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	Шахов Владимир 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		

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.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студенту:

Группа	ФИО
8В2А	Шахову Владимиру 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 аналогичных систем, разработка архитектуры системы, выбор инструментов для реализации (языки программирования, фреймворки, библиотеки).
Перечень графического материала	Схема базы данных, скриншоты интерфейса, диаграмма с описанием архитектуры 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	Шахов Владимир Сергеевич		10.09.2015

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
7.10.2015	Изучение материалов по системам управления проектами	5
4.11.2015	Выбор средств реализации	5
2.12.2015	Разработка архитектуры системы	20
3.02.2016	Реализация базового функционала системы	20
6.04.2016	Тестирование, отладка и доработка системы	20
4.05.2016	Завершение работы над системой	15
20.05.2015	Социальная ответственность	5
27.05.2015	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 72 страницы, 6 рисунков, 4 таблицы, 19 источников.

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

Объект исследования работы – управление проектами и системы управления проектами.

Цель работы – разработка системы управления проектами с использованием современных технологий разработки web-приложений, имеющей потенциал для того, чтобы конкурировать с уже существующими системами.

Основные задачи, выполненные в ходе работы, включают в себя исследование существующих аналогов, разработку архитектуры системы, выбор средств разработки, разработку, отладку и тестирование.

В результате работы была разработана система управления проектами с использованием технологий REST и SPA.

Область применения – использование в небольших компаниях для организации делопроизводства или использование в личных целях для организации работы над личными проектами.

Значимость работы заключается в использовании современных технологий и подходов для создания системы, позволяющей пользователям просто и быстро управлять своими проектами.

В дальнейшем планируется расширение функциональных возможностей системы, оптимизация программного кода для увеличения эффективности работы системы.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И НОРМАТИВНЫЕ ССЫЛКИ

1. JSON (JavaScript Object Notation) – формат передачи данных, использующий для структурирования информации синтаксис языка JavaScript[1].

2. JWT (JSON Web Token) – основанный на JSON стандарт для передачи сообщений между двумя сторонами (например, между клиентом и сервером). В работе используется для обеспечения аутентификации пользователей в системе [1].

3. MVC (Model-view-controller) – архитектура организации приложения и способа его взаимодействия с пользователем, при которой система разбивается на три составные части – модели (models), представления (views) и контроллеры (controllers) [2].

4. PHP – язык программирования, широко применяющийся для разработки серверной части web-приложений.

5. REST (Representational State Transfer) – архитектура web-приложения, при которой сервер не хранит состояний взаимодействия с клиентами в явном виде, а вся информация о состоянии передается клиентом в запросе [3].

6. SWOT-анализ (Strengths, Weaknesses, Opportunities, Threats) – метод управления рисками, при котором для организации или ее проекта проводится анализ сильных и слабых сторон, позитивных и негативных рисков, а также возможных ситуаций их наступления. Проводится для предупреждения ситуаций, способных оказать негативное влияние на организацию.

7. QuaD-анализ – метод оценки выполнения проекта, при котором оцениваются его различные показатели (как финансовые, так и технические) и на основании этих показателей формируется общее представление о конкурентоспособности, качестве и степени завершенности проекта, целесообразности дальнейшей разработки.

8. SPA (Single Page Application, одностраничное приложение) – вид организации web-приложений, при котором пользователь загружает всего одну web-страницу, которая затем динамически обновляется при переходе из одного состояния в другое. При этом не возникает необходимости в постоянной загрузке новых страниц, что снижает нагрузку на сеть и обеспечивает лучшее взаимодействие пользователя с системой [4].

9. Контроллер – один из компонентов архитектуры MVC, который отвечает за организацию взаимодействия пользователя и системы, обработку запросов пользователя и выдачу ему запрашиваемой информации, получаемой из моделей, в виде представлений [5].

10. Модель – один из компонентов архитектуры MVC, который отвечает за хранение информации, приобретение информации и передачу ее контроллеру для дальнейшего использования при формировании представлений. Зачастую модель взаимодействует с базой данных [6].

11. Представление – один из компонентов архитектуры MVC, отвечающий за отображение запрашиваемой информации пользователям. Представления формируются на основе информации, получаемой из моделей. Управлением представлениями занимаются контроллеры [7].

12. Проект – ограниченное временными рамками мероприятие, целью которого является создание конечного продукта или услуги. Проект состоит из некоторого количества ограниченных по времени, задач, выполнение которых приведет к достижению цели проекта. В проекте, как правило, могут принимать участие несколько человек [8].

13. СУП – система управления проектами.

14. Токен – информационное сообщение, целью которого является идентификация пользователя в системе. Токены используются для того, чтобы определить на совершение каких действий у пользователя имеются необходимые привилегии.

15. Фреймворк – программный продукт, использующийся для облегчения разработки других программ. Фреймворк представляет из себя

каркас, определяющий архитектуру приложения и предоставляющий ему различные встроенные функции для упрощения разработки. Ключевым отличием фреймворка от библиотеки является тот факт, что код фреймворка является основным по отношению к пользовательскому коду.

ОБЪЕКТ И МЕТОДЫ ИССЛЕДОВАНИЯ

Объектом исследования являются проекты и системы управления проектами. В работе применяются следующие методы исследования:

1. Анализ – проводится анализ требований к приложению, по результатам которого составляются требования к системе.
2. Обобщение – рассмотрение аналогичных систем с целью нахождения общих закономерностей.
3. Сравнение – поиск отличий между разрабатываемой системой и аналогами с целью определения направления дальнейшей разработки.
4. Эксперимент – реализация новых функций системы на стадии разработки с целью определения того, нужны ли они в конечном продукте.
5. Моделирование – создание экспериментальных условий, в которых проверяется работа системы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Большинство представленных на рынке систем управления проектами предоставляют ограниченный функционал при использовании бесплатной версии, например, может присутствовать ограничение на количество участников, которые могут присоединиться к проекту.

Разработанная в ходе работы система может быть установлена на сервер конечного пользователя (hosted-on-premises), что обеспечивает доступность при отключении доступа к сети Интернет, а также безопасность конфиденциальных данных пользователя.

Основной целью работы является разработка системы управления проектами с использованием современных технологий разработки web-приложений, имеющей потенциал для того, чтобы конкурировать с уже существующими системами.

Для определения того, какими функциями и характеристиками должна обладать система управления проектами, проводится анализ предметной области и изучение существующих аналогов. На основании проведенного анализа формируются требования к разрабатываемой системе.

Практическая значимость работы заключается в том, что в настоящее время системы управления проектами крайне востребованы различными организациями. Качество и надежность системы управления проектами, используемой организацией напрямую влияет на успешность выполнения проектов.

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

Практическая новизна разрабатываемого продукта заключается в использовании современных веб-технологий для обеспечения быстрой и удобной работы пользователя в системе.

Разработанный продукт может применяться в небольших компаниях для организации делопроизводства или использоваться в личных целях для организации работы над личными проектами.

В работе была спроектирована и реализована серверная часть системы.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Классификация СУП по назначению

1.1.1 Персональные

Персональная система управления проектами используется в домашних условиях, обычно для управления хозяйством или организации работы над личными проектами пользователя. Персональные системы управления проектами схожи с однопользовательскими системами, однако обычно имеют более простой интерфейс [9].

1.1.2 Однопользовательские

Однопользовательские 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 [10].

1.1.3 Многопользовательские

Многопользовательские системы разрабатываются таким образом, чтобы поддерживать одновременное редактирование различных участков плана несколькими пользователями. Например, пользователи могут иметь возможность редактировать участки проекта, за выполнение которых они ответственны, а все изменения в совокупности позволяют оценивать состояние выполнения проекта в целом. Веб-ориентированные приложения зачастую относятся к этой категории [10].

1.2 Классификация СУП по платформе

Системы управления проектами по типу платформы делятся на два основных типа: прикладные и web-ориентированные [11].

Прикладные приложения реализованы в виде приложения под конкретные операционные системы и требуют установки на ПК пользователя. Плюсом этого подхода является быстродействие таких приложений, в то время как недостатком является необходимость использовать дополнительные средства для переноса на другие ОС.

Web-ориентированные приложения реализованы с помощью HTML, CSS и JavaScript, и предназначены для использования через Web-браузеры. Это значительно упрощает распространение приложения на разные ОС и разные устройства, такие как смартфоны и планшеты, т.к. для их работы нужен только современный браузер, который присутствует в этих системах по умолчанию.

1.3 Функциональные возможности аналогичных систем

1. Обеспечение совместной работы – возможность создания, поддержки и обмена различными процессами, задачами и файлами среди нескольких пользователей и/или систем. ПО с данной особенностью позволяют двум или более удаленным пользователям совместно работать над одними и теми же задачами или проектами.

2. Система отслеживания ошибок (issue tracking system, bug tracking system) – набор ПО, позволяющий вести и отслеживать список ошибок и проблем в той или иной задаче организации. Данный вид ПО часто применяется разработчиками программного обеспечения для учета и контроля ошибок, найденных в программах, а также пожеланий пользователей, для слежения за процессом устранения найденных ошибок и выполнением пожеланий пользователей [12].

3. Система управления документами (document management system) – ПО, используемое для учета, хранения и редактирования электронных документов. В контексте управления проектами данные системы позволяют хранить документооборот, относящийся к проектам и его задачам [12].

4. Система управления рабочими процессами – система, обеспечивающая инфраструктуру для создания, выполнения и отслеживания

определенной последовательности задач, представленных в виде диаграммы рабочих процессов [12].

5. Планирование – процесс организации, контроля и оптимизации работ и рабочей нагрузки в производственных процессах. Средства планирования в системах управления проектами используются для упорядочивания задач и других работ в проекте и установки сроков и требуемых ресурсов для их выполнения.

1.4 Открытость исходного кода

Всего около 80% систем управления проектами – проприетарные, остальные распространяются по одной из лицензий для свободного программного обеспечения (GNU General Public License, Apache License так далее).

1.5 Используемые языки программирования

В системах, для которых известен язык разработки, используются следующие языки:

Таблица 1.1 – Используемые языки программирования

Язык программирования	Количество систем
ASP.NET	6
C++	3
Java	17
PHP	17
Python	5
Ruby	9

1.6 Примеры некоторых популярных систем

1.6.1 Microsoft Project

Microsoft Project – система управления проектами, разработанная Microsoft. Входит в семейство программ Microsoft Office. Проприетарное ПО для ПК, доступное по месячной подписке [13].

Основные возможности

Microsoft Project позволяет манипулировать различными ресурсами проекта (такими, как люди, оборудование или материалы), обладающими стоимостью. Ресурсы распределяются по отдельным заданиям, после чего вычисляется стоимость их использования для выполнения заданий, групп заданий и всего проекта. Ресурсы могут быть общими для нескольких проектов.

Каждому ресурсу соответствует расписание, в котором содержится информация о том, в какое время ресурс доступен. Планирование осуществляется исходя из того, в какие дни доступны требуемые для выполнения заданий ресурсы.

Имеется возможность нахождения критического пути для проекта и визуализация работ в виде диаграммы Гантта.

Существует несколько классов пользователей с различными уровнями доступа к данным, связанным с проектами. Также Microsoft Office Project Server позволяет нескольким пользователям работать удаленно через сеть Интернет: данные о проектах хранятся на сервере в базе данных.

Возможно построение различных графиков, отображающих состояние выполнения проекта, например, график соотношения количества выполненных и ожидающих выполнения заданий.

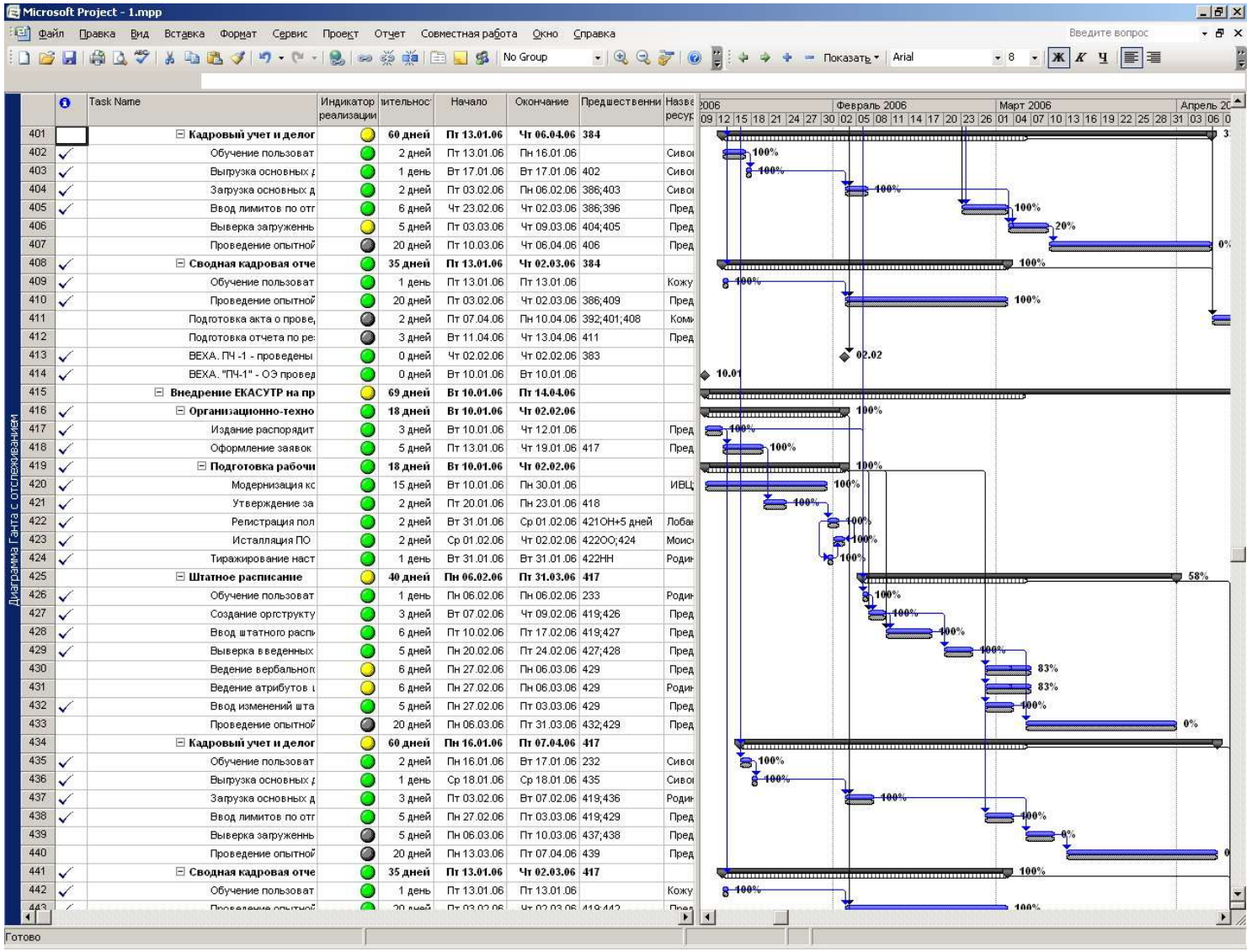

Рисунок 1.1 – Основное окно программы Microsoft Project

1.6.2 Wrike

Wrike – онлайн система управления проектами, разработанная одноименной компанией. Для пользования доступна бесплатная версия с ограниченным количеством участников проекта (до пяти пользователей) и платная по годовой подписке [14].

Далее описываются основные возможности системы.

В приложении присутствует лента новостей, которая агрегирует последние события в проекте и позволяет быстро получать информацию обо всех изменениях.

Существует мобильное приложения для работы с системой при отсутствии доступа к ПК.

Имеется система уведомлений о предстоящих событиях (email, оповещения в браузере).

Также присутствуют различные уже описанные выше возможности, такие как построение диаграммы Гантта, создание визуальных отчетов о состоянии проекта и так далее.

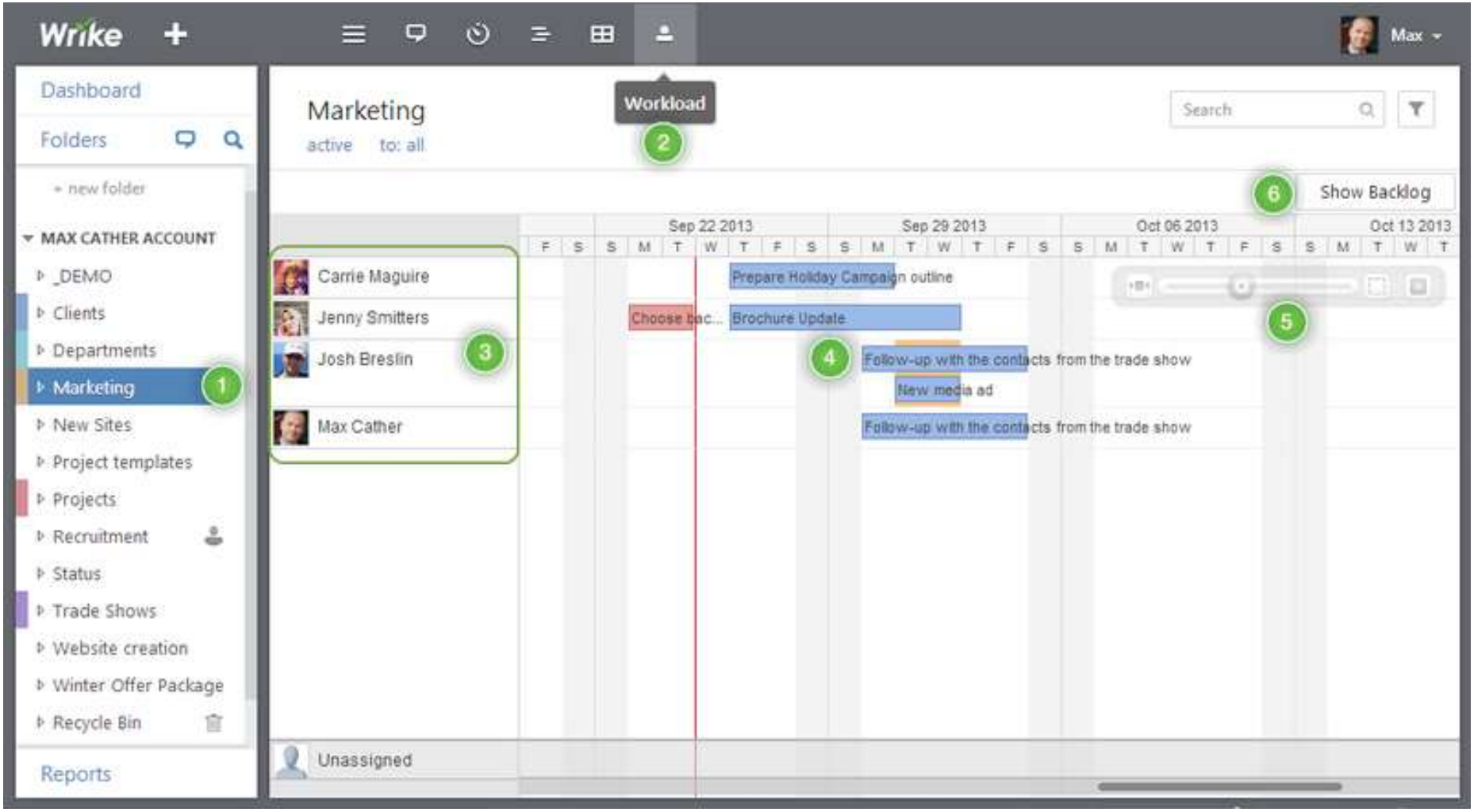

Рисунок 1.2 – Окно программы Wrike

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 Архитектура системы

Разрабатываемая система является web-ориентированной, а по назначению – многопользовательской. Реализация системы в виде веб-сервиса имеет ряд преимуществ:

- мобильность – система доступна в любое время в любом месте при наличии мобильного устройства с доступом в интернет;
- кроссплатформенность – доступ с различных устройств и операционных систем;
- отсутствие необходимости в установке – система доступна прямо из браузера;
- доступ с различных устройств – после того как пользователь завел аккаунт, он может получить доступ к системе с любого устройства.

Система имеет в своем составе клиентскую и серверную части.

Клиентская часть отвечает за предоставление конечному пользователю запрашиваемого контента, в то время как серверная часть отвечает за хранение и обработку данных (в нашем случае – различной информации о проектах), а также предоставление контента конечному пользователю.

Серверная часть приложения реализована по архитектуре MVC. Далее представлена диаграмма, наглядно демонстрирующая архитектуру системы:

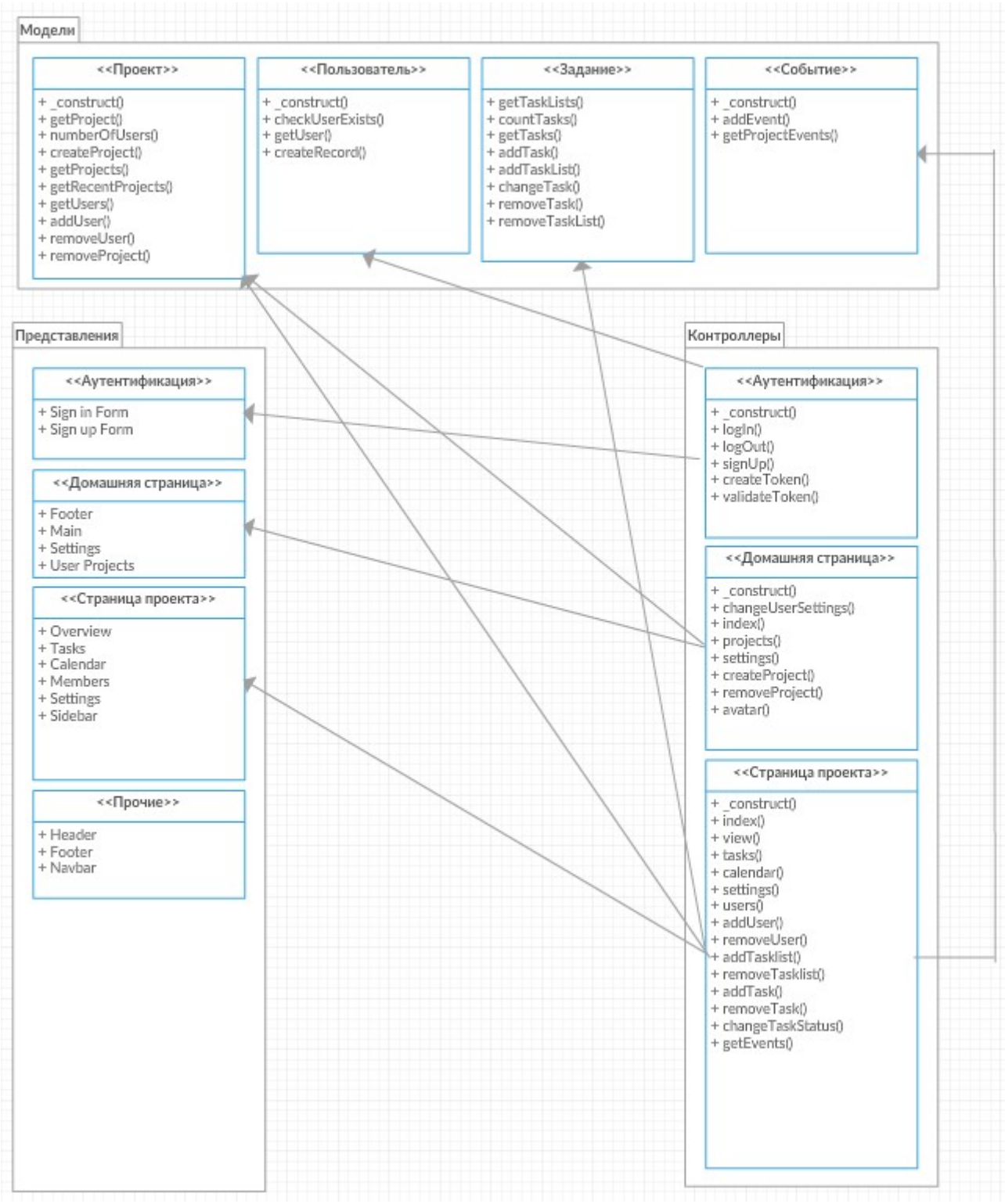

Рисунок 2.1 – Архитектура системы

Архитектура MVC была выбрана, так как она получила широкое распространение в веб-разработке и позволяет упростить написание приложения за счет того, что отдельные компоненты системы независимы друг от друга и легко поддаются модификации.

Подробное описание моделей и контроллеров приведено в разделе «Реализация приложения».

2.2 База данных

В ходе работы над приложением была разработана представленная далее концептуальная модель базы данных [15]:

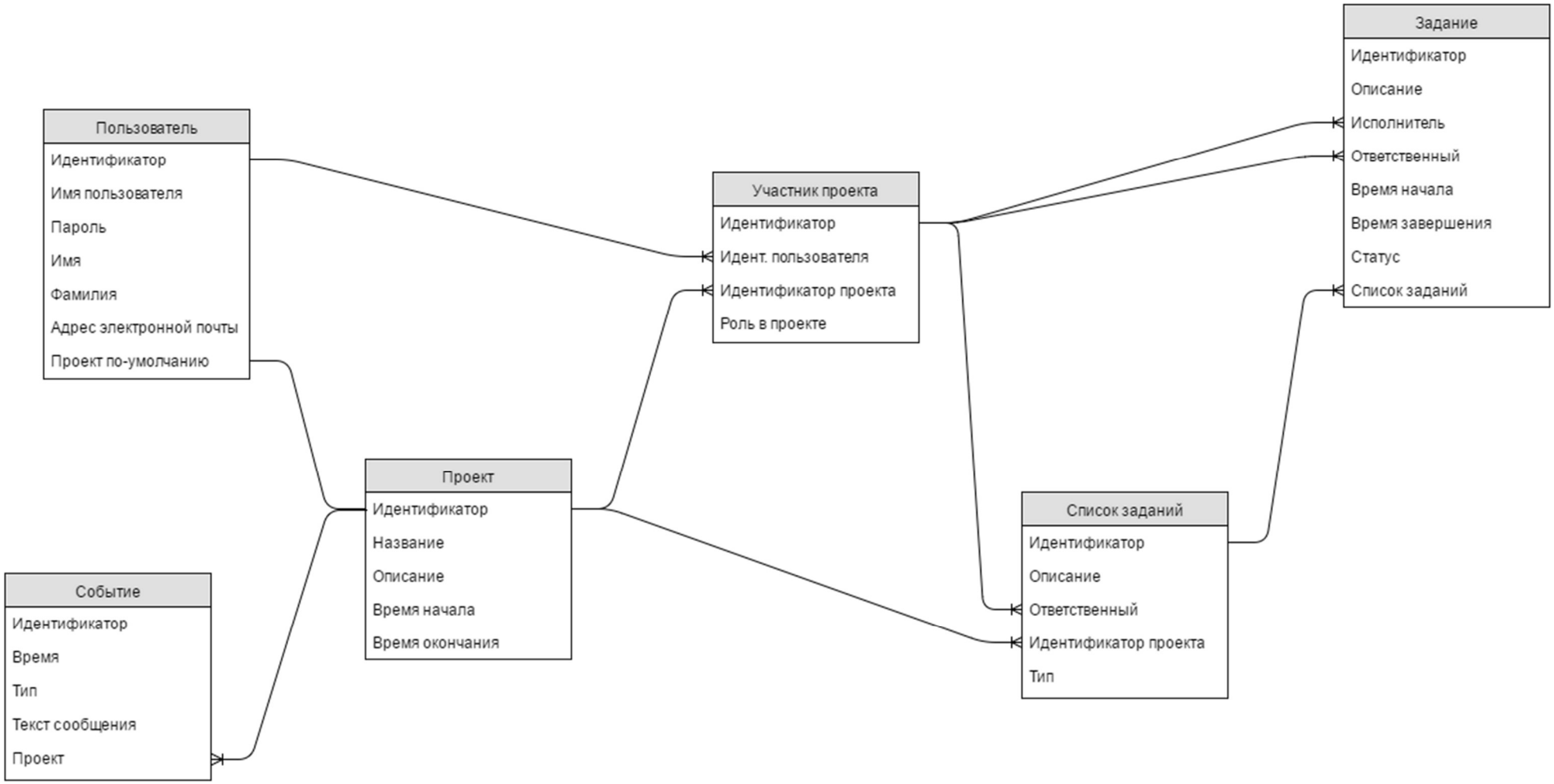

Рисунок 2.2 – Концептуальная модель базы данных

Далее описана ее структура.

Таблицы:

- пользователь;
- проект;
- задание;
- список заданий;
- участник проекта;
- событие.

Таблица «пользователь»:

- идентификатор;
- имя пользователя;
- пароль;
- имя;
- фамилия;
- адрес электронной почты;
- проект по умолчанию – проект, автоматически открывающийся при успешной авторизации.

Таблица «задание»:

- идентификатор;
- описание;
- исполнитель – пользователь, выполняющий задание – связано с идентификатором пользователя.
- проверяющий выполнение задания (ответственный) – связано с идентификатором пользователя;
- время начала выполнения;
- время окончания выполнения;
- состояние выполнения – например, «в работе», «выполнено», «отменено» и так далее;

- список заданий – указывает принадлежность задания к какому-либо списку – связано с идентификатором списка заданий;

Таблица «список заданий»:

- идентификатор;
- описание;
- создатель – связано с идентификатором пользователя;
- проект, в котором находится список заданий – связано с идентификатором проекта;

Таблица «участник проекта»:

- идентификатор проекта;
- идентификатор пользователя;
- роль;

Таблица «проект»:

- идентификатор;
- название;
- описание.
- время начала выполнения;
- время окончания выполнения;

Таблица «событие»:

- идентификатор;
- дата и время;
- тип сообщения – наряду с полем текст определяет содержимое сообщения;
- текст – различная информация, зависящая от типа сообщения, например, название задания, имя пользователя и так далее;
- дата и время.

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1 Используемые при реализации инструменты и технологии

3.1.1 Архитектура REST

REST – архитектура проектирования интернет-приложений. Представляет собой набор условий проектирования, соблюдение которых потенциально приводит к увеличению производительности и сопровождаемости системы. Системы, построенные по архитектуре REST, иногда называют RESTful [1].

Архитектурные особенности, которыми должна обладать система для того, чтобы удовлетворять требованиям REST:

- клиент-серверная архитектура;
- отсутствие состояний: сервер не хранит контекст клиента между запросами; информацию о текущей сессии должен хранить клиент и формировать запросы на сервер таким образом, чтобы в нем содержались все необходимые данные для его обработки;
- кэшируемость – ответы сервера на запросы клиента могут определяться как кэшируемые или некэшируемые. Кэширование некоторых ответов позволяет уменьшить необходимое взаимодействие между клиентом и сервером;
- многоуровневость – в системе возможно использование промежуточных серверов для уменьшения нагрузки. При этом клиент не узнает, что подключен к промежуточному серверу;
- использование единообразного интерфейса – позволяет упростить и разделить архитектуру для того, чтобы отдельные ее части могли разрабатываться независимо;
- идентификация ресурсов – ресурсы определяются в запросах. На основании ресурсов формируются представления, которые отправляются пользователю;

- совершение действий над ресурсами с помощью представлений – если клиенту необходимо модифицировать ресурс, обладая представлением он имеет всю необходимую информацию для изменения ресурса;
- подробные сообщения – сообщения содержат информацию о том, как их нужно обрабатывать;
- текущее состояние передается непосредственно в самом запросе.

3.1.2 Серверная часть

Серверная логика приложения описана на языке PHP. За основу был взят фреймворк CodeIgniter 3.

CodeIgniter – общеизвестный MVC-фреймворк с открытым исходным кодом, написанный на языке программирования PHP, для разработки полноценных веб-систем и приложений [16].

Рисунок 3.1 – Логотип фреймворка CodeIgniter

Отличительными сторонами фреймворка являются [17]:

- качественная и полная документация и поддержка интернет сообщества;
- быстрота и малые требования к ресурсам сервера;
- минимальная настройка;
- простая и удобная реализация MVC модели, благодаря чему достигается четкое разделение кода логики приложения, представлений и взаимодействия с базами данных;
- множество классов-помощников для работы с различными аспектами веб-сервиса.

В качестве системы управления базой данных была использована MySQL. MySQL являет свободной реляционной системой управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. MySQL является решением для малых и средних приложений. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц [18].

Рисунок 3.2 – Логотип MySQL

Достоинства системы:

- простота в использовании и установке;
- система обладает несколькими слоями безопасности для защиты конфиденциальных данных от злоумышленников. Внутри самой системы есть возможность настройки привилегий для пользователей СУДБ;
- данная СУБД является ПО с открытым кодом и имеет возможность как свободного распространения, так и распространения по коммерческой лицензии;
- высокая скорость работы;
- масштабируемость: MySQL может обрабатывать вплоть до 50 миллионов строк в таблице, при этом по умолчанию размер таблицы установлен в 4 Гб, а максимальный размер ограничивается максимальным размеров файла используемой файловой системы;
- MySQL может работать на многих операционных системах, таких как Windows, Linux, различных вариантах UNIX-подобных систем и других;

- существуют API для большого числа языков программирования, таких как Delphi, C, C++, Java, LISP, Perl, PHP, Python, Ruby, Smalltalk, библиотеки для языков платформы .NET и другие.

3.1.3 Клиентская часть

Так как клиентом разрабатываемого приложения является браузер, клиентская часть приложения написана с использованием HTML5, CSS3, JavaScript.

В качестве основного фреймворка был использован Bootstrap 3 – свободный набор инструментов для создания сайтов и веб-приложений. Bootstrap включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов.

Также использована JavaScript-библиотека React для реализации пользовательского интерфейса.

3.1.4 XAMPP

Для тестирования приложения был использован XAMPP – кроссплатформенная сборка веб-сервера с открытым кодом, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер. Изначально XAMPP создавался как инструмент для разработчиков, позволяя веб-дизайнерам и программистам тестировать свою работу без помощи внешних серверов. Для упрощения работы некоторые возможности и настройки безопасности отключены по умолчанию, и в целом XAMPP рекомендуется к использованию только в очень дружественном окружении.

3.2 Аутентификация

В разработанной системе для аутентификации пользователей используется механизм под названием JWT (JSON Web Token, предлагаемое произношение – «jot»).

JWT обладает рядом преимуществ:

- стандартизированность – обмен информацией осуществляется при помощи широко распространенного формата JSON (JavaScript Object Notation);
- самодостаточность – вся информация, необходимая для обработки JWT содержится в нем самом. JWT включает в себя информацию для обработки (заголовок), пользовательскую информацию, а также цифровую подпись;
- реализации существуют для большого количества различных языков программирования, например, PHP, Python, .NET, Ruby, Javascript, Node.js и так далее;
- простота передачи – поскольку при передаче JWT используется всего лишь одна структура, записанная в формате JSON, его можно вставлять в заголовок HTTP при аутентификации или передавать при помощи URL.

JWT имеет несколько применений:

- аутентификация (используется в разработанной системе и описывается далее);
- безопасный обмен информацией – JWT позволяют безопасно обмениваться информацией, так как при его создании используется ассиметричное шифрование, что позволяет удостовериться в личности отправителя. Также в JWT присутствует цифровая подпись, вычисляемая на основании заголовка и пользовательских данных, что гарантирующая целостность передаваемой информации.

JWT состоит из трех частей, разделенных точками:

- заголовок;
- полезная нагрузка (payload);
- подпись.

Таким образом, в общем случае JWT выглядит так: «заголовок.полезная_нагрузка.подпись»

Заголовок состоит из двух частей:

- название токена (JWT);
- используемый алгоритм хэширования.

Пример заголовка:

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

Записанный таким образом заголовок кодируется с помощью Base64 и становится первой частью JWT.

Во второй части JWT содержатся так называемые «claims» (заявки). Заявки представляют собой информацию, которую мы хотим передать. Существуют различные виды заявок:

- зарезервированные: iss – эмитент токена, aud – получатели токена и так далее;
- публичные (public) – такая информация, как имя пользователя или информация о нем. Чтобы избежать конфликтов, имена заявок должны быть определены;
- частные (private) – стороны договариваются об именах заявок, при этом возможны конфликты.

Третья часть JWT – подпись. Подпись создается на основе трех компонентов:

- заголовок;
- полезная нагрузка;
- кодовое слово.

Кодовое слово хранится на сервере и используется для создания новых токенов и проверки уже существующих.

JWT работает следующим образом: при первом обращении пользователя к системе происходит проверка введенных регистрационных данных. При

успешном входе в систему пользователю выдается JWT, который используется при всех последующих действиях, требующих аутентификации.

Далее представлен пример формирования JWT на стороне сервера (в случае успешной аутентификации):

```
if ($credentialsValid) {            //при успешной аутентификации
    $tokenID = base64_encode(mcrypt_create_iv(32)); //уникальный id
    $issued  = time();               //получение текущего времени
    $nbf     = $issued + 3;          //токен действителен с
    $expire  = $nb + 3600;           //время истечения действия токена
$issuer  = 'projectr';

$data    = [

    'iat'  => $issued,           //время создания токена

    'jti'  => $tokenID,          //уникальный идентификатор

    'iss'  => $issuer,           //создатель токена

            'nbf'  => $nbf,              //действителен с
            'exp'  => $expire,           //время истечения действия токена
            'data' => [
                'userID'   => $id,       //id пользователя
            ]
        ];
}
```

В примере создается токен со следующими параметрами:

- tokenID – уникальный идентификатор токена;
- issued – время создания токена;
- nbf – время начала действия токена;
- expire – время истечения действия токена;
- issuer – имя сервера, создавшего токен;
- userID – идентификатор пользователя.

Далее полученный массив $data шифруется с помощью секретного ключа сервера:

```
$jwt = JWT::encode(
 $data,      //шифруемые данные
        $secretKey, //ключ, которым производится шифрование
        'HS512'     //алгоритм шифрования
```

```
);
$array = ['jwt' => $jwt];
echo json_encode($array); //отправка jwt пользователю
```

Функция JWT::encode создает заголовок для JWT и шифрует данные $data с помощью секретного ключа $secretKey.

Далее представлена реализация сохранения, полученного JWT на стороне клиента:

```
$(function(){
    var store = {};
    store.setJWT = function(data){
        this.JWT = data;
    }
    //отправка формы аутентификации
    $("#LoginForm").submit(function(e){ //при отправке формы
 e.preventDefault();
 //совершить POST запрос, содержащий данные формы
 $.post('auth/token', $("#LoginForm").serialize(),
 function(data){
        store.setJWT(data.JWT); //при успехе – сохранить JWT
 }).fail(function(){
 ...
 });
    });
});
```

На стороне клиента происходит отправка формы аутентификации. В POST запросе отправляются данные для входа в систему. При успешной аутентификации происходит сохранение данных в переменную store. Иначе (при получении HTTP ответа с кодом состояния 4xx) вызывается функция fail и присвоения не происходит.

Для передачи токена на сервер клиент присваивает полю Authorization в HTTP заголовке значение «Bearer[JWT]».

Далее представлен пример обработки запросов, содержащих JWT для аутентификации:

```
$authHeader = $this->input->get_request_header('Authentication', TRUE);

if ($authHeader) {
 list($jwt) = sscanf( $authHeader, 'Authorization: Bearer %s');
        if ($jwt) {
```

```
                try {
                        $token = JWT::decode($jwt, $secretKey, array('HS512'));
                        header('Content-type: application/json');
                        echo json_encode(
                                        ... //отправка данных пользователю
                                );
                    } catch (Exception $e) {//не удалось расшифровать токен
                                header('HTTP/1.0 401 Unauthorized');
                    }
                } else {
                    header('HTTP/1.0 400 Bad Request'); //не удалось извлечь токен
из //запроса
                }
            } else {//не найдено поле Authentication
                header('HTTP/1.0 400 Bad Request');
                echo 'Token not found in request';
            }
        }
```

3.3 Модели

Согласно концепции MVC, используемой в разработанной системе на стороне сервера, было реализовано некоторое множество моделей. Модель получает необходимую информацию из базы данных. На основе этой информации генерируются представления, которые видит пользователь. С помощью контроллеров происходит управление моделями (добавление, удаление, модификация данных).

В системе используются следующие модели:

- пользователь (User model);
- проект (Project model);
- задание (Task model);
- событие (Event model).

3.3.1 Модель «пользователь»

Модель «пользователь» позволяет манипулировать информацией о пользователях, зарегистрированных в системе. С помощью этой модели можно

добавлять новых пользователей (при регистрации нового аккаунта), проверять, существует ли пользователь с указанным именем (необходимо, например, при регистрации нового пользователя или при добавлении уже существующего пользователя в проект), а также получать информацию о пользователе.

Далее приведено содержание модели «пользователь»:

```
function __construct()
{
     $this->load->database();
}
```

При создании модели происходит подключение к базе данных. Информация для подключения (такая, как логин, пароль, адрес и порт подключения и так далее) содержится в конфигурационном файле фреймворка CodeIgniter.

```
function checkUserExists($username)
{
      $this->db->select('*');
      $this->db->from('user');
      $this->db->where('username', $username);

       $query = $this->db->get();
       if ($query->num_rows()) {
            return $query->result_array();
       }
       return false;
}
```

Функция checkUserExists проверяет, зарегистрирован ли пользователь в системе. Если пользователь с заданным именем существует, возвращаются его данные, иначе возвращается значение false.

```
function getUser($id)
{
     $this->db->select('*')->from('user')->where('id', $id);
     return $this->db->get()->result_array()[0];
}
```

Функция getUser возвращает информацию о пользователе с указанным идентификатором.

```
function createRecord($userData)
{
```

```
        if ($this->db->insert('user', $userData)) {
              return $this->db->insert_id();
        }
        return false;
    }
```

Функция createRecord позволяет нового пользователя с указанными регистрационными данными userData. При успешном выполнении операции возвращается идентификатор пользователя, иначе – значение false.

3.3.2 Модель «проект»

Модель «пользователь» позволяет манипулировать информацией о проектах, содержащихся в системе.

```
function getProject($id)
{
    $this->db->select('*')->from('project')->where('id', $id);
    $result = $this->db->get();
    return $result->result_array()[0];
}
```

Функция getProject позволяет получить информацию о проекте с указанным идентификатором.

```
function numberOfUsers($id)
{
    $this->db->select('*')->from('project_members')->     where('project_id',
$id);
    $result = $this->db->get();
    return $result->num_rows();
}
```

Функция numberOfUsers позволяет получить информацию о количестве пользователей, являющихся участниками проекта с указанным идентификатором.

```
function createProject($info)
{
   ...
}
```

Функция createProject позволяет создать новый проект с указанными параметрами. В список параметров входят:

- название проекта;
- создатель проекта;
- краткое описание проекта;
- время начала выполнения;
- время окончания выполнения проекта.

```
function getProjects($user_id, $recent = false, $limit = 3)
{
      ...
}
```

Функция getProjects позволяет получить список проектов, в которых состоит указанный пользователь. При этом есть возможность получить список последних проектов пользователя. Для этого нужно вызвать функцию со значением аргумента $recent равным true. Аргумент $limit позволяет устанавливать количество возвращаемых функцией проектов (по умолчанию возвращается 3 проекта).

```
function getRecentProjects($user_id)
{
       return $this->getProjects($user_id, true);
}
```

Функция getRecentProjects позволяет получить последние проекты пользователя с указанным идентификатором. Это осуществляется при помощи вызова функции getProjects с аргументом $recent, равным true.

```
function getUsers($id)
{
      ...
}
```

Функция getUsers позволяет получить информацию о пользователях, принимающих участие в проекте с указанным идентификатором. Данная информация используется, например, при выводе сводной информации о проекте на соответствующей странице.

```
function addUser($project_id, $username)
{
       ...
}
```

Функция addUser позволяет добавить нового пользователя с указанным именем в проект. При этом сначала осуществляется поиск пользователя с заданным именем в системе; если такой пользователь существует – происходит проверка того, не состоит ли он уже в указанном проекте. После этого пользователь добавляется в проект.

```
function removeUser($project_id, $username)
{
	...
}
```

Функция removeUser позволяет удалить пользователя с указанным идентификатором из проекта.

```
function removeProject($id)
{
	...
}
```

Функция removeProject позволяет удалить проект с указанным идентификатором. При этом также происходит удаление всех заданий, списков заданий и информации о принадлежности пользователей проекту.

Функция changeUserStatus позволяет изменять роль пользователя.

3.3.3 Модель «задание»

Модель «задание» позволяет манипулировать информацией о заданиях, из которых состоят проекты. С помощью этой модели можно создавать и удалять задания и списки заданий в проектах, а также получать различную информацию о них.

```
function getTaskLists($id)
{
	...
}
```

Функция getTaskLists позволяет получить информацию обо всех списках заданий, из которых состоит проект.

```
function countTasks($id)
{
	...
}
```

Функция countTasks позволяет получить информацию о количестве заданий в определенном проекте. Эта информация используется при выводе обзорной информации о проекте.

```
function getTasks($tasklist, $filter, $user = 0)
{
        ...
}
```

Функция getTasks возвращает структурированную информацию о заданиях в указанном проекте. Возвращаемые данные представляют собой список всех списков заданий со вложенными в них заданиями. Таким образом можно получить полную информацию обо всех заданиях в проекте. С помощью указания аргументов $filter и $user можно отфильтровать получаемые задания по исполнителю.

```
function _getSystemTasklist ($id)
{
        ...
}
```

С помощью этой функции можно получить список заданий в проекте, которые не включены ни в один из списков (неотсортированные задания).

```
function addTask($info)
{
        ...
}
```

С помощью функции addTask можно добавить в проект задание с указанными параметрами. В список указываемых параметров входят:

1. исполнитель (идентификатор);
2. проверяющий выполнение;
3. проект, в который входит задание;
4. список заданий, в который входит задание (выбирается из списка уже существующих);
5. время начала выполнения;
6. время окончания выполнения.

```
function addTasklist($data)
{
```

```
        ...
}
```

Функция addTasklist позволяет добавить новый список заданий в проект. При добавлении указывается название (краткое описание) и ответственный за выполнение.

```
function changeTask($id, $newval, $field)
{
            ...
}
```

Функция changeTask позволяет присвоить указанному параметру ($field) задания значение $newal.

Функции removeTask и removeTasklist позволяют удалять из проекта определенное задание или список заданий соответственно.

При совершении определенных действий (таких как добавление пользователя или задания в проект) создается событие, которое добавляется в базу данных. События отображаются на главной странице пользователей, принимающих участие в проекте.

3.4 Контроллеры

Контроллеры в MVC обеспечивают обработку запросов пользователя, выполнение требуемых действий над моделью и подготовку представлений для выдачи пользователю.

В системе используются следующие контроллеры:

1. контроллер аутентификации;
2. контроллер домашней страницы;
3. контроллер проекта.

Доступ к функциям контроллера в CodeIgniter с помощью URL осуществляется следующим образом:

```
example.com/ctrl/fn/param1/param2/... ,
```

где ctrl – название класса контроллера, func – название функции в классе, param(1, 2, …) – аргументы функции. Таким образом при переходе по представленному выше адресу произойдет вызов функции

```
function func($param1, $param2) { ... }
```
в контроллере ctrl.

3.4.1 Контроллер аутентификации

В контроллере аутентификации реализуются такие функции, как регистрация в системе (sign up), вход и выход из системы (log in и log out). Для аутентификации используются механизм JWT, описанный в разделе «Аутентификация».

3.4.2 Контроллер домашней страницы

Данный контроллер обрабатывает действия пользователя, совершаемые на домашней странице.

При обращении к функции index, которая соответствует домашней странице пользователя, происходит получение данных о пользователе и его последних проектах из соответствующих моделей, а также установка заголовка страницы:

```
$data['recent_projects'] = $this->_recentProjects($id);
$data['title'] = 'Home';
$data['user'] = $id;
```

На основе этих данных создается представление, отправляемое пользователю:

```
$this->load->view('header', $data);
$this->load->view('navbar', $data);
$this->load->view('home/main', $data);
$this->load->view('home/footer', $data);
$this->load->view('footer');
```

header, navbar, footer и прочее – различные элементы страницы. Передача $data позволяет модифицировать передаваемую пользователю страницу на основе информации, полученной из модели.

projects() соответствует странице со списком проектов пользователя. Аналогично происходит запрос для получения информации о проектах пользователя из модели, а затем генерация представления на основе полученных данных.

Также присутствует функция, соответствующие страницам настроек (settings).

3.4.3 Контроллер проекта

Контроллер домашней страницы обеспечивает взаимодействие системы с пользователем традиционным способом (в ответ на запрос пользователя ему присылается готовая страница с разметкой). Контроллер проекта, напротив, работает по REST-технологии, то есть выдает пользователю не готовую разметку, а присылает только нужные данные в формате JSON. Пример:

```
header('Content-Type: application/json');
echo json_encode(array('title' =>$data['title'], 'tasks' => $data['tasks']));
```

Пользователю в формате JSON отправляется информация о заголовке страницы и о заданиях в текущем проекте.

Контроллер проекта обеспечивает взаимодействие со следующими разделами:

- function view($id) – раздел с обзорной информацией о проекте;
- function tasks($id) – раздел с заданиями проекта;
- function calendar($id) – раздел с календарем проекта;
- function users($id) – раздел участников проекта;
- function settings($id) – раздел настроек проекта.

Также в контроллере присутствуют различные функции для совершения операций над моделями, такими как создание и удаление заданий, добавления пользователей и так далее. Пример добавления пользователя:

```
$data['description'] = $this->input->post('task_desc');
$data['supervisor'] = $this->input->post('supervisor');
$data['timestart'] = $this->input->post('timestart');
$data['timedue'] = $this->input->post('timedue');
$data['status_id'] = $this->input->post('status_id');
$data['tasklist'] = $this->input->post('tasklist');
$data['implementer'] = $this->input->post('implementer');
$this->task_model->addTask($data);
```

В этом примере сначала извлекаются данные из POST запроса пользователя и формируется массив данных, необходимых для добавления нового задания, затем происходит вызов соответствующей функции модели, совершающей действия над базой данных, в результате чего происходит добавление нового задания.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Группа	ФИО
8В2А	Шахов Владимир Сергеевич

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

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
– *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих* – *Нормы и нормативы расходования ресурсов* – *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:	
• *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения* • *Формирование плана и графика разработки и внедрения инженерного решения* • *Формирование организационной структуры управления инженерным проектом* • *Планирование потребности в человеческих ресурсах*	*Оценка ресурсной, социальной эффективности НИ и потенциальных рисков.*
Перечень графического материала *(с точным указанием обязательных чертежей)*:	
• *Оценка плана проведения НИ* • *QuaD-технология* • *Матрица SWOT* • *Таблица результатов проведения НИ*	

Дата выдачи задания для раздела по линейному графику	14.05.2016

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Николаенко Валентин Сергеевич			14.05.2016

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В2А	Шахов Владимир Сергеевич		14.05.2016

4 ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

4.1 Введение

Настоящая разработка представляет собой веб-ориентированную систему управления проектами. Основная функция, выполняемая системой, – предоставление множеству пользователей, принимающих участие в проекте, инструментов для отслеживания текущего состояния проектов, внесения изменений и разделения проектов на более мелкие составные части (задания и списки заданий).

Система представляет собой веб-приложение, что позволяет получить к ней доступ с большинства устройств, имеющих браузер и доступ в интернет. Приложение разработано по одностраничной технологии, что значительно ускоряет работу с ним.

Подобной системой могут заинтересоваться несколько различных целевых аудиторий. Во-первых, это небольшие компании, не готовые тратить крупные суммы на приобретение дорогостоящих профессиональных систем управления проектами, но которым, тем не менее, необходимо средство для организации протекающих в ней процессов.

Во-вторых, это индивидуальные предприниматели и частные лица, которым может понадобиться приложение для систематизации их дел. Например, пользователь может использовать ее для организации своих личных проектов.

В настоящем разделе проводится анализ ресурсоэффективности и конкурентоспособности разработки, рассмотрение возможности проведения дальнейших исследований, определение эффективности разработанной системы.

4.2 Цели и задачи

Основная цель настоящего раздела ВКР – оценка конкурентоспособности разработки, ее эффективности в областях ресурсосбережения и ресурсоэффективности.

Для оценки разработки применяется SWOT-анализ (Strength, Weaknesses, Opportunities, Threats) и технология оценки QuaD (Quality Advisor).

SWOT-анализ позволяет выявить сильные и слабые стороны разработки, благоприятные и неблагоприятные варианты развития событий. Результаты SWOT-анализа помогают принимать решения в ходе разработки и продвижения проекта [19].

В ходе выполнения SWOT анализа необходимо решить следующие задачи:

- описать сильные и слабые стороны проекта;
- описать позитивные и негативные риски;
- найти корреляцию сильных и слабых сторон проекта и рисков.

Технология QuaD позволяет оценить разработку используя взвешенные значения разнообразных показателей, объединенных в две группы:

- Показатели оценки коммерческого потенциала разработки;
- показатели оценки качества разработки.

Сами показатели, которые составляют эти группы, выбираются в зависимости от специфики проекта, его разработки, реализации и продвижения.

В заключение проводится определение возможных альтернатив проведения дальнейших исследований.

4.3 SWOT-анализ

Таблица 4.1 – Результаты SWOT-анализа

	Сильные стороны 1. Бесплатное распространение 2. Высокая скорость работы 3. Качественный пользовательский интерфейс 4. Использование современных технологий 5. Кроссплатформенность системы 6. Востребованность в системах управления проектами	Слабые стороны 1. Отсутствие поддержки некоторых устаревших браузеров 2. Наличие достаточного количества конкурентов 3. Незавершенность (некоторые части системы имеют потенциал для улучшения)
Возможности 1. Возможность расширения существующего функционала системы 2. Возможность коммерциализации проекта	В2С6: коммерциализация в сочетании с наличием высокого спроса могут обеспечить прибыль	В1Сл2: расширение функционала системы позволит получить преимущество над системами конкурентов
Угрозы 1. Недостаточность привлеченных ресурсов 2. Ошибки в реализации 3. Недостаточная по сравнению с конкурентами функциональная мощность системы	У2С2С3С5: ошибки в реализации системы могут уменьшить спрос на систему, несмотря на такие преимущества, как высокая скорость работы или качественный интерфейс	У3Сл2: так как на рынке присутствуют конкуренты, недостаточная по сравнению с ними функциональная мощность может привести к непопулярности системы

4.4 QuaD-анализ

Таблица 4.2 – Оценочная карта

Критерии оценки	Вес	Баллы	Макс. балл	Отн. Знач.	Взвеш. знач.
Показатели оценки качества разработки					
1. Функциональная мощность	0.05	70	100	0.7	0.035
2. Простота эксплуатации	0.05	90	100	0.9	0.045
3. Устойчивость к сбоям	0.05	90	100	0.9	0.045
4. Безопасность	0.1	80	100	0.8	0.08
5. Нагрузка на сеть	0.05	90	100	0.9	0.045
6. Нагрузка на сервер	0.05	90	100	0.9	0.045
7. Качество пользовательского интерфейса	0.1	100	100	1	0.1
8. Ремонтопригодность	0.05	70	100	0.7	0.035
9. Универсальность выполняемых задач	0.05	80	100	0.8	0.04
10. Кроссплатформенность	0.05	90	100	0.9	0.045
11. Функциональная полнота	0.1	90	100	0.9	0.09
Показатели оценки коммерческого потенциала разработки					
12.Конкурентоспособность разработки	0.05	70	100	0.7	0.035
13. Уровень проникновения на рынок	0.05	0	100	0	0
14. Перспективность рынка	0.05	90	100	0.9	0.045
15. Цена	0.05	100	100	1	0.05
16. Срок выхода на рынок	0.05	70	100	0.7	0.035
17. Законченность работы	0.05	80	100	0.8	0.04

18. Финансовая эффективность разработки	0.05	70	100	0.7	0.035
Итог:					0.845

Далее описываются выбранные для оценки с помощью технологии QuaD показатели.

Показатели оценки качества разработки:

Функциональная мощность – количество и качество исполнения разнообразных функций, предоставляемых пользователю в рамках системы.

Простота эксплуатации – способность пользователя выполнять наиболее необходимые действия за минимальное количество совершенных операций.

Устойчивость к сбоям – способность системы восстанавливать работоспособность после сбоя за сравнительно небольшой период времени или даже полностью предотвращать возможные сбои без каких-либо последствий.

Безопасность – способность системы противостоять атакам злоумышленников, а также обеспечивать сохранность пользовательских данных при совершении разнообразных операций.

Нагрузка на сеть – минимальное необходимое количество данных, передаваемое между серверной и клиентской частью приложения в единицу времени, которое обеспечивает оптимальную работу пользователя в системе без сбоев и нарушений производительности (например, ситуация, когда пользователь вынужден ожидать выполнение запрашиваемой операции в течении длительного промежутка времени).

Нагрузка на сервер – необходимые для оптимальной работы системы ресурсы на компьютере, поддерживающем работы серверной части приложения.

Качество интерфейса – интуитивная понятность пользователем совершаемых им действий в системе, возможность быстро запомнить расположение основных элементов управления, адекватная обратная связь.

Ремонтопригодность – способность администратора системы в кратчайшие строки выявить неполадку в системе и совершить необходимые действия для ее устранения и ликвидации последствий неполадки.

Универсальность выполняемых задач – разнообразие всевозможных задач, которые можно выполнить с использованием разработанной системы.

Кроссплатформенность – способность системы работать на разнообразных операционных системах и устройствах с одинаковым уровнем производительности и функционалом.

Функциональная полнота – способность системы предоставить пользователю весь необходимый функционал для решения тех задач, на которые ориентирована разработанная система.

Показатели оценки коммерческого потенциала разработки:

Конкурентоспособность разработки – способность разработки соперничать с аналогичными продуктами на рынке и иметь достаточную долю пользователей.

Уровень проникновения на рынок – соотношение текущего количества пользователей разработанного продукта и потенциального количества его пользователей.

Перспективность рынка – определяется прогнозами относительно, например, заинтересованности покупателей в продуктах данного сегмента рынка, динамикой спроса и предложения, цен и так далее. Определяет вероятность успешного продвижения разрабатываемого продукта на рынок.

Цена – предполагаемая или реальная цена продукта.

Срок выхода на рынок – время, в течении которого возможен запуск продукта на рынок.

Законченность работы – характеризует то, на какой стадии разработки или тестирования находится разработка.

Финансовая эффективность разработки – соотношение реальной или предполагаемой прибыли к затратам.

4.5 Определение возможных альтернатив проведения исследований

Сущность морфологического анализа заключается в выделении возможных характеристик объекта исследования и изучения различных комбинаций этих характеристик. Далее представлена таблица, в которой описаны различные морфологические характеристики и их возможные варианты.

Таблица 4.3 – Морфологическая матрица для генетического алгоритма

	1	2	3
А. Архитектура системы	Одностраничная (single page application)	Классическая	Смешанная
Б. Front-end фреймворк	React	AngularJS	Ember
В. Способ организации клиент-серверного взаимодействия	REST	SOAP	graphQL
Г. Язык программирования, используемый для написания серверной части системы	PHP	Python	Node.js
Д. Поддерживаемые платформы	Настольное приложение	Веб-приложение	Программный комплекс (десктоп приложение, мобильное

			приложение, веб приложение)
Е. Используемая база данных	MySQL	MsSQL	PostgreSQL
Ж. Архитектура сервера	MVC	MVVM	
И. Количество пользователей	Однопользовательская система	Многопользовательская система	
К. Модель распространения	Платная	Бесплатная	

4.6 Вывод

В настоящем разделе были рассмотрены и применены на практике различные методы оценки эффективности разработки, проведен SWOT и QuaD анализ разработанной системы управления проектами, проведен морфологический анализ для выявления перспективных направлений проведения исследований.

В результате SWOT-анализа выявлены сильные и слабые стороны разработанной системы, позитивные и негативные риски, связанные с ее жизненным циклом. Выявлены и описаны зависимости между сильными и слабыми сторонами и рисками.

В ходе QuaD-анализа составлена оценочная карта разработанного продукта.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА

«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8В2А	Шахов Владимир 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. Ошибки в логике работы системы	• Вызваны ошибками в структуре базы данных или ошибками в коде серверной части • Некоторые задания в системе могут быть не выполнены • Задания могут быть выполнены несколько раз • Некоторые участники могут не получить доступ к информации • Для предотвращения подобных ошибок проводится тестирование
2. Ошибки в пользовательском интерфейсе	• Отсутствие информационных сообщений затрудняет взаимодействие с системой • Неправильное расположение элементов может привести к ошибкам • Нелогичное поведение системы • Ошибки ухудшают взаимодействие пользователя с системой
3. Утечка персональных данных	• В системе хранятся персональные данные пользователей • Утечка персональных данных может привести к серьезным последствиям, ряд лиц может быть привлечен к ответственности

4. Утечка данных компании	• В системе хранится информация об организации, утечка которой может привести к финансовым убыткам • Информация о внутренней структуре организации • Информация о ходе выполнения проекта (например, сроки) • Информация о составных частях проекта. При получении этой информации конкуренты могут получить преимущество • Для предотвращения используются такие механизмы, как проверка личности пользователя и использование прав доступа
5. Потеря данных	• Потеря данных о целом проекте или его частях приведет к излишним временным затратам для восстановления потерянной информации • Потеря доступа к системе на какое-то время приведет к простою в работе организации • Причиной потери данных может быть ошибка в серверной части кода системы или ошибка пользователя • Причиной потери доступа к системе может быть DoS атака

Дата выдачи задания для раздела по линейному графику	14.05.2016

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Невский Егор Сергеевич			14.05.2016

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В2А	Шахов Владимир Сергеевич		14.05.2016

5 СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

5.1 Введение

В настоящем разделе выпускной квалификационной работы рассматривается влияние разработанного программного продукта на общество, различные способы уменьшения или предотвращения наносимого вреда, а также факторы, оказывающие негативное влияние на общество и его безопасность.

Программные продукты не оказывают непосредственного негативного влияния на окружающую среду, в отличие от реально существующих объектов, таких, как, например, атомная электростанция, однако ошибки в проектировании или реализации программных продуктов способны оказать значительное влияние на безопасность объектов, которые управляются с помощью ПО, а также на безопасность пользователей ПО (частных лиц или организаций).

При проектировании программных продуктов необходимо рассматривать такие уже ставшие классическими вопросы обеспечения безопасности, как:

Конфиденциальность информации – информация, не подлежащая разглашению должна оставаться внутри системы. Злоумышленники или пользователи с недостаточным уровнем прав не должны иметь возможности получить к ней доступ.

Сохранность информации – информация, хранящаяся в системе, не должна исчезать или изменяться ненадлежащим образом при хранении, передаче или совершении каких-либо операций над ними.

Свободный доступ к информации – пользователь, обладающий надлежащими правами, должен иметь беспрепятственный доступ к информации, которую позволяет просматривать или изменять его уровень доступа.

Рассмотрению проблем, которые могут возникнуть при обеспечении конфиденциальности, сохранности и доступа к информации, а также рассмотрению некоторых других проблем посвящен настоящий раздел.

5.2 Опасные факторы

Разработанный программный продукт представляет из себя систему управления проектами, что подразумевает собой оперирование сведениями о частных лицах и организациях, участвующих в выполнении проекта, сведениями о самом проекте, сроках его выполнения и прочей важной информацией.

В связи с этим возникает опасность утечки конфиденциальных данных из системы, что может нанести значительный вред компании, ответственной за выполнение проекта. Таким образом, требуется использование различных механизмов защиты системы от несанкционированного доступа и извлечения злоумышленниками информации из системы.

Другой фактор риска – повреждение хранящейся в системе информации. Так как основная цель системы управления проектами – хранение и отображение структурированной информации о текущем состоянии выполнения проектов, целостность хранящейся в системе информации – первоочередная задача. Возникает необходимость обеспечения надежности хранения и совершение операций над данными таким образом, чтобы результат был предсказуем и корректен.

Также существует вероятность отказов или простоев в работе системы, когда пользователь не может оперативно получить доступ к требуемой информации или совершить необходимую операцию. Такая ситуация может негативно повлиять на выполнении проекта: требуемые задания не будут обновлены своевременно, что способно привести к простоям в работе над проектом.

Проект, отслеживаемый с помощью системы, может оказывать влияние на окружающую среду, жизнь и здоровье людей (если проект имеет отношение к опасному производству или, например, медицине). Таким образом неисправности в работе системы способны нанести вред не только организации и пользователям, использующим систему, но и посторонним людям и окружающей среде.

Таким образом, некоторые особенности работы реализованной системы способны оказать существенное негативное влияние на пользователей, а также опосредованное влияние на окружающую среду и общество, что вынуждает рассмотреть наиболее возможные ситуации некорректной работы системы для выявления слабых мест и проработки вариантов их устранения. Далее рассматриваются некоторые такие ситуации и возможные пути решения.

5.3 Ошибки в логике работы системы

Ошибки в логике работы системы включают в себя такие ситуации, когда система ведет себя неадекватно в ответ на действия пользователя. Например, при добавлении нового задания в Проект №1 задание добавляется в Проект №2. Или, например, другой случай: при присвоении заданию статуса «в работе» в результате ему присваивается другой статус, например, «выполнено».

Ошибки в логике работы системы могут быть вызваны, в первую, очередь, ошибками в структуре базы данных и в коде серверной части приложения.

Подобные ошибки способны нарушить структуру заданий в проекте, состав пользователей проекта и так далее. В результате нарушается целостность информации в системе и, как следствие, участники проекта получают недостоверную информацию о его текущем состоянии.

Типичные ситуации, которые могут возникнуть в результате таких ошибок:

Некоторые задания не будут выполнены. В результате не все части проекта будут выполнены вовремя (до обнаружения ошибки в системе), что приведет к задержке выполнения проекта, и, как следствие, финансовым потерям организации.

Задания будут выполнены несколькими участниками одновременно. Подобная ситуация приведет к неэффективному использованию трудовых ресурсов организации.

Некоторые участники не получат доступ к информации о проекте. Это приведет к тому, что появятся барьеры в коммуникации – не вся актуальная информация о проектах будет доходить до участников своевременно. Потребуется дополнительное время для того, чтобы оповестить администратора проекта о неисправности и заново добавить участника в проект.

Для предотвращения подобных ошибок на стадии реализации проводится исчерпывающее тестирование результатов совершения различных операций с базой данных. При выявлении ошибки вносятся изменения в структуру базы данных или в запрос к базе данных, с помощью которого проводится ее модификация.

5.4 Ошибки в пользовательском интерфейсе

Пользовательский интерфейс – это основное средство взаимодействия пользователя с системой. Интерфейс включает в себя совокупность элементов управления, с помощью которых пользователь вводит данные и посылает команды, элементов вывода информации, а также способов взаимодействия пользователя с этими элементами.

Правильно спроектированный и разработанный пользовательский интерфейс облегчает работу в системе, позволяет ускорить выполнение различных операций за счет того, что они становятся интуитивно понятными пользователю.

Пользовательский интерфейс, в котором присутствуют ошибки, замедляет работу пользователя в системе, так как ему постоянно приходится совершать лишние действия, которые можно было бы упростить, объединить в одно или вовсе исключить.

Ошибки в интерфейсе ухудшают опыт взаимодействия пользователя с системой, что может приводить к раздражительности, усталости, негативным эмоциям, что в свою очередь, напрямую сказывается на производительности труда и количестве совершаемых пользователем ошибок.

Далее перечислены некоторые примеры того, как можно допустить ошибки при разработке интерфейса:

Отсутствие информационных сообщений, например, таких как: «неправильно введен пароль», «добавление задания не удалось», «пользователь успешно добавлен в проект» и так далее. При отсутствии обратной связи, пользователь будет тратить время на то, чтобы проверить успешно ли было совершено действие или нет и в случае ошибки – что стало ее причиной.

Неправильное расположение элементов. Наиболее часто используемые элементы всегда должны быть под рукой. Таким образом максимально сокращается время необходимое на то, чтобы воспользоваться нужными функциями системы.

Нелогичное поведение системы. Пример – порядок выбора элементов ввода с помощью клавиши Tab должен соответствовать их геометрическому расположению на странице, иначе пользователь вынужден будет потратить время на то, чтобы привыкнуть к необычному поведению элементов управления.

5.5 Утечка персональных данных

При работе в системе управления проектами неизбежны такие случаи, когда придется хранить какую-либо информацию об участниках, которая носит конфиденциальный характер. Такая информация при разглашении способна нанести вред лицу, к которому она относится.

К таким сведениям могут относиться, например, контактные данные человека (номер телефона, адрес электронной почты). Также это может быть информация о поле, возрасте человека, пароль, используемый для входа в систему (который также может быть использован для доступа к ряду других ресурсов) и различная другая информация.

При утечке персональных данных может пострадать ряд лиц. Это и тот человек, к чьим данным был осуществлен несанкционированный доступ, и работодатель, несущий ответственность за безопасность персональных данных сотрудника, и разработчик программного продукта.

В разработанной системе не предусмотрена возможность хранения особо личной информации, распространение которой может нанести серьезный вред человеку при утечке, такой, как информация о вероисповедании или состоянии здоровья, для хранения и обработки которой необходимо специальное разрешение.

Тем не менее, очевидна целесообразность обращения внимания на вопросы безопасности хранения данных о пользователях в системе.

5.6 Утечка данных компании

Работа в системе управления проектами подразумевает хранение информации об организации, выполняющей этот проект. Если эта информация попадет в руки конкурентов, это может привести к финансовым убыткам организации. Далее приведены некоторые данные, способные нанести вред при разглашении:

Информация о внутренней структуре организации: количество и состав отделов, количество сотрудников, занятых выполнением проекта и так далее.

Информация о ходе выполнения проекта. При несанкционированном доступе к системе возможна утечка таких данных, как сроки выполнения отдельных работ над проектом или временные рамки проекта в целом. Это может дать конкурентам, получившим эти данные, преимущество: они выпустят свой продукт раньше, чем компания, чьи данные были разглашены.

Информация о составе проекта. При получении такой информации конкуренты могут получить представление о том, из каких частей состоит разрабатываемый проект, что позволит им создать аналогичную систему или улучшить свою таким образом, чтобы она была более привлекательной.

В целом, если данные компании или ее сотрудников были разглашены, конкуренты получают над ней преимущество. Как следствие, возможность доступа посторонних лиц к проекту крайне нежелательна. В системе управления проектами должны быть предусмотрены различные механизмы защиты от доступа неавторизованных лиц. Такие средства включают в себя, например:

- Вход в систему с помощью логина и пароля;
- Проверку на то, авторизован ли пользователь при совершении им действий;
- Проверку на то, есть ли у конкретного пользователя права на совершение необходимого действия (например, добавлять или удалять новых пользователей из проекта может только администратор проекта).

Перечисленные выше средства защиты применяются в разработанной системе.

5.7 Потеря данных

Потеря данных или невозможность оперативного доступа к ним также является одной из проблем, которая способна нанести вред использующей систему организации. При потере данных потребуется дополнительное время и

усилия на то, чтобы восстановить их, если восстановление окажется принципиально возможным.

Если теряется информация о некоторых заданиях, то работник, ответственный за их выполнение, не сможет вовремя приступить к работе, что приведет к задержкам выполнения как его части работы, так и выполнения всего проекта в целом. При потере данных обо всем проекте потребуется огромное количество времени на восстановление актуального состояния информации о проекте.

Существует несколько причин, из-за которых возможна потеря информации или невозможность доступа к ней:

Ошибки в запросах к базе данных. Некоторые запросы к базе данных могут быть составлены некорректно, что в определенных ситуациях приведет к невозможности добавления или извлечения записей из нее. При этом пользователь не сможет получить доступ к требуемым ему данным. Также возможны ситуации, когда по причине некорректно составленного запроса произойдет непреднамеренное удаление записей из базы данных. Решение данной проблемы – тестирование всех запросов и определение того, как они влияют на содержание базы данных, определение ситуаций, при которых запросы могут выполняться некорректно.

DoS атаки (Denial of Service, отказ в доступе). Злоумышленники могут провести атаку на сервер, что приведет к потере доступа к нему. При этом пользователи, которые хотят получить доступ к системе в этом время, не смогут этого сделать. При этом организация понесет финансовые убытки из-за простоя в работе системы и не сможет контролировать выполнение проекта. Для решения этой проблемы можно использовать специализированные сервисы по защите от DoS атак, например, CloudFlare или incapsula.

Ошибка пользователя. Пользователь может непреднамеренно совершить нежелательную операцию, например, случайно удалить проект. Для решения этой проблемы нужно делать пользовательский интерфейс более понятным и

простым, а также защищенным от ошибок (например, использовать диалоговые окна подтверждения).

5.8 Заключение

В настоящем разделе были рассмотрены различные особенности разработанного программного продукта, которые влияют на использующих его лиц, безопасность окружающей среды и общества.

Рассмотрены основные ситуации неисправной работы системы, способные привести к потере данных, невозможности доступа пользователей к системе, а также утечке данных. Примеры таких ситуаций включают в себя, например, ошибки пользователей, ошибки при проектировании и разработке системы, которые приводят к искажению или потере данных о проекте, атаки злоумышленников.

Также рассмотрены некоторые способы предотвращения опасных ситуаций и устранения их последствий, такие как тестирование разработанной системы, использование защитных механизмов или использование специализированных сервисов для защиты системы от атак.

ЗАКЛЮЧЕНИЕ

Разработанная система позволяет систематизировать и визуализировать информацию о разнообразных проектах, позволяет одновременно работать над одним проектом нескольким пользователем. В то время как функционал системы уступает крупным профессиональным системам, таким как Microsoft Project, она может найти применение в тех случаях, когда бесплатный доступ, скорость работы и простота интерфейса играют большую роль, нежели богатый функционал. Например, система может использоваться в небольших организациях, работающих над несложными проектами.

Реализация системы в виде web-приложения позволила относительно простым путем добиться возможности одновременной работы нескольких пользователей над проектами, а также обеспечила кроссплатформенность. Web-приложения могут работать на любых операционных системах и устройствах, для которых существуют современные браузеры. Минусом данного подхода можно назвать тот факт, что возможности web-приложения ограничены средой исполнения – браузером.

В дальнейшем возможно расширение функциональных возможностей системы и оптимизация программного кода с целью увеличения эффективности работы, что позволит расширить область ее применения. Например, возможно внедрение системы управления версиями для хранения и манипулирования различными версиями документации, относящейся к проекту. Подобная функция позволит облегчить разработку, например, программного обеспечения, при которой с проектом связано большое количество документов и файлов с исходным кодом программ, для которых требуется контроль версий.

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.

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.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

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

2. Chrome Developer [Электронный ресурс]: MVC Architecture. URL: https://developer.chrome.com/apps/app_frameworks. – Свободный доступ. (дата обращения: 08.06.2016).

3. 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).

4. Manning Publications Co. [Электронный ресурс]: SPA Design and Architecture. URL: http://www-legacy.manning.com/scott2/SPA_MEAP_CH01.pdf. – Свободный доступ. (дата обращения: 08.06.2016).

5. Microsoft Developer Network [Электронный ресурс]: MVC Overview. URL: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx. – Свободный доступ. (дата обращения: 08.06.2016).

6. Microsoft Developer Network [Электронный ресурс]: Model-View-Controller. URL: https://msdn.microsoft.com/en-us/library/ff649643.aspx. – Свободный доступ. (дата обращения: 08.06.2016).

7. Tutorials Point [Электронный ресурс]: MVC Framework. URL: http://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. – Свободный доступ. (дата обращения: 08.06.2016).

8. CARE [Электронный ресурс]: Project Management Information Systems. URL: http://www.careclimatechange.org/files/toolkit/CARE_DME_Project.pdf. – Свободный доступ. (дата обращения: 08.06.2016).

9. Chandler [Электронный ресурс]: Project Management Methodology Guidelines. URL:

https://www.chandleraz.gov/Content/PM000PMMethodologyGDE.pdf. – Свободный доступ. (дата обращения: 08.06.2016).

10. Computer Science University of Toronto [Электронный ресурс]: Database Design. URL: http://www.cs.toronto.edu/~faye/343/w08/lectures/wk10/10_DBDesignStG-4up.pdf. – Свободный доступ. (дата обращения: 08.06.2016).

11. National Innovation Agency [Электронный ресурс]: Project Management. URL: http://www.adi.pt/docs/innoregio_pmanagement.pdf. - Свободный доступ. (дата обращения: 08.06.2016).

12. Проектная практика [Электронный ресурс]: Процессы управления проектами. URL: http://pmpractice.ru/knowledgebase/managment/keypoints/process/. – Свободный доступ (дата обращения: 08.06.2016).

13. Asciutto [Электронный ресурс]: Microsoft Project 2010 User Guide. URL: www.asciutto.com/project2010/Project2010_eBook.pdf. - Свободный достп (дата обращения 08.06.2016).

14. Wrike [Электронный ресурс]: Resources for Project Managers. URL: https://www.wrike.com/resources/projectmanagement/. – Свободный доступ. (дата обращения: 08.06.2016).

15. Eclipse [Электронный ресурс]: Database Schema Tutorial. URL: http://wiki.eclipse.org/images/2/28/Databaseschema-aperi.pdf. – Свободный доступ. (дата обращения: 08.06.2016).

16. CodeIgniter [Электронный ресурс]: CodeIgniter Documentation. URL: https://www.codeigniter.com/docs. – Свободный доступ. (дата обращения: 08.06.2016).

17. Php.net [Электронный ресурс]: Original MySQL API. URL: http://php.net/manual/en/book.mysql.php. – Свободный доступ. (дата обращения: 08.06.2016).

18. MySQL [Электронный ресурс]: MySQL Documentation. URL: http://dev.mysql.com/doc/. – Свободный доступ. (дата обращения: 08.06.2016).

19. Free Management Ebooks [Электронный ресурс]: SWOT Analysys. URL: http://www.free-management-ebooks.com/dldebk-pdf/fme-swot-analysis.pdf. – Свободный доступ. (дата обращения: 08.06.2016).