fetchMembers(id) {
		return (dispatch) => {
			getMembers(id);
		}
	}

}

export default alt.createActions(MemberActions);
```

Файл settings.actions.js

```
import alt from '../alt';

class SettingsActions {
	updateSettings(settings) {
		return settings;

	}

	changeSettings(id, settings) {
		return {
			id,
			settings
		};
	}

	setLoading() {
		return true;
	}

	unsetLoading() {
		return true;
	}



	fetchSettings(id) {
		return (dispatch) => {
			getSettings(id);
		}
	}

}

export default alt.createActions(SettingsActions);
```

Файл navbar.js

```
import React from 'react';
import { Router, Route, Link } from 'react-router';
const enhanceWithClickOutside = require('react-click-outside');

class CustomMenu extends React.Component {
	constructor(...args) {
		super(...args);
		this.state = { value: '' };
	}

	render() {
		return (
			<div
				className={"dropdown-menu"}
				style={{ padding: 0}}>
				<div className="user-info-content">
					<div className="row">
						<div className="col-md-4 col-xs-4">
							<img src="" alt="" className="img-
responsive img-circle center-block" style={{width: '80px'}}/>
						</div>
						<div className="col-md-8 col-xs-8">
							<span> {this.props.user.firstname + " " +
this.props.user.lastname} </span>
							<p className="text-muted small">
{this.props.user.email} </p>
```

```
                                                <div className="divider"></div>
                                                      <a href="/projman-
git/home/settings" className="btn btn-default btn-sm pull-right">Настройки</a>
                                                </div>
                                          </div>
                                    </div>
                                    <div className="user-info-footer-content">
                                          <div className="row">
                                                <div className="col-md-8 col-md-offset-4 col-xs-
8">
                                                <a href="/projman-git/auth/logout"
className="btn btn-info btn-sm pull-right">Выйти</a>
                                                </div>
                                          </div>
                                    </div>
                              </div>
                        );
                  }
}

const Navbar = React.createClass ({
      handleClickOutside() {
            this.props.handleToggleClickedOutside();
      },

      render() {
            const { projectId } = this.props;
            return (
                  <nav className="navbar navbar-default navbar-fixed">
                        <div className="container-fluid">
                              <div className="navbar-header">
                                    <button type="button" className="navbar-toggle"
data-toggle="collapse" data-target="#navigation-example-2"
onClick={this.props.handleToggleClicked}>
                                          <span className="sr-only">Toggle
navigation</span>
                                          <span className="icon-bar" />
                                          <span className="icon-bar" />
                                          <span className="icon-bar" />
                                    </button>
                                    <a className="navbar-brand"
href="">Dashboard</a>
                              </div>
                              <div className="collapse navbar-collapse">
                                    <ul className="nav navbar-nav navbar-right">
                                          <li><a href="/projman-
git/home/settings">Account</a></li>
                                          <li><a href="/projman-git/home/signout"><i
className="pe-7s-power"/> Sign out</a></li>
                                    </ul>
                              </div>
                        </div>
                  </nav>
            );
      }
});

export default enhanceWithClickOutside(Navbar);
```

Файл sidebar.js

```
import React from 'react';
import { Router, Route, Link } from 'react-router';
```

```
export default class Sidebar extends React.Component {
	constructor(props) {
		super(props);

		this.state = {
			full: false
		}
	}
	componentWillReceiveProps(nextProps) {
		let toggle = () => {this.setState({full: nextProps.navOpen})};
		let timeout = 0;
		if (!nextProps.navOpen) timeout = 300;
		setTimeout(toggle, timeout);
	}
	render() {
		const { projectId } = this.props;
		let nav = <div className="side-nav" style={{borderTop: '1px solid rgba(255,
255, 255, 0.2)'}}>
				<li>
					<a href="/projman-git/home/settings"
activeClassName="active">
						<i className="pe-7s-user" />
						<p>Account</p>
					</a>
				</li>
				<li>
					<a href="/projman-git/home/signout"
activeClassName="active">
						<i className="pe-7s-power" />
						<p>Sign out</p>
					</a>
				</li></div>;
		return (
			<div className="sidebar" data-color="azure" data-image="/light-
bootstrap-dashboard/assets/img/sidebar-5.jpg">
				<div className="sidebar-wrapper">
					<div className="logo">
						<a href="" className="simple-text">Projectr</a>
					</div>
					<ul className="nav">
						<li>
							<Link to={`/view/${projectId}`}
activeClassName="active">
								<i className="pe-7s-graph" />
								<p>Dashboard</p>
							</Link>
						</li>
						<li>
							<Link to={`/tasks/${projectId}`}
activeClassName="active">
								<i className="pe-7s-note2" />
								<p>Tasks</p>
							</Link>
						</li>
						<li>
							<Link to={`/calendar/${projectId}`}
activeClassName="active">
								<i className="pe-7s-date" />
								<p>Calendar</p>
							</Link>
						</li>
						<li>
							<Link to={`/members/${projectId}`}
activeClassName="active">
```

```
                                                        <i className="pe-7s-users" />
                                                        <p>Members</p>
                                                 </Link>
                                          </li>
                                          <li>
                                                 <Link to={`/settings/${projectId}`}
activeClassName="active">
                                                        <i className="pe-7s-tools" />
                                                        <p>Settings</p>
                                                 </Link>
                                          </li>
                                          {this.state.full ? nav : ''}
                                          <li className="active active-pro">
                                                 <a href="">
                                                        <i className="pe-7s-rocket" />
                                                        <p>Upgrade to PRO</p>
                                                 </a>
                                          </li>
                                   </ul>
                            </div>
                            <div className="sidebar-background" style={{backgroundImage:
'url(/projman-git/modules/img/sidebar-5.jpg)'}} />
                     </div>

              );
       }
}
```

Файл members.js

```
import React from 'react';
import { Table, Button, Modal, ButtonInput, DropdownButton, MenuItem, Dropdown } from
'react-bootstrap';

export default class Members extends React.Component {
       constructor(props) {
              super(props);
              this.state = {
                     showModal: false, users: [], selectedRows: [], showDelButton: false
              }
       }

       fetchUsers(cb) {
              const projectId = this.props.params.id;
              $.get('/projman-git/project/users_json/' + projectId, cb);
       }

       componentDidMount() {
              this.fetchUsers(function (result) {
                     result.users_in_proj = result.users_in_proj || {};
                     this.setState({
                            users: result.users_in_proj
                     });
              }.bind(this));
       }

       closeModal() {
              this.setState({showModal: false});
       }

       openModal() {
              this.setState({showModal: true});
       }
```

```
	onChange(e) {
		this.setState({
			username: e.target.value
		});
	}

	handleSubmit(e) {
		e.preventDefault();
		const projectId = this.props.params.id;
		$.ajax({
			method: "POST",
			url: "/projman-git/project/adduser",
			data: "project_id=" + projectId + "&username=" + this.state.username
		}).done(( msg ) => {
			this.closeModal();
			this.fetchUsers(function (result) {
				this.setState({
					users: result.users_in_proj
				});
			}.bind(this));
		});
	}

	handleRowClick(e) {
		// this.setState({cl: this.state.cl === '' ? 'selected': ''});
		var index = this.state.selectedRows.indexOf(e.currentTarget.rowIndex - 1);
		if (index == -1)
			this.state.selectedRows.push(e.currentTarget.rowIndex - 1);
		else {
			this.state.selectedRows.splice(index, 1);
		}
		if (this.state.selectedRows.length) this.setState({showDelButton: true});
		else this.setState({showDelButton: false});
	}

	handleRemoveUsers(e) {
		const projectId = this.props.params.id;
		if (!confirm('Are you sure?')) return;

		var userNames = [];
		for (var i = 0; i < this.state.selectedRows.length; i++) {

	userNames.push(this.state.users[this.state.selectedRows[i]].username);
		}

		$.ajax({
			url: '/projman-git/project/removeuser',
			type: "POST",
			data: {
				"project_id": projectId,
				"usernames": userNames
			},
			success: (data, textStatus) => {
				this.setState({showDelButton: false});
				this.fetchUsers(function (result) {
					this.setState({
						users: result.users_in_proj
					});
				}.bind(this));
			}
		});
	}

	render() {
```

```
            var userNodes = this.state.users.map((user, index) => {
                  let cl = this.state.selectedRows.indexOf(index) == - 1 ? '' :
'selected'
                  return (
                        <tr key={index + 1} className={cl}
onClick={this.handleRowClick.bind(this)} style={{cursor: 'pointer'}}>
                              <td>{index + 1}</td>
                              <td>{user.firstname}</td>
                              <td>{user.lastname}</td>
                              <td>{user.email}</td>
                        </tr>
                  );
            });
            var cl = this.state.showDelButton ? '' : 'hidden';
            cl += " btn btn-danger";
            return (
                  <div className="">
                        <div className="row">
                              <div className="col-md-12" style={{'marginBottom':
20}}>
                                    <button type="button" className="btn btn-
primary" onClick={this.openModal.bind(this)}> Добавить <span><i className="fa fa-user-
plus"></i></span></button>
                                    <button type="button" id="del" className={cl}
onClick={this.handleRemoveUsers.bind(this)} style={{marginLeft: 20}}>
                                          Delete <i className="fa fa-user-
times"></i>
                                    </button>
                              </div>
                              <Modal show={this.state.showModal}
onHide={this.closeModal.bind(this)}>
                                    <Modal.Header closeButton>
                                          <Modal.Title><h4 className="modal-title"
id="myModalLabel"> Добавить пользователя </h4></Modal.Title>
                                    </Modal.Header>
                                    <form id="adduserform" method="post"
onSubmit={this.handleSubmit.bind(this)}>
                                          <Modal.Body>
                                                <h4> Имя пользователя </h4>
                                                <div className="input-group">
                                                      <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-user"></i></span>
                                                      <input type="text"
id="usernameinput" name="username" className="form-control" placeholder="например Bob
Blob " value={this.state.username} onChange={this.onChange.bind(this)} aria-
describedby="basic-addon1" autoFocus/>
                                                </div>
                                          </Modal.Body>
                                          <Modal.Footer>
                                                <Button
onClick={this.closeModal.bind(this)}>Отмена</Button>
                                                <input type="submit" className="btn
btn-primary" value="Добавить"/>
                                          </Modal.Footer>
                                    </form>
                              </Modal>
                        </div>
                        <div className="row" >
                              <div className="col-md-12">
                                    <div className="card">
                                          <div className="col-md-12">
                                                <div className="header">

                                                      <h4
className="title">Members</h4>
```

```
                                                        </div>
                                                </div>
                                                <table className="table table-hover table-
striped">
                                                        <thead>
                                                                <tr>
                                                                        <th className="col-
md-3" data-field="id">#</th>
                                                                        <th className="col-
md-3" data-field="id">Имя</th>
                                                                        <th className="col-
md-3" data-field="name">Фамилия</th>
                                                                        <th className="col-
md-3" data-field="name">E-mail </th>
                                                                </tr>
                                                        </thead>
                                                        <tbody>
                                                                {userNodes}
                                                        </tbody>
                                                </table>
                                        </div>
                                </div>
                        </div>
                </div>
        );
        }

}
```

Файл settings.js

```
import React from 'react';
import DatePicker from 'react-datepicker';
import update from 'react-addons-update';

const Settings = React.createClass({
        getInitialState() {
                return {
                        title: '',
                        project_info: {},
                        checked: false
                };
        },
        componentDidMount() {
                const projectId = this.props.params.id;
                $.get(`/projman-git/project/settings_json/${projectId}`, function(res) {
                        res.checked = res.project_info.default_project ==
res.project_info.id;
                        res.project_info.timestart = moment(res.project_info.timestart,
"DD.MM.YYYY");
                        res.project_info.timedue = moment(res.project_info.timedue,
"DD.MM.YYYY");
                        this.setState(res);
                }.bind(this));
        },

        handleChange(e) {
                var newState = update(this.state, {
                        project_info: {
                                description: {$set: e.target.value}
                        }
                });
                this.setState(newState);
```

```
	},

	handleChecked(e) {
		var newState = update(this.state, {
			checked: {$set: e.target.checked}
		});
		this.setState(newState);
	},

	handleSubmit(e) {
		e.preventDefault();
		const projectId = this.props.params.id;

		$.ajax({
			method: "POST",
			url: "/projman-git/project/change_project_settings/" + projectId,
			data: "project_name=" + this.state.title + "&description=" +
this.state.project_info.description +
				"&manager=" + this.state.project_info.manager + "&timestart="
+ this.state.project_info.timestart.format("DD.MM.YYYY") +
				"&timedue=" +
this.state.project_info.timedue.format("DD.MM.YYYY") + "&default_project=" +
this.state.checked + "&project_id=" + projectId
		})
			.done(( msg ) => {
			});

	},

	handleDeleteProject(e) {
		e.preventDefault();
		const projectId = this.props.params.id;
		if (!confirm('Are you sure you want to delete?')) return;
		$.ajax({
			url: '/projman-git/home/removeproject',
			type: "POST",
			data: "project_id=" + projectId,
			success: function (data, textStatus) {
				window.document.location = '/projman-git/home';
			}
		});
	},

	handleLeaveProject(e) {
		e.preventDefault();
		if (!confirm('Are you sure you want to laeve this project?')) return;
		const projectId = this.props.params.id;
		$.ajax({
			url: '/projman-git/project/leave_project/' + projectId,
			type: "POST",
			data: "project_id=" + projectId,
			success: function (data, textStatus) {
				window.location = '/projman-git/home';
			}
		});
	},

	handleTimeStartChange(date) {
		var newState = update(this.state, {
			project_info: {
				timestart: {$set: date}
			}
		});
		this.setState(newState);
	},
```

```
	handleTimeDueChange(date) {
		var newState = update(this.state, {
			project_info: {
				timedue: {$set: date}
			}
		});
		this.setState(newState);
	},

	render() {
		const projectId = this.props.params.id;
		return (
			<main className="">
				<div className="row">
					<div className="col-md-12">
						<div className="card">
							<div className="header">
								<h4 className="title">Общая
информация</h4>
								<hr/>
								{/*<p className="category">Last
Campaign Performance</p>*/}
							</div>
							<div className="content">
								<h5 style={{marginTop: 0}}>
Название проекта </h5>
								<input id="project-name"
className="form-control" value={this.state.title} type="text"/>
								<h5> Руководитель </h5>
								<input id="manager"
className="form-control" value={this.state.project_info.manager} type="text"/>
								<h5> Временные рамки </h5>
								<div className="col-md-6">
									<DatePicker

	selected={this.state.project_info.timestart}

	dateFormat="DD.MM.YYYY"

	onChange={this.handleTimeStartChange} />
								</div>
								<div className="col-md-6"
style={{marginBottom: 10}}>
									<DatePicker

	selected={this.state.project_info.timedue}
											dateFormat=
"DD.MM.YYYY"

	minDate={this.state.project_info.timestart}

	onChange={this.handleTimeDueChange} />
								</div>
								<h5> Описание </h5>
								<textarea className="form-control"
name="description" id="" cols="30" rows="10" value={this.state.project_info.description}
onChange={this.handleChange}></textarea>
								<h5> Проект по умолчанию</h5>
								<input type="checkbox"
name="default_project" onClick={this.handleChecked} checked={this.state.checked}/>
								<span> Заходить в этот проект при
входе в систему</span>
							</div>
					</div>
```

```
                                </div>
                            </div>
                            <div className="row" style={{marginBottom: 20}}>
                                <div className="col-md-12">
                                    <button type="submit" className="submit-button
btn btn-lg btn-primary"> Применить </button>
                                </div>
                            </div>
                            <div className="row">
                                <div className="col-md-12">
                                    <div className="card">
                                        <div className="header">
                                            <h4 className="title"><div
className="alert alert-danger">Опасная зона</div></h4>
                                            {/*<p className="category">Last
Campaign Performance</p>*/}
                                        </div>
                                        <div className="content">
                                            <form id="deleteproject-form"
role="form" onSubmit={this.handleDeleteProject} style={{display: 'inline-block',
marginRight: 30}}>
                                                <button type="submit"
className="btn btn-lg btn-danger"> Удалить проект </button>
                                            </form>
                                            <form id="leaveproject-form"
role="form" onSubmit={this.handleLeaveProject} style={{display: 'inline-block'}}>
                                                <button type="submit"
className="btn btn-lg btn-danger"> Покинуть проект </button>
                                            </form>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </main>
                );
        }
});

export default Settings;
```

Файл tasks.js

```
import React from 'react';
import { Router, Route, Link } from 'react-router';
import { Button, Modal } from 'react-bootstrap';

import DatePicker from 'react-datepicker';

const Tasks = React.createClass({
        getInitialState() {
                return  {
                        title: "",
                        tasklists: [],
                        showModal: false,
                        showDelButton: false
                };
        },

        openModal() {
                this.setState({showModal: true});
        },

        handleSubmit(){
```

```
	},

	onXClick(e){

	},

	closeModal() {
		this.setState({showModal: false});
	},

	onTaskDescriptionChange(e) {
		this.setState({
			taskdescription: e.target.value
		});
	},
	onSupervisorChange(e) {
		this.setState({
			supervisor: e.target.value
		});
	},
	onImplementerChange(e) {
		this.setState({
			implementer: e.target.value
		});
		console.log(this.state.implementer);
	},
	onTasklistSelectChange(e) {
		this.setState({
			tasklistselect: e.target.value
		});
		console.log(this.state.tasklistselect);
	},
	componentDidMount() {
		const projectId = this.props.params.id;
		this.serverRequest = $.get('/projman-git/project/tasks_json/' + projectId,
(result) => {
			var title = result.title;
			var tasklists = result.tasklists;
			console.log(result);
			this.setState({
				title: title,
				tasklists: tasklists
			});
		});
	},

	handleTimeStartChange(date) {
		this.setState({
			time_start: date
		});
	},
	handleTimeDueChange(date) {
		this.setState({
			time_due: date
		});
	},


	render() {
		return (
		<div className="row">
			<div className="col-md-12">
				<div className="page-header" style={{marginTop: '-15px'}}>
					<h1>Задания</h1>
```

```
                        </div>
                    </div>

                    <div className="dropdown">
                        <button type="button" className="btn btn-primary btn-sm" data-
toggle="dropdown" onClick={this.openModal}>
                            Добавить <i className="fa fa-pencil-square-o"></i><span
className="caret"></span>
                        </button>
                        <ul className="dropdown-menu" aria-labelledby="dropdownMenu1">
                            <li><a href="#" data-toggle="modal" data-
target="#addtask">Новое задание</a></li>
                            <li><a href="#" data-toggle="modal" data-
target="#addtasklist">Новый список заданий</a></li>
                        </ul>
                    </div>
                    <Modal show={this.state.showModal} onHide={this.closeModal}>
                        <Modal.Header closeButton>
                            <Modal.Title><h4 className="modal-title"
id="myModalLabel"> Добавить задание </h4></Modal.Title>
                        </Modal.Header>


                                    <form id="addtaskform" method="post"
onSubmit={this.handleSubmit}>
                                        <Modal.Body>
                                            <h4>Описание</h4>
                                            <div className="input-group">
                                                <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-pencil-square-o"></i></span>
                                                <input name="task_desc"
className="form-control" placeholder="$language['taskdescription']?>" aria-
describedby="basic-addon1" type="text" value={this.state.taskdescription}
onChange={this.onTaskDescriptionChange} />
                                            </div>
                                            <h4>Список заданий</h4>
                                            <div className="input-group">
                                                <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-tasks"></i></span>
                                                    <select
name="tasklist" className="form-control" value={this.state.tasklistselect}
onChange={this.onTasklistSelectChange} >
                                                    {

        this.state.tasklists.map((tasklist, index) => {
                                                                return (

        <option key={index} value={tasklist.id}>

        {tasklist.description}

        </option>
                                                                )
                                                            }
                                                    )}
                                                    </select>
                                            </div>

                                            <h4>Сроки</h4>
                                            <div className="row">
                                                <div className="col-md-6">
                                                    <DatePicker

        selected={this.state.time_start}
```

```
       dateFormat="DD.MM.YYYY"

       onChange={this.handleTimeStartChange} />
                                                              </div>
                                                              <div className="col-md-6">
                                                                     <DatePicker

       selected={this.state.time_due}
                                                                                   dateFormat=
"DD.MM.YYYY"

       minDate={this.state.time_start}

       onChange={this.handleTimeDueChange} />
                                                              </div>

                                                       </div>
                                                       <h4>Руководитель</h4>
                                                       <div className="input-group">
                                                              <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-user"></i></span>
                                                              <input name="supervisor"
className="form-control" placeholder="$language['egbobblob']?>" aria-describedby="basic-
addon1" type="text" value={this.state.supervisor} onChange={this.onSupervisorChange}  />
                                                       </div>
                                                       <h4>Исполнитель</h4>
                                                       <div className="input-group">
                                                              <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-user"></i></span>
                                                              <input name="implementer"
className="form-control" placeholder="" aria-describedby="basic-addon1" type="text"
value={this.state.implementer} onChange={this.onImplementerChange}   />
                                                       </div>
                                                       <h4>Состояние</h4>
                                                       <div className="input-group">
                                                              <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-check-square-o"></i></span>
                                                       {/*  <input name="status_id"
className="form-control" placeholder="" aria-describedby="basic-addon1" type="text"> */}
                                                              <select name="status_id"
className="form-control">
                                                                     <option
value="1">Ожидание</option>
                                                                     <option
value="2">Отложено</option>
                                                                     <option value="3">В
работе</option>
                                                                     <option
value="4">Проверка</option>
                                                                     <option
value="5">Выполнено</option>
                                                                     <option
value="6">Отменено</option>
                                                              </select>
                                                       </div>
                                                </Modal.Body>
                                                <Modal.Footer>
                                                       <button type="button"
className="btn btn-default" data-dismiss="modal">Закрыть</button>
                                                       <input className="btn btn-primary"
value="$language['submit']?>" type="submit"/>
                                                </Modal.Footer>
                                         </form>
                     </Modal>
              <div style={{display: 'none'}} className="modal fade" id="addtasklist"
```

```
tabIndex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">
                          <div className="modal-dialog ">
                                 <div className="modal-content">
                                        <div className="modal-header">
                                               <button type="button" className="close"
data-dismiss="modal" aria-label="Close"><span aria-hidden="true">?</span>
                                               </button>
                                               <h4 className="modal-title"
id="myModalLabel">Добавить список заданий</h4>
                                        </div>
                                        <form id="addtasklistform" method="POST"
onsubmit="addTaskList($project_id ?>)">
                                               <div className="modal-body">
                                                      <h4>Описание</h4>
                                                      <div className="input-group">
                                                             <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-pencil-square-o"></i></span>
                                                             <input name="tasklist_desc"
className="form-control" placeholder="$language['tasklistdescription']?>" aria-
describedby="basic-addon1" type="text"/>
                                                      </div>
                                                      <h4>Руководитель</h4>
                                                      <div className="input-group">
                                                             <span className="input-
group-addon" id="basic-addon1"><i className="fa fa-user"></i></span>
                                                             <input
name="tasklist_supervisor" className="form-control"
placeholder="$language['egbobblob']?>" aria-describedby="basic-addon1" type="text"/>
                                                      </div>
                                               </div>
                                               <div className="modal-footer">
                                                      <button type="button"
className="btn btn-default" data-dismiss="modal">Закрыть</button>
                                                      <input className="btn btn-primary"
value="$language['submit']?>" type="submit"/>
                                               </div>
                                        </form>
                                 </div>
                          </div>
                   </div>
                   <div className="row" style={{marginTop: '15px'}}>
                          <div className="col-md-12">
                                 <table width="100%">
                                        <tbody>
                                               <tr className="table-header">
                                                      <td className="col-md-
5">Задание</td>
                                                      <td className="col-md-
1">Начало</td>
                                                      <td className="col-md-1">Срок</td>
                                                      <td className="col-md-1">Рук.</td>
                                                      <td className="col-md-
1">Исполнитель</td>
                                                      <td className="col-md-
1">Состояние</td>
                                               </tr>
                                        </tbody>
                                               {
                                                      this.state.tasklists.map((tasklist,
index) => {
                                                        var tasks =
tasklist.tasks.map((task, task_index) => {
                                                             return (
                                                                    <tr className={"task
out accord" + (index + 1)} key={task_index}>
```

```
                                                                    <td
className="col-md-5"><div className="invis delete-button"
onClick={this.onXClick}>×</div>{task.description}</td>
                                                                    <td
className="col-md-1">{task.timestart}</td>
                                                                    <td
className="col-md-1">{task.timedue}</td>
                                                                    <td
className="col-md-1">{task.supervisor}</td>
                                                                    <td
className="col-md-1">{task.implementer}</td>
                                                                    <td
className="col-md-1">
                                                                           <div
className="dropdown task-status">

       <div href="#" className="dropdown-toggle" data-toggle="dropdown" data-taskid={"\""
+ task.id + "\""}>{task.status_id}<span className="caret"></span></div>

       <ul className="dropdown-menu">

       <li><a className="status-changer" href="#">$language['pending'] ?></a>

       </li>

       <li><a className="status-changer" href="#">$language['postponed'] ?></a>

       </li>

       <li><a className="status-changer" href="#">$language['processing'] ?></a>

       </li>

       <li><a className="status-changer" href="#">$language['validating']?></a>

       </li>

       <li><a className="status-changer" href="#">$language['accomplished'] ?></a>

       </li>

       <li><a className="status-changer" href="#">$language['cancelled']?></a>

       </li>

       </ul>
                                                                           </div>
                                                                    </td>
                                                             </tr>
                                                             );
                                                    });

                                                           return (
                                                                <tbody key={index}>
                                                                <tr className="tasklist-
header" data-toggle="collapse" data-target={".accord" + (index + 1)} >

       <td><strong>{tasklist.description + " (" + "supervisedby" + " " +
tasklist.supervisor + ")" }</strong></td>
                                                                    </tr>
                                                                {tasks}
                                                             </tbody>
                                                                    );
                                                     }
                                                    )
```

```
                                                }
                                    </table>
                                </div>
                            </div>
                        </div>
                        );
            }
});

export default Tasks;
```