

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин,
комплексов, систем и сетей специального
назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ ПРИЛОЖЕНИЕ КОНТРОЛЯ
ВЫПОЛНЕНИЯ ЗАДАЧ

Студент Кузнецова Полина Сергеевна
(подпись, дата) (ФИО полностью)

Руководитель Слепцов Н.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент Васильев А.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично

(протокол заседания ГЭК от 19.06.2023 г. № 4)

Секретарь ГЭК

Гурин Е.И.
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

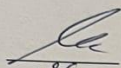
Факультет

Кафедра

Вычислительной техники

Вычислительная техника

«Утверждаю»
Заведующий кафедрой



Митрохин М.А.

«26» апреля 2023 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Кузнецова Полина Сергеевна гр. 18ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Программное приложение контроля выполнения задач».

Тема утверждена приказом ПГУ № от . 2023 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Слепцов Н.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Разработать программное приложение контроля выполнения задач.

Программное приложение контроля выполнения задач должно выполнять следующие функции: создание тестовых случаев; редактирование тестовых случаев; удаление тестовый случаев; изменение статуса тестовых случаев (созданный / в работе / готовый); просмотр всей доски с тестовыми случаями; закрепление тестового случая за определенным специалистом по тестированию; возможность отмечать основные этапы прохождения тестирования; использование шаблона для создания тестового случая; просмотр карточки тестового случая с подробной информацией о ней и проделанных этапах тестирования; написание комментариев к созданному тестовому случаю. Программное приложение контроля выполнения задач должно разрабатываться с помощью среды разработки Visual Studio Code. База данных должна разрабатываться для СУБД «PostgreSQL».

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать алгоритм работы программы, разработать диаграммы: прецедентов, активностей, последовательностей, объектов, классов, с последующим написанием программы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	26.04.2023 - 28.04.2023	
2.	Проектирование программного приложения	10	29.04.2023 - 30.04.2023	
3.	Разработка базы данных	20	31.04.2023 - 03.05.2023	
4.	Отладка и тестирование	10	04.05.2023 - 08.05.2023	
5.	Разработка экономической и бжд части	10	09.05.2023 - 15.05.2023	
6.	Оформление графической части	10	16.05.2023 - 23.05.2023	
7.	Оформление выпускной работы	30	24.05.2023 - 02.06.2023	

Дата выдачи задания 26 «04» 2023 года.

Руководитель дипломного проекта
Н.В.

(подпись, дата)

Слепцов

Задание к исполнению принял студент
П.С.

(подпись, дата)

Кузнецова

Работу к защите допустить
Декан факультета

(подпись, дата)

Фионова Л.Р.

Реферат

Пояснительная записка 120 листов, 12 рисунков, 21 таблиц и 5 приложений
ПРОГРАММНОЕ ПРИЛОЖЕНИЕ, КОНТРОЛЬ ВЫПОЛНЕНИЯ ЗАДАЧ,
ВЕБ-РАЗРАБОТКА, БАЗА ДАННЫХ, СПЕЦИАЛИСТ ПО
ТЕСТИРОВАНИЮ, НАЧАЛЬНИК ОТДЕЛА

Объектом исследования является процесс контроля выполнения задач специалистами по тестированию.

Цель работы – разработка программного приложения контроля выполнения задач.

В процессе работы необходимо выполнить анализ предметной области, сформулировать техническое задание для разрабатываемого программного приложения, разработать логическую и физическую модели данных для разработки базы данных.

В результате проделанной работы разработано программное приложение контроля выполнения задач, состоящее из базы данных и веб-приложения для работы с ней.

База данных спроектирована с использованием CASE-средства OpenModelShere 3.2. База данных разработана для СУБД PostgreSQL, приложение реализовано в среде разработки Visual Studio Code.

					ПГУ1.090501.05.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Кузнецова П.С.					
Провер.		Слепцов Н.В.					
Н. Контр.		Бычков А.С.					
Утверд.		Митрохин М.А.					
					Программное приложение контроля выполнения задач		
					Пояснительная записка		
					Лит.	Лист	Листов
						2	120
					ФВТ, каф. ВТ		
					гр.18ВОЭ1		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1. Анализ предметной области	8
1.1 Описание предметной области.....	8
1.2 Анализ аналогов.....	12
1.3 Цели и задачи разрабатываемого программного приложения	16
1.4 Анализ целевой аудитории	16
1.5 Формирование опроса о функциональной работе приложения.....	17
2. Техническое задание.....	24
2.1 Назначение разработки	25
2.2 Требования к программе	25
2.2.1 Требования к функциональным характеристикам	25
2.2.2 Требования к надежности	26
2.2.3 Требования к составу и параметрам технических средств	27
2.3 Требования к программной документации	27
2.4 Стадии и этапы разработки.....	27
2.5 Порядок контроля и приемки	28
3. Обоснование целесообразности выбранных заданных средств разработки. 30	
3.1 Язык разработки JavaScript.....	30
3.1.1 Библиотека React	31
3.2 Язык разработки HTML5	32
3.3 Язык разработки CSS	32
3.4 Сборщик модулей Webpack.....	33
3.5 CASE-средство OpenModelSphere 3.2.....	34
3.6 СУБД PostgreSQL	34
3.7 Среда разработки Visual Studio Code.....	35
4. Разработка программного приложения	37
4.1 Логическое проектирование базы данных	37
4.2 Физическое проектирование базы данных.....	42

4.3	Описание функций программы	45
4.4	Методика испытаний программы	47
5.	Оценка затрат на разработку и внедрение программного приложения контроля выполнения задач.	49
5.1	Организация производства разработки программного приложения.....	49
5.2	Расчет себестоимости программного обеспечения для программного приложения контроля выполнения задач.....	52
5.3	Оценка эффективности программного приложения контроля выполнения задач	58
5.4	Анализ рисков при разработке и внедрении программного приложения контроля выполнения задач	59
6.	Общие принципы обеспечения безопасности на рабочем месте оператора программного приложения контроля выполнения задач.	61
6.1	Актуальность вопросов безопасности на рабочем месте	61
6.2	Требования СанПиН при работе с компьютером.....	62
6.3	Требования безопасности при работе с электроприборами.....	62
6.4	Влияние ПЭВМ на физиологию человека.....	63
6.5	Техника безопасности при работе с персональным компьютером	64
	Заключение	65
	Список использованных источников	66
	Приложение А – UML-диаграммы.....	68
	Приложение А.1	69
	Приложение А.2	70
	Приложение А.3	71
	Приложение А.4.....	72
	Приложение А.5.....	73
	Приложение Б – SQL-скрипт	74
	Приложение В – Результаты испытаний	80
	Приложение Г – Листинг программы	86
	Приложение Д – Презентация	109

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ГОСТ – государственный стандарт

НФ – нормальная форма

ПЭВМ – персональная электронно-вычислительная машина

ЭВМ – электронно-вычислительная машина

FK (Foreign Key) – внешний ключ

ID (Identifier) – идентификатор объекта

PK (Primary Key) – первичный ключ

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ1.090501.05.001 ПЗ	Лист
						5
Из	Лист	№ докум.	Подпись	Дата		

Введение

У современного человека установлено огромное количество программ на компьютере или ноутбуке. Начинающих пользователей обладает 20-30 программами, а вот у тех, чья работа связана с работой за компьютером, может достигать 70, а то и большего количества. Все они действительно необходимы, так как в настоящее время автоматизированные системы получили широкое распространение именно после передачи программному обеспечению большого количества рутинных и сложных задач. Что, во-первых, делает их более простыми для человека, а во-вторых, легко выполняемыми.

Рассмотрим программные приложения с точки зрения эффективности и полезности использования. Большая их часть основана на базах данных, что удобно для хранения и работы с информацией. База данных – это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, не зависящая от прикладных программ.[1] Эти данные относятся к определенной предметной области и сделаны так, что используются для решения многих задач разными пользователями.

Но помимо работы в программных приложениях, их еще используют как систему контроля выполнения задач сотрудниками. Как известно, персонал – движущая сила компании, именно от него зависит репутация и успешность предприятия. Без качественного контроля выполнения поставленных задач достичь высоких результатов очень сложно.

Главным условием реализации контроля над сотрудниками является постоянство и отсутствие чрезмерности. Можно выделить три основных показателя эффективности:

- 1) Количество – объем нормы, сверхнормативная выработка, дополнительные задачи и поручения;
- 2) Качество – отсутствие ошибок и жалоб клиентов, соответствие установленным регламентам;

					ПГУ1.090501.05.001 ПЗ	Лист
						6
Из	Лист	№ докум.	Подпись	Дата		

3) Сроки – продолжительность выполнения задач.[2]

Одним из крупных и развивающихся направлений в сфере контроля выполнения задач является тестирование. Специалист по тестированию, или QA инженер – человек, который проверяет, как работает приложение или программа. В их задачи входит поиск ошибок, отслеживание корректности работы программы, согласно заявленным требованиям, проверка работы приложений на разных устройствах и операционных системах, а также составление отчетной документации по проделанной работе.

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.[3]

Высокая востребованность IT сферы сейчас сделала работу специалистов по тестированию еще более необходимой. Ни один новый программный продукт не обходится без тестирования. А значит, и наличие качественного программного обеспечения, для комфортной работы специалистов необходимо.

В представленной работе описывается программное приложение контроля выполнения задач, позволяющее упростить и автоматизировать процесс работы специалистов по тестированию.

Программное приложение контроля выполнения задач включает в себя базу данных и программное веб-приложение обработки данных. При проектировании базы данных использовалось CASE - средство Open ModelSphere 3.2 для СУБД PostgreSQL, для реализации приложения была применена среда разработки Visual Studio Code.

					ПГУ1.090501.05.001 ПЗ	Лист
						7
Из	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Описание предметной области

Программное приложение контроля выполнения задач предназначено для оптимизации работы специалистов по тестированию на их локальном ПК, а также для упрощения контроля над сотрудниками от руководителей.

Работа тестировщика может развиваться по двум направлениям – это автоматизированное и ручное или по-другому поведенческое тестирование. Первые занимаются тестированием приложения с помощью программного обеспечения, для их работы требуется более глубокое изучение языков программирования и знание алгоритмов работы программы. При этом, автоматизированное тестирование подразумевает авто-запись своей работы, следовательно, тестировщику достаточно только загрузить полученный файл в недельный отчет. Вторые вручную имитируют действия пользователя приложения, часто с использованием метода «черного ящика». Данный метод является стратегией, при которой тестирование основывается исключительно на требованиях, но не известно, как устроена система внутри, следовательно, производится тестирование внешнего интерфейса приложения или системы. Поведенческое тестирование является самым простым способом для оценки качества приложения. При планировании ручных тестов специалист по тестированию учитывает технические требования, и должен проверить приложение на совпадение заявленных в техническом задании функций с фактической их реализацией.

Для того чтобы работать тестировщиком необходимо обладать рядом профессиональных компетенций. Во-первых, специалист должен изучить основу теории тестирования. Классификации тестирования, методы и инструменты, а также создание планов тестирования – этот набор является базовым набором знаний, с которого тестировщики начинают свою работу. Вторым важным критерием является умение работать с командной строкой и

					ПГУ1.090501.05.001 ПЗ	Лист
						8
Из	Лист	№ докум.	Подпись	Дата		

браузерами. Если рассматривать автоматизированное тестирование, то дополнительным критерием является знание таких инструментов как: HP-UFT, Selenium, Sahi и др. Знание основ программирования на разных языках, умение работать с базами данных. Важно отметить, что специалисты по тестированию должны обладать и софт-скилами, к ним можно отнести: умение анализировать большой объем информации, критическое мышление и внимательность к мелочам.

Тестирование является неотъемлемой частью разработки любого продукта, так как главной его задачей является предоставление наиболее полной информации о качестве и степени готовности продукта для всех участников команды. А работа специалистов по тестированию начинается уже на ранних этапах разработки, ведь по частям проверять удобнее, нежели сразу готовый проект, так как выявить место ошибки оказывается проще. Большой блок работы тестировщика начинается в момент объединения фронт части с бэк частью. Именно в этот момент необходимо протестировать качество работы функций в рамках выбранного интерфейса. Параллельно с тестированием на каждом этапе специалисты фиксируют и дополняют тест планы. Тест план – это определенный перечень необходимых к выполнению действий для проверки выбранных функций приложения. После чего, по составленному плану тестирование повторяется, так как при добавлении новых функций старые могут измениться.

Наиболее распространенным форматом отчетности специалистов по тестированию является набор планов тестирования функций приложения. На основании данного отчета можно выявить объем проделанной сотрудником работы и текущее состояние разрабатываемого приложения. Для этого достаточно посмотреть, сколько тест планов изменили свой статус на той или иной сборке приложения.

Финальным показателем, что приложение готово к внедрению и эксплуатации является заверченный статус на всех ранее созданных тестовых

					ПГУ1.090501.05.001 ПЗ	Лист
						9
Из	Лист	№ докум.	Подпись	Дата		

планах, а также отсутствие новых тестовых планов на финальной сборке.

Существенным минусом работы специалиста по тестированию является система для формирования отчетности. В большинстве компаний тестировщикам необходимо выполнять фиксацию своей работы в течение недели сначала в текстовых редакторах, а после по очереди заполнять карточки планов тестирования на определенном компьютере, который имеет необходимое для этого ПО. Во-первых, это создает очередь, ведь в один день всему кабинету необходимо заполнить информацию. Во-вторых, хранить и переносить от рабочего места до необходимого компьютера, созданный файл необходимо на флеш-носителе, следовательно, есть риск его повреждения или потери.

На основе описания предметной области можно сформировать UML-диаграмму вариантов использования «Как есть». Полученная диаграмма представлена на рисунке 1.

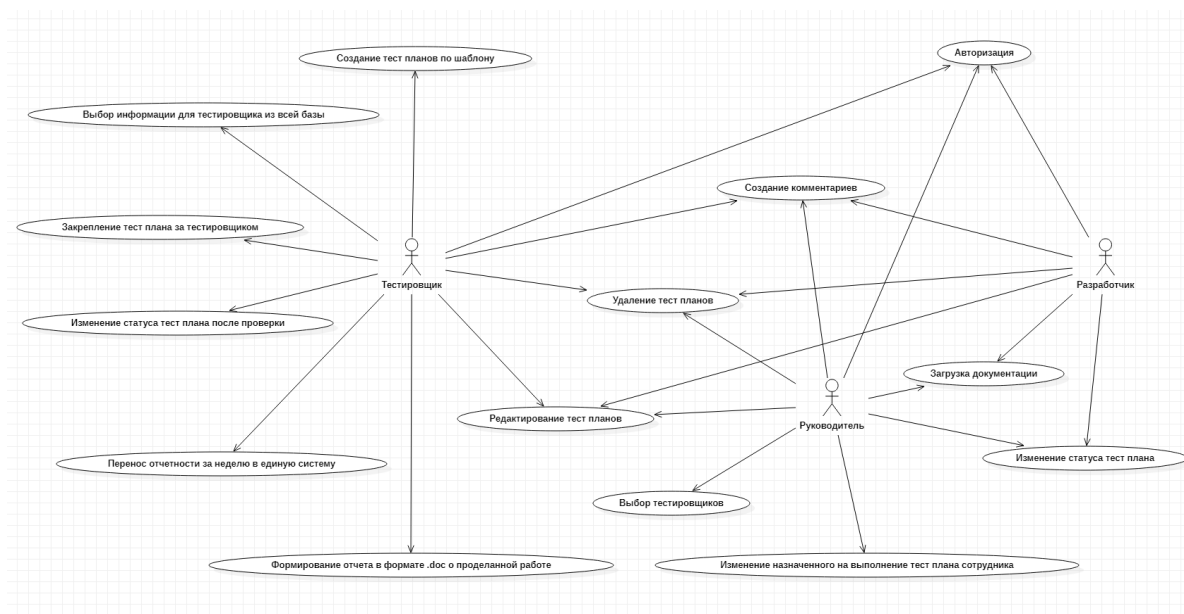


Рисунок 1 – Диаграмма вариантов использования «Как есть»

При использовании приложения данного вида для первого входа необходимо осуществить регистрацию под одним из видов пользователей: разработчик, руководитель, тестировщиков регистрирует руководитель самостоятельно, передавая им потом пароль для входа в приложение.

Данное приложение обладает рядом индивидуальных, а также общих функций для каждого из типов пользователей.

Для разработчика все функции являются общими с руководителем. Загрузка документов предполагает загрузку технической документации в приложение с целью ее сохранения, а также часть этой документации позднее используется тестировщиком при выполнении его работы. Изменение статуса тест плана подразумевает под собой переход карточки тест плана из состояния созданной в состояние в работе, а после завершенное.

Для руководителя персональной функцией является выбор тестировщиков, которая означает, что из авторизованных ранее пользователей-тестировщиков руководитель может закрепить за собой ряд этих пользователей с целью дальнейшего контроля их работы. Еще одной персональной функцией является изменение назначенного на выполнение тест плана сотрудника, это означает, что руководитель может переназначить ответственного за выполнение той или иной задачи.

Тестировщик обладает следующими индивидуальными функциями – создание тест планов по шаблону, значит, что все создаваемые им тест планы будут обладать единым видом и блоками информации. Выбор информации из всей загруженной ранее базы, предполагает анализ всей имеющейся технической документации с последующим выбором необходимой для выполнения конкретного тест плана. Закрепление тест плана за собой, означает, что при создании новой карточки, тестировщик может назначить ответственным за ее выполнение только себя. Изменение статуса тест плана после проведения проверки гарантирует, что тестировщик провел необходимую проверку и только после этого сможет изменить статус выбранной карточки тест плана. Помимо этого специалисту по тестированию необходимо параллельно заполнять отчет в виде документа с расширением .doc о проделанной за неделю работе, после чего полученный документ необходимо перенести в порядке очереди в единую систему, используемую в компании.

					ПГУ1.090501.05.001 ПЗ	Лист
						11
Из	Лист	№ докум.	Подпись	Дата		

Важно заметить, что в данном приложении есть ряд функций, которые используются сразу тестировщиком, руководителем и разработчиком. К этим функциям относится создание комментариев под карточкой тест плана, что создает прямой диалог между сотрудниками компании. Также если создавать план тестирования может только тестировщик, то редактировать и удалять его могут все вышеперечисленные пользователи, это означает, что любой из сотрудников может внести правки в содержание карточки тест плана, а при ее неактуальности совсем удалить.

1.2 Анализ аналогов

Рассмотрим существующие программы, предназначенные для контроля выполнения задач. Среди огромного количества можно выделить ряд наиболее функциональных и понятных для использования: Jira, Moon.team, Flowlu, Kaiten, Advanta и Штаб.

Jira – инструмент управления проектами, с помощью которого возможно оптимизировать работу команды, отслеживать процессы с момента их запуска до завершения;

Moon.team – сервис для совместной работы команд, с помощью которого возможно выстроить процессы в коллективе и оптимизировать их;

Flowlu – платформа, с помощью которой возможно управлять проектами, задачами и финансами;

Kaiten - единое пространство, с помощью которого возможно выполнять совместные работы над проектами и задачами;

Advanta – корпоративная система, с помощью которой возможно управлять проектами, также есть поддержка совместной работы;

Штаб – единое пространство, с помощью которого налаживаются все рабочие процессы (управление проектами; временем, заложенным на данный проект; задачами, необходимыми к выполнению).

					ПГУ1.090501.05.001 ПЗ	Лист
						12
Из	Лист	№ докум.	Подпись	Дата		

На основе открытых источников были выявлены основные функции этих приложений и составлена сравнительная таблица. Результат анализа представлен в таблице 1.

Таблица 1 – Сравнение функционала выбранных приложений

Функция	Jira	Moon.team	Flowlu	Kaiten	Advanta	Штаб
База знаний	+	+	+	-	+	-
Отчеты	+	+	+	+	+	+
Редактор задачи	+	+	+	+	+	+
Текстовый редактор в задачах	+	+	+	-	+	+
Добавление файлов к задачам	+	+	+	+	+	+
Фильтры для показа задач	+	+	+	+	+	+
Паспорт проекта	+	-	+	-	+	-
Учет времени	+	+	+	+	-	+
Шаблон для задач	+	+	+	+	+	+
Уведомления об изменениях в проекте	+	+	+	+	+	+
Напоминание о задачах	+	+	+	+	+	+
Интеграция почты	+	-	+	+	+	-
Импорт файлов	+	-	+	+	-	-
Календарь задач	+	+	+	-	+	+

Функция	Jira	Moon.team	Flowlu	Kaiten	Advanta	Штаб
Файловое хранилище	+	+	+	-	-	+
Индикатор просмотра задачи	+	-	-	-	-	-
Сортировка	+	-	-	-	+	+
Поиск	+	-	-	-	-	+
Справка	+	-	-	-	-	-
Доступность среды	-	+	+	+	+	+

Важно отметить, что Jira настраивается под потребности отдельного предприятия индивидуально. Она обладает базовым функционалом, а также дополнительными плагинами, которые можно поставить отдельно. Необходимо подчеркнуть функцию отслеживания и учета времени с помощью, которой руководитель может следить за нагрузкой на сотрудников и затраченным на задачу временем. Безусловно, Jira является прекрасным инструментом, но на основе санкций с марта 2022 года Atlassian – разработчик Jira, начала отключать российские аккаунты и предупредила о приостановке продажи своих новых продуктов в России.

На основе вышесказанного рассмотрим российские аналоги инструмента Jira.

Особенностью сервиса Moon.team является наличие в интерфейсе фильтрации, скачивании отчетов и сортировки. В дополнение к этому, есть таблица сформированных отчетов для удобства пользователя. Также для пользователя является удобным быстрый выбор дедлайна из предложенного списка, при этом есть возможность выбора даты самостоятельно через календарь. Отрицательной особенностью данного сервиса является отсутствие в интерфейсе подписей у иконок, часть из которых не вполне очевидны.

Использование простых и понятных иконок делает сервис проще, но отчасти менее понятно для пользователя. Время на задачу указывается в формате ЧЧ:ММ (часы : минуты), что существенно неудобно если на выполнение выделяются дни / недели / месяцы.

На платформе Flowlu есть различные виды отчетности: проекты, финансы, простые задачи и их фильтрация. Данные представляются в табличном виде, что не всегда удобно. Меню навигации представлено в виде бокового меню, и это удобное решение, так как оно всегда доступно. Одним из существенных минусов Flowlu является задваивание информации, на примере выбора всех задач сначала с помощью чекбокса, а после кнопкой «Выбрать все». Также в бесплатной версии платформы не предусмотрен учет времени, что очень неудобно как для сотрудника, так и для руководителя. Не столь значительным недостатком является поле ввода даты, где ее необходимо выбрать из ограниченного выпадающего списка заданных интервалов, а непривычным пользователю способом через календарь.

Единое пространство Kaiten дает возможность пользователю выбор из 11 отчетов, каждый из которых имеет собственные фильтры, часть отчетов доступна для скачивания. Недостатком же является отсутствие подсказок, как следствие, трудность восприятия терминов и слов, значения которых пользователь может не знать.

Корпоративная система Advanta не имеет отдельной страницы с отчетами, в отличие от других сервисов. Данная функция реализована на боковой панели дашборда, где из выпадающего списка пользователь добавляет отчет и после настраивает его параметры. Существенным недостатком системы является отсутствие импорта файлов со сторонних ресурсов, следовательно, и отсутствие файлового хранилища.

Особенностью единого пространства Штаб является наличие трех видов отчетов, которые расположены на боковой панели меню. Также внутри отчета пользователь может настраивать поиск и фильтрацию, после чего скачивать

					ПГУ1.090501.05.001 ПЗ	Лист
						15
Из	Лист	№ докум.	Подпись	Дата		

отчет. Дополнительным плюсом является наличие карточки задачи, которая находится всегда на виду при работе с задачей.

Недостатками Штаба являются:

- отсутствие функции сброса всех параметров, что замедляет работу пользователя, ведь ему приходится очищать все поля самостоятельно;
- при отсутствии необходимых данных для формирования отчета, нет указаний, чего именно данных не хватает;
- отсутствие интеграции с почтовой системой.

На основе проведенного анализа существующих программных приложений контроля выполнения задач, я сделала вывод об актуальности создания нового приложения. В нем будут собраны наиболее используемые специалистами по тестированию.

1.3 Цели и задачи разрабатываемого программного приложения

Цели автоматизации деятельности специалистов по тестированию:

- увеличение скорости фиксации работ;
- обеспечения всей деятельности сотрудника по планированию тестирования и созданию тестовых случаев;
- структуризация деятельности специалистов.

В результате внедрения программного приложения планируется достичь ряда задач, а именно:

- повысить эффективность работы сотрудников;
- снизить лишнюю нагрузку на сотрудников.

1.4 Анализ целевой аудитории

На основании того, что все вышеописанные программные продукты предназначены для работы разработчиков или полной команды проекта, продуктов подходящих только для локальной работы специалистов по

					ПГУ1.090501.05.001 ПЗ	Лист
						16
Из	Лист	№ докум.	Подпись	Дата		

тестированию не было найдено. Как следствие, разработка программного приложения контроля выполнения задач является актуальной как для частных компаний, так и для государственных компаний, у которых есть штат специалистов по тестированию.

Делая вывод, основной целевой группой являются специалисты по тестированию частных и государственных компаний, но за счет простоты и универсальности рабочего пространства приложения, его могут использовать все, кому необходимо фиксировать этапы выполнения работы.

1.5 Формирование опроса о функциональной работе приложения

На основании анализа аналогов был сформирован опрос для конкретной целевой группы – специалистов по тестированию. Опрос состоит из 7 вопросов с вариантами ответов. А именно: Ваш пол (Женский / Мужской), Ваш возраст (До 20 лет / 20-27 лет / 27-35 лет / 35+ лет), Ваш опыт работы специалистом по тестированию (До 1 года / 1-3 года / Более 3 лет), Программный продукт для контроля Вашей работы, используемый вашей компанией Вас устраивает? (Точно да / Скорее да, чем нет / Скорее нет, чем да / Точно нет), Если в предыдущем вопросе Вы ответили «Точно нет / Скорее нет, чем да», то что именно Вас не устраивает? (Интерфейс / Цветовая гамма / Сложность в работе / Избыток не используемых функций / Все устраивает), Важна ли для Вас цветовая гамма интерфейса? (Да / Нет / Не влияет на работу), Какие по Вашему мнению функции являются самыми необходимыми для специалиста по тестированию? (Создание / редактирование / удаление багов / Установка статусов / Возможность видеть других пользователей / Подробная информация о пользователе / Формирование багов в карточки / Запароленный доступ / Свободный доступ / Наглядное представление карточек по статусам / Дополнительная загрузка документов / Возможность закрепить задачу за конкретным человеком / Краткая информация о пользователе /

					ПГУ1.090501.05.001 ПЗ	Лист
						17
Из	Лист	№ докум.	Подпись	Дата		

Расширенное меню / Возможность весь ход проекта / Выбор багов из списка).
Опрос прошли 30 респондентов, и были получены следующие результаты, представленные на рисунках 2-8:

1) 2/3 всех опрошенных – мужчины, это свидетельствует о том, что большая часть работающих в данной профессии мужчины;

Ваш пол

30 ответов

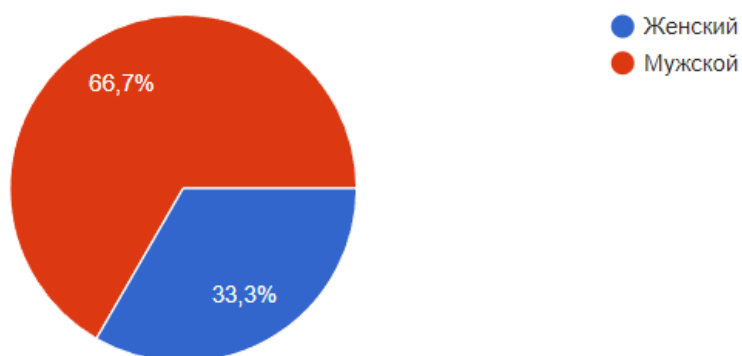


Рисунок 2 – Результаты ответа на первый вопрос

2) большая часть опрошенных – молодежь в возрасте до 35 лет, работающая в данной профессии;

Ваш возраст

30 ответов

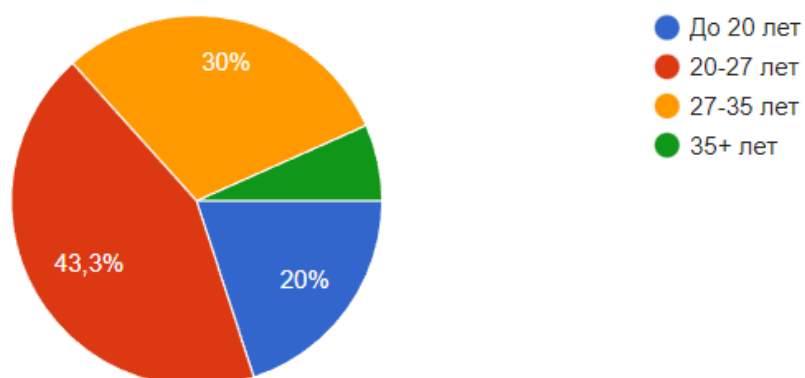


Рисунок 3 – Результаты ответов на второй вопрос

3) больше 40% опрошенных работают от 1 до 3 лет, а 30 % уже более 3 лет, остальные до 1 года;

Ваш опыт работы специалистом по тестированию

30 ответов

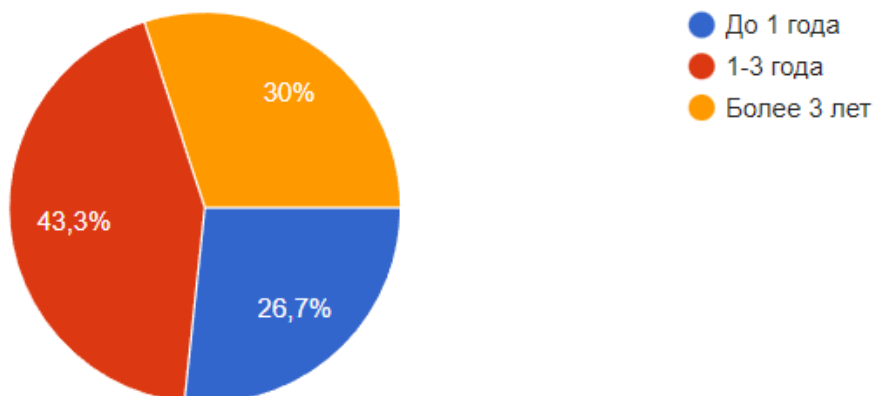


Рисунок 4 – Результаты ответов на третий вопрос

4) более 60% респондентов не устраивает действующее приложение;

Программный продукт для контроля Вашей работы, используемый вашей компанией Вас устраивает ?

30 ответов

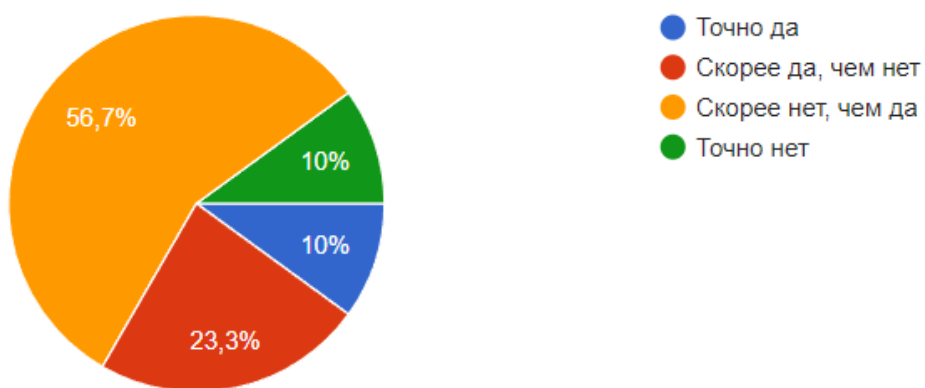


Рисунок 5 – Результаты ответов на четвертый вопрос

5) 80% респондентов считают, что приложения, в которых они работают, имеют избыток не используемых функций, более 50% выделяют сложность при работе и неудобный интерфейс, для 36,7% важна цветовая гамма приложения;

Если в предыдущем вопросе Вы ответили «Точно нет / Скорее нет, чем да», то что именно Вас не устраивает?

 Копировать

30 ответов

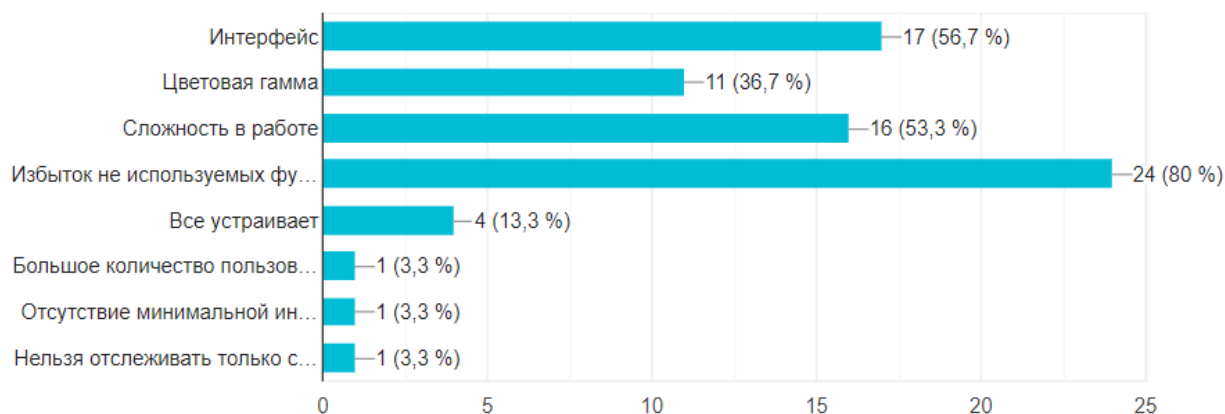


Рисунок 6 – Результаты ответов на пятый вопрос

6) 90% респондентов считают цветовую гамму интерфейса приложения важным аспектом или же не влияющим на работу. Только 10% считают, что это не важно;

Важна ли для Вас цветовая гамма интерфейса?

30 ответов

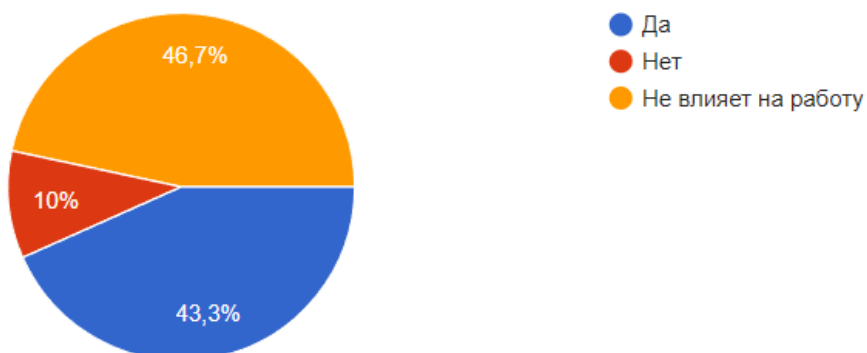


Рисунок 7 – Результаты ответов на шестой вопрос

7) больше 60% считают самыми необходимыми функциями это создание/ редактирование/ удаление багов и свободный доступ к приложению без пароля. Более 50% считают необходимым установку статуса багов и формирование их в виде карточек, почти половина респондентов считают важным закрепление задачи за конкретным специалистом. Свыше 30% выявили важными функциями наглядное представление карточек по статусам и наличие краткой информации о пользователе.

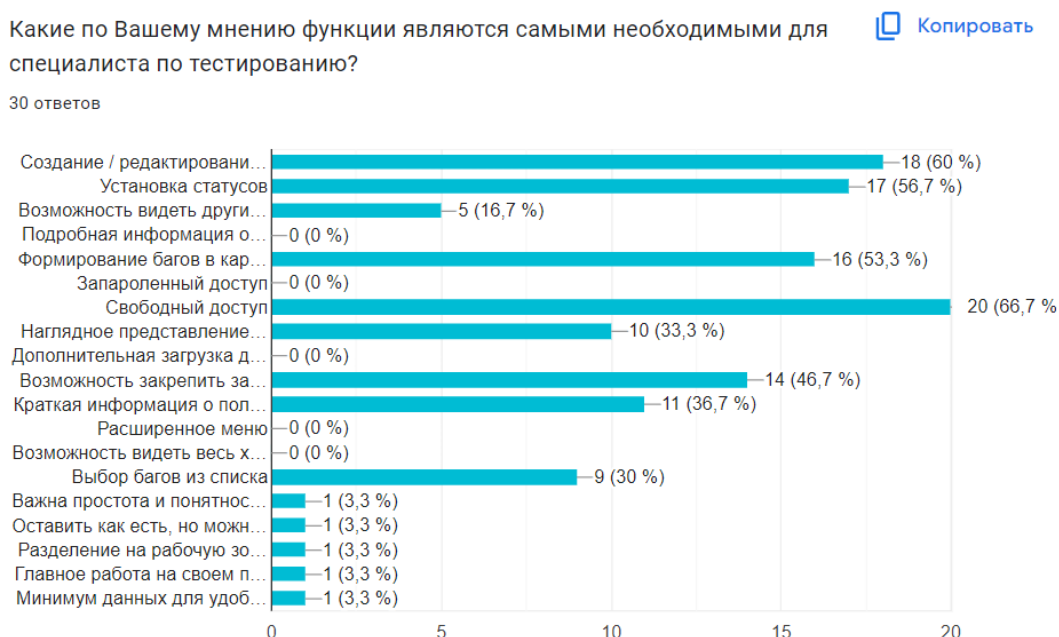


Рисунок 8 – Результаты ответов на седьмой вопрос

На основании полученных результатов была разработана UML-диаграмма вариантов использования программного приложения контроля выполнения задач «Как должно быть». В ней отражен весь используемый специалистом по тестированию функционал. Полученная диаграмма представлена на рисунке 9.

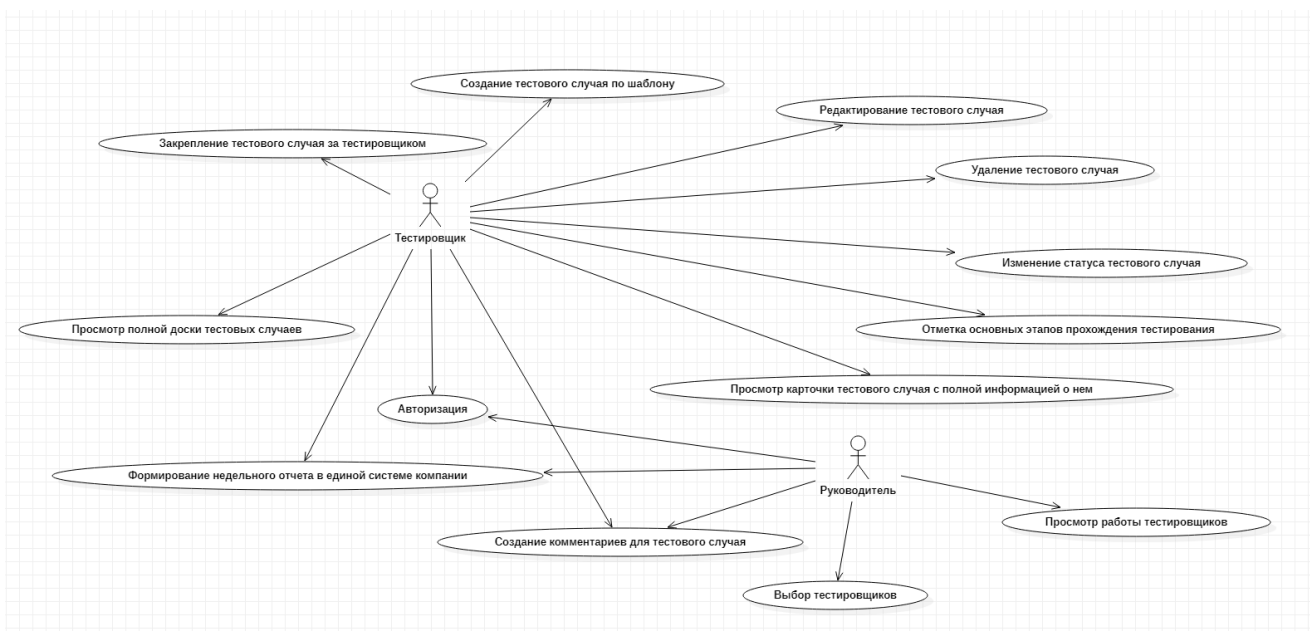


Рисунок 9 – Диаграмма вариантов использования «Как должно быть»

Для входа в приложение, необходимо пройти первичную регистрацию, после чего логин и пароль вносятся в таблицу базы данных, и пользователь при последующем входе должен просто ввести ранее придуманный логин и пароль. Зарегистрироваться могут как тестировщики, так и их руководители.

Данное приложение разрабатывается для непосредственной работы специалистов по тестированию, именно поэтому руководитель обладает двумя персональными функциями, а именно выбор тестировщиков, которая означает, что из авторизованных ранее пользователей-тестировщиков руководитель может закрепить за собой ряд этих пользователей. Вторая функция руководителя – это просмотр работы тестировщиков, которая подразумевает, что руководитель в течение недели параллельно с работой тестировщика может смотреть, что сотрудник выполнил. При этом важно отметить, что доступа к редактированию и удалению карточек тест планов у руководителя нет.

Для тестировщиков большинство функций является индивидуальными, а именно, создание тестового случая по шаблону, это означает, что все

карточки, во-первых, будут единого образца, во-вторых, могут создаваться только специалистом по тестированию и заноситься в базу данных. Редактирование тестового случая, предполагает внесение изменений в карточку тест плана, с последующим занесением этих же изменений в базу данных. Удаление тестового случая – это означает, что данный план тестирования успешно пройден и больше не является актуальным, после удаления тестового случая из приложения информация о нем стирается и в базе данных. Изменение статуса тестового случая и отметка основных этапов прохождения тестирования означает, что тестирующий во время своей работы может перевести карточку из одного состояния в другое (примеры состояний: созданная, в работе, завершенная). После формирования карточки с тест планом специалист может посмотреть полную карточку со всей информацией, это необходимо для повторной проверки выполняемой работы. Затем тестирующий закрепляет созданный тест план за собой, чтобы при дальнейшей работе легко идентифицировать свои тест планы. Еще одной функцией является просмотр всей доски тестовых случаев, это необходимо, чтобы при занесении своей работы не оказалось дублирующих друг друга карточек.

Несмотря на то, что руководитель не может редактировать и удалять карточки тест планов, он, как и тестирующий может создавать комментарии для тестового случая, который будет виден как ему так и создатели плана тестирования. Это необходимо для упрощения работы тестирующего, так как любые изменения в карточке, будь то название или статус, могут привести к ее потере среди всех существующих задач. Неотъемлемой частью любой работы является составление отчетов о проделанной работе, данный отчет будет оформляться только руководителем в единой системе компании, так как работу его подчиненных он может проверить в любой момент.

На основании полученных результатов анализа UML-диаграммы «Как должно быть» было сформировано техническое задание разрабатываемого программного приложения контроля выполнения задач, где были учтены как

					ПГУ1.090501.05.001 ПЗ	Лист
						23
Из	Лист	№ докум.	Подпись	Дата		

функциональные требования целевой аудитории, так и требования к интерфейсу и цветовой гамме.

Таким образом, задача разработки программного приложения контроля выполнения задач является актуальной.

Программное приложение должно включать в себя базу данных и программное веб-приложение обработки данных. База данных может быть использована для хранения, редактирования, удаления информации о тестовых случаях, вносимых специалистом по тестированию, а также информация о самих специалистах по тестированию и руководителях. Программное веб-приложение включает в себя приложение для работы с базой данных.

Внедрение программного приложения контроля выполнения задач позволит сократить время выполнения работы, улучшить качество выполняемой работы, и повысить производительность труда специалистов по тестированию, упростить ведение и анализ тестовых случаев.

					ПГУ1.090501.05.001 ПЗ	Лист
						24
Из	Лист	№ докум.	Подпись	Дата		

2 Техническое задание

2.1 Назначение разработки

Программное приложение контроля выполнения задач должно быть предназначено для автоматизации основных направлений деятельности специалиста по тестированию

Цель данного проекта – разработка базы данных и программного веб-приложения для программного приложения контроля выполнения задач.

2.2 Требования к программе

2.2.1 Требования к функциональным характеристикам

Программное приложение контроля выполнения задач должно обеспечивать выполнение следующих функций:

- создание тестового случая;
- редактирование тестового случая;
- удаление тестового случая;
- изменение статуса тестового случая (созданный/ в работе/ готовый);
- просмотр всей доски с тестовыми случаями;
- закрепление тестового случая за определенным специалистом;
- возможность отмечать основные этапы прохождения тестирования;
- использование шаблона для создания тестового случая;
- просмотр карточки тестового случая с подробной информацией о ней и проделанных этапах тестирования;
- написание комментариев к созданной задаче.

Входными данными для системы должны быть:

- сведения о ФИО специалиста по тестированию;
- сведения о поле специалиста по тестированию;
- сведения о возрасте специалиста по тестированию;
- сведения о номере телефона специалиста по тестированию;

					ПГУ1.090501.05.001 ПЗ	Лист
						25
Из	Лист	№ докум.	Подпись	Дата		

- сведения о должности специалиста по тестированию;
- сведения о ФИО руководителя;
- сведения о поле руководителя;
- сведения о возрасте руководителя;
- сведения о номере телефона руководителя;
- сведения о должности руководителя;
- сведения о названии тестового случая;
- сведения о дате создания тестового случая;
- сведения об описании тестового случая;
- сведения о владельце тестового случая;
- сведения о комментариях к тестовому случаю.

2.2.2 Требования к надежности

Приложение должно работать без прерываний и ошибок. В случае если пользователь совершает неверные действия, система должна уведомить его об этом.

Программное приложение должно обеспечивать контроль вводимых данных. В случае ввода некорректных данных, или не ввода данных совсем приложение должно информировать об этом пользователя.

При вводе входной информации должен осуществляться ее контроль на допустимость вводимых значений по следующим правилам:

- вводимые данные не должны противоречить заявленному типу данных;
- данные имеющие формат номера телефона должны вноситься по соответствующему формату;
- вводимые данные не должны противоречить законам логики.

При выполнении операции по изменению и удалению данных должна быть обеспечена целостность базы данных.

Конечный пользователь не должен иметь доступ к исходным данным приложения. Для модификации программной части системы пользователю

					ПГУ1.090501.05.001 ПЗ	Лист
						26
Из	Лист	№ докум.	Подпись	Дата		

необходимо обратиться к разработчику данного программного приложения.

2.2.3 Требования к составу и параметрам технических средств

Разрабатываемое приложение должно работать в условиях локальной сети предприятия (или компании).

Программное приложение предназначено для работы на персональных компьютерах специалистов по тестированию.

Минимальная конфигурация технических средств должна включать:

- операционная система Windows 7 и выше;
- процессор Intel Core 1,9 ГГц и выше;
- свободное место на жестком диске 2 ГБ и выше;
- оперативную память 2 ГБ и выше;
- монитор с диагональю не менее 16 дюймов;
- клавиатура;
- мышь.

2.3 Требования к программной документации

Программное приложение контроля выполнения задач должно сопровождаться документацией, содержащей:

- техническое задание;
- описание логического проектирования БД;
- описание физического проектирования БД;
- описание программы;
- программу и методику испытаний;
- описание применения.

2.4 Стадии и этапы разработки

Этапы разработки программного приложения контроля выполнения задач представлены в таблице 2.

					ПГУ1.090501.05.001 ПЗ	Лист
						27
Из	Лист	№ докум.	Подпись	Дата		

Таблица 2 — Стадии и этапы разработки

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
1 Анализ предметной области	Анализ предметной области	10		Кузнецова П.С.
2 Проектирование программного приложения	-концептуальное проектирование БД -логическое проектирование БД -физическое проектирование БД	10		Кузнецова П.С.
3 Разработка базы данных	Разработка базы данных	20		Кузнецова П.С.
4 Отладка и тестирование	Отладка и тестирование	10		Кузнецова П.С.
5 Разработка экономической части	Оценка затрат на разработку и внедрение программного приложения контроля выполнения задач	5		Кузнецова П.С.
6 Разработка раздела БЖД	Оценка тяжести труда эксплуататора программного приложения	5		Кузнецова П.С.
7 Оформление графической части	Оформление графической части	10		Кузнецова П.С.
8 Оформление Выпускной работы	Оформление пояснительной записки	30		Кузнецова П.С.

2.6 Порядок контроля и приемки

Для проверки выполнения программой всех ее функций в соответствии с требованием технического задания необходимо разработать тестовые наборы

					ПГУ1.090501.05.001 ПЗ	Лист
						28
Из	Лист	№ докум.	Подпись	Дата		

данных. Необходимо протестировать действия программы на возможные реакции пользователя: запуск программы, заполнение полей ввода, выбор пунктов меню, нажатие кнопок, работа с базой данных. Приложение соответствует требованиям технического задания, если оно выполняет все заданные функции при различных входных данных из тестовых наборов данных.

Проверка и приемка программы должна осуществляться при ее работоспособности и в присутствии заказчика, выполнения всех возложенных на нее функциональных возможностей, а также при наличии полной документации на программу (пояснительная записка с приложениями: текстом программы и результатом текстовых примеров).

					ПГУ1.090501.05.001 ПЗ	Лист
						29
Из	Лист	№ докум.	Подпись	Дата		

3 Обоснование целесообразности использования выбранных средств разработки

На основании анализа требований к функциональным характеристикам программного приложения контроля выполнения задач были выбраны наиболее актуальные средства разработки.

3.1 Язык разработки JavaScript

JavaScript является динамическим скриптовым языком программирования высокого уровня, который используется для создания интерактивных веб-страниц и приложений, разработанный компанией Netscape в 1995 году и с тех пор стал одним из самых популярных языков программирования в мире.

JavaScript позволяет создавать динамические веб-страницы, которые могут реагировать на действия пользователя. Еще одной важной особенностью JavaScript является его интеграция с HTML и CSS. Он может использоваться для добавления интерактивности на веб-страницы, изменения стилей элементов и выполнения других задач. Кроме того, JavaScript имеет обширную библиотеку функций, которая позволяет выполнять различные задачи, такие как работа с файлами, обработка ошибок и многое другое. В целом, JavaScript является мощным и гибким языком программирования, который широко используется для создания веб-приложений и интерактивных веб-сайтов. Одной из главных особенностей JavaScript является его объектно-ориентированный подход. Объекты в JavaScript представляют собой экземпляры классов, которые могут содержать свойства и методы. Это позволяет создавать более структурированный и понятный код.[5] Немаловажной особенностью языка является частичная обработка веб-страниц на компьютере пользователя без лишних запросов к серверу, что сильно снижает на него нагрузку и экономит трафик и время. Одним из недостатков является то, что свободный скриптовый язык подвержен внедрению

					ПГУ1.090501.05.001 ПЗ	Лист
						30
Из	Лист	№ докум.	Подпись	Дата		

вредоносного кода. А также отсутствие поддержки удаленного доступа, что делает невозможным сейчас создание сетевых приложений и отсутствие возможности чтения и загрузки файлов.

3.1.1 Библиотека React

React — это библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. React нацелен на создание представлений приложений через инкапсулированные единицы, которые сохраняют состояние и генерируют элементы пользовательского интерфейса.

В данной библиотеке есть четыре преимущества – это универсальность, гибкость, отличный DevX, читаемость.

1. Универсальность.

Данная JavaScript-библиотека использует популярный язык программирования JavaScript, используемый повсюду. Работает в большом количестве окружений и устройств.

2. Гибкость.

Библиотека является гибкой, потому что позволяет изменять инструменты разработки, создавать полностью приложение на React, или для определенных целей, выбирать какие библиотеки использовать.

3. Отличный DevX.

DevX или опыт разработчика, то насколько комфортно, удобно и просто разработчику взаимодействовать с технологиями.

Всё это делает React - удобно для разработчика. Важно, что у библиотеки много пользователей, следовательно, есть возможность перенять опыта.

4. Читаемость.

Библиотека React показывает, в какой части кода проблема, так как приложение разбивается на компоненты.[6]

					ПГУ1.090501.05.001 ПЗ	Лист
						31
Из	Лист	№ докум.	Подпись	Дата		

3.2 Язык разработки HTML5

HTML5 (HyperText Markup Language, version 5) — язык для структурирования и представления содержимого всемирной паутины, является пятой версией HTML. Новая открытая платформа, используемая для создания веб-приложений с графикой, видео, аудио и анимацией.

HTML5 обладает большим количеством новых синтаксических особенностей, а именно, добавлены новые элементы и атрибуты, с помощью которых отражается типичное использование разметки на современных веб-сайтах.[7] Это например добавление тегов аудио, видео, и дополнительная возможность использовать математические формулы. Все это разработано для упрощения создания и управления графическими и мультимедийными объектами. Теперь нет необходимости использовать сторонние API и плагины. Также было создано ряд других элементов, предназначенных для обогащения семантического наполнения страницы. Помимо этого были введены и новые атрибуты, а ряд старых был стандартизирован, изменен или вовсе удален. Основными частями спецификации HTML5 стали API и DOM. HTML5-совместимые браузеры, используют алгоритмы разбора ошибок разметки в документах, чтобы построить правильную объектную модель (DOM). Жесткое определение требований к юзер-агентам необходимо для достижения совместимости между браузерами от разных производителей.

3.3 Язык разработки CSS

CSS (Cascading Style Sheets)— это язык, с помощью которого описывается внешний вид документа, определяется стиль и расположение элементов на веб-странице. Данный язык работает в связке с HTML, который, в свою очередь, отвечает за размещение непосредственно контента на странице. Вместо использования тегов для форматирования текста, CSS позволяет задавать стили для элементов HTML, что упрощает и улучшает структуру исходного кода.

					ПГУ1.090501.05.001 ПЗ	Лист
						32
Из	Лист	№ докум.	Подпись	Дата		

CSS переводится как каскадные таблицы стилей, он получил свое название за принцип работы каскадом, то есть прописанное ниже правило является приоритетным.

Использование CSS значительно упрощает и улучшает разработку веб-страниц, так как позволяет создавать красивый и удобный дизайн.[8] Но главное позволяет не усложнять исходный код HTML-документа.

Данный язык имеет собственный синтаксис, который включает в себя правила, с помощью которых задают внешний вид элементов. Каждое CSS-правило состоит из нескольких элементов: селектора, CSS-свойств и их значений. Блоком объявления стилей называют определенную связку «свойство: значение».

3.4 Сборщик модулей Webpack

Webpack — один из первых сборщиков фронтенда. Является сборщиком модулей JavaScript с открытым исходным кодом.

Изначально Webpack создан для JavaScript, но сейчас может преобразовывать и внешние ресурсы (HTML, CSS) при условии включения соответствующих загрузчиков.

Особенностью этого сборщика является возможность работать как из командной строки, так и при помощи файла конфигурации. С помощью этого файла определяются правила, плагины и др.

Принцип работы Webpack следующий: в сам проект устанавливается сборщик, как npm-библиотека, с описанием его конфигурации, обычно, в dev-зависимости, а для работы удобно использовать cli-утилиту. Она содержит все правила сборки (какие файлы, обработчики, куда помещать результат). После этого на выходе Webpack формирует файл / набор файлов, которые готовы для загрузки в браузер. Дальнейшее же использование этих файлов переходит к программисту.

Можно отметить, что Webpack обладает встроенным сервером, который используется в качестве HTTP-сервера чтобы обслуживать файлы в процессе разработки.

3.5 CASE-средство OpenModelSphere 3.2

Для разработки базы данных используются специальные CASE-средства для разработки концептуальных, логических и реляционных моделей.

Популярным CASE-средством является OpenModelSphere 3.2 - бесплатные инструменты моделирования, доступные в виде программного обеспечения с открытым исходным кодом, выпущенного под GPL (общедоступная лицензия GNU). [9]

Преимущества OpenModelSphere 3.2:

- обеспечивание поддержки прямого и обратного проектирования между UML и реляционными схемами;
- работает с Oracle, Informix, Microsoft SQL Server, Sybase, DB2, PostgreSQL;
- построение графической модели БД в нотациях: IDEF1X, IE, UML;
- создание таблицы связей БД, в т.ч. «многие ко многим»;
- бесплатное ПО.

Недостатками OpenModelSphere 3.2 являются:

- ограничены возможности по обеспечению многопользовательской среды;
- невозможность построения моделей и их синхронизации с БД на сервере.

3.6 СУБД PostgreSQL

Для функционирования базы данных необходимо выбрать систему управления базами данных.

					ПГУ1.090501.05.001 ПЗ	Лист
						34
Из	Лист	№ докум.	Подпись	Дата		

СУБД – это набор программ, управляющие структурой БД и контролирующие доступ к данным, хранящимся в БД. Система управления служит посредником между пользователем и БД. Сама структура БД хранится в виде набора файлов, и единственный способ получить доступ к данным в этих файлах – через СУБД.

СУБД PostgreSQL – это свободная объектно-реляционная система управления базами данных.

Выделяется ряд особенностей PostgreSQL:

- поддерживается БД неограниченного размера;
- обладает легкой расширяемостью;
- расширяемая система встроенных языков программирования;
- функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД;
- триггеры определяются как функции, иницируемые DML-операциями;
- механизм правил – есть механизм создания пользовательских обработчиков;
- наследование реализовано на уровне таблиц, которые могут наследовать характеристики и наборы полей от других таблиц.[10]

При разработке сайтов и веб-приложений PostgreSQL опережает по использованию MySQL и MariaDB. Большое количество фреймворков, такие как Ruby on Rails, Symfony, Yii и др. поддерживают использование PostgreSQL в разработке.

3.7 Среда разработки Visual Studio Code

Visual Studio Code – расширенный редактор кода, который используется для большого количества языков программирования. VS Code прост в освоении, подходит для начинающих разработчиков и мало весит. Обладает горячими клавишами, для использования функций без обращения к меню, а

также в него встроен отладчик для, используемого при разработке приложения, языка JavaScript.

Выбранный редактор кода является бесплатным и подходит для актуальных операционных систем.

Visual Studio Code обладает рядом преимуществ:

- легкость освоения;
- маленький вес, за счет этого быстро работает и менее требователен в ресурсам компьютера;
- кроссплатформенность;
- поддержка большого количества языков программирования;
- гибкость, за счет множества расширений и удобных функций.

					ПГУ1.090501.05.001 ПЗ	Лист
						36
Из	Лист	№ докум.	Подпись	Дата		

Описание атрибутов сущностей логической модели приведено в таблицах 3 – 7.

Таблица 3 – Описание атрибутов сущности «leaders»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
IDleader	ID руководителя	РК	
fullname	ФИО руководителя		
gender	Пол руководителя		
age	Возраст руководителя		
numberphone	Номер телефона руководителя		
jobtitle	Должность руководителя		

Таблица 4 – Описание атрибутов сущности «testers»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
IDtesters	ID тестирующего	РК	
fullname	ФИО тестирующего		
gender	Пол тестирующего		
age	Возраст тестирующего		
numberphone	Номер телефона тестирующего		
jobtitle	Должность тестирующего		

Таблица 5 - Описание атрибутов сущности «loginpass»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
IDuserpass	ID пользователя	РК	
fullname	ФИО пользователя		

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
gender	Пол пользователя		
age	Возраст пользователя		
numberphone	Номер телефона пользователя		
jobtitle	Должность пользователя		

Таблица 6 – Описание атрибутов сущности «taskbord»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
IDtaskbord	ID доски задач	РК	
Task IDtask	ID задачи		FK

Таблица 7 – Описание атрибутов сущности «task»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
IDtask	ID задачи	РК	
nametask	Название задачи		
descriptions	Описание задачи		
notes	Комментарий к задаче		
leaders IDleader	ID руководителя		FK1
loginpass IDuserpass	ID пользователя		FK2
testers IDtesters	ID тестирующего		FK3

Процесс нормализации – это последовательный процесс приведения базы данных к эталонному виду, то есть переход от одной нормальной формы к следующей.

Отношение находится в 1 НФ, если все строки таблицы различные и все атрибуты являются атомарными.

Отношение находится во 2 НФ, если оно находится в 1 НФ и каждый не ключевой атрибут таблицы функционально полно зависит от первичного ключа.

Отношение находится в 3 НФ, если оно находится во 2 НФ и между не ключевыми атрибутами не существует транзитивных зависимостей.

Все атрибуты пяти сущностей атомарны и находятся в 1 НФ. Для подтверждения данного фактора необходимо рассмотреть каждую сущность:

- сущность «leader» имеет ключевой атрибут «IDleader». Атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности являются простыми и неделимыми;

- сущность «testers» имеет ключевой атрибут «IDtesters». Атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности являются простыми и неделимыми;

- сущность «loginpass» имеет ключевой атрибут «IDuserpass». Атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности являются простыми и неделимыми;

- сущность «taskbord» имеет ключевой атрибут «IDtaskbord». Атрибут «Task IDtask» данной сущности является простым и неделимым;

- сущность «task» имеет ключевой атрибут «IDtask». Атрибут «nametask», «descriptions», «notes», «leaders IDleader», «loginpass IDuserpass», «testers IDtesters» данной сущности является простым и неделимым.

Все отношения логической модели данных находятся во 2 НФ, так как их не ключевые атрибуты функционально полно зависят от первичных ключей.

Сущность «leader» находится во 2 НФ, так как имеет единственный ключевой атрибут «IDleader». Не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности функционально полно

зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «testers» находится во 2 НФ, так как имеет единственный ключевой атрибут «IDtesters». Не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «loginpass» находится во 2 НФ, так как имеет единственный ключевой атрибут «IDuserpass». Не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «taskbord» находится во 2 НФ, так как имеет единственный ключевой атрибут «IDtaskbord». Не ключевой атрибут «Task IDtask» данной сущности функционально полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «task» находится во 2 НФ, так как имеет единственный ключевой атрибут «IDtask». Не ключевые атрибуты «nametask», «descriptions», «notes», «leaders IDleader», «loginpass IDuserpass», «testers IDtesters» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

В сущности «leader» не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «testers» не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данного отношения не зависят друг от друга.

					ПГУ1.090501.05.001 ПЗ	Лист
						41
Из	Лист	№ докум.	Подпись	Дата		

Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «loginpass» не ключевые атрибуты «fullname», «gender», «age», «numberphone», «jobtitle» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «taskbord» не ключевой атрибут «Task IDtask» данного отношения не зависит от других атрибутов. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «task» не ключевые атрибуты «nametask», «descriptions», «notes», «leaders IDleader», «loginpass IDuserpass», «testers IDtesters» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

Таким образом, была разработана логическая модель данных.

4.2 Физическое проектирование

На основании разработанной логической модели данных была построена физическая модель данных. Физическая модель данных разработана для СУБД PostgreSQL и представлена на рисунке 12.

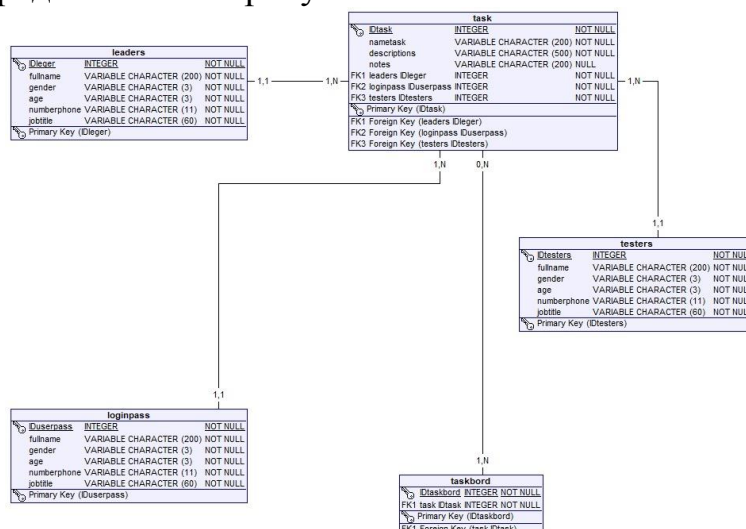


Рисунок 12 – Физическая модель данных

Физическая модель данных содержит пять таблиц. Описание таблиц физической модели приведено в таблицах 8 - 12.

Таблица 8 – Описание полей таблицы «leaders»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
IDleader	IDleader	Не допустимо	PK	INTEGER
fullname	fullname	Не допустимо		VARCHAR(200)
gender	gender	Не допустимо		VARCHAR(3)
age	age	Не допустимо		VARCHAR(3)
numberphone	numberphone	Не допустимо		VARCHAR(11)
jobtitle	jobtitle	Не допустимо		VARCHAR(60)

Таблица 9 – Описание полей таблицы «testers»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
ID testers	ID testers	Не допустимо	PK	INTEGER
fullname	fullname	Не допустимо		VARCHAR(200)
gender	gender	Не допустимо		VARCHAR(3)
age	age	Не допустимо		VARCHAR(3)
numberphone	numberphone	Не допустимо		VARCHAR(11)
jobtitle	jobtitle	Не допустимо		VARCHAR(60)

Таблица 10 – Описание полей таблицы «loginpass»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
IDuserpass	IDuserpass	Не допустимо	PK	INTEGER

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
fullname	fullname	Не допустимо		VARCHAR(200)
gender	gender	Не допустимо		VARCHAR(3)
age	age	Не допустимо		VARCHAR(3)
numberphone	numberphone	Не допустимо		VARCHAR(11)
jobtitle	jobtitle	Не допустимо		VARCHAR(60)

Таблица 11 – Описание полей таблицы «taskbord»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
IDtaskbord	IDtaskbord	Не допустимо	PK	INTEGER
Task IDtask	Task IDtask	Не допустимо	FK	INTEGER

Таблица 12 – Описание полей таблицы «task»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
IDtask	IDtask	Не допустимо	PK	INTEGER
nametask	nametask	Не допустимо		VARCHAR(200)
descriptions	descriptions	Не допустимо		VARCHAR(500)
notes	notes	Допустимо		VARCHAR(200)
leaders IDleader	leaders IDleader	Не допустимо	FK1	INTEGER
loginpass IDuserpass	loginpass IDuserpass	Не допустимо	FK2	INTEGER
testers IDtesters	testers IDtesters	Не допустимо	FK3	INTEGER

Таким образом, была разработана физическая модель данных для СУБД PostgreSQL.

4.3 Описание функций программы

К основным функциям программного приложения контроля выполнения задач можно отнести: добавление нового пользователя, которая включает в себя подфункции добавление нового тестировщика и добавление нового руководителя; добавление задачи, а именно ввод данных и проверка на выполнение всех основных этапов тестирования; рендер карточек, что подразумевает передвижение карточки с задачей по стартовому окну приложения.

Функцию добавления нового пользователя рассмотрим на примере добавления нового тестировщика, так как поля для заполнения данных у тестировщика и руководителя одинаковые.

Для добавления нового тестировщика необходимо в приложении перейти на вкладку тестировщики, ввести необходимые данные, после чего справа появится иконка с новым тестировщиком.

```
const hendaleCreateTesters = () => {
  const id = new Date().getTime()
  setTesters([ ...testers, {
    id, name, floor, age, numberTester, post,
  }])
  localStorage.setItem('testers', JSON.stringify([
...testers, {
  id, name, floor, age, numberTester, post,
}]))
  setName('')
  setFloor('')
  setAge(0)
  setNumberTester(0)
  setPost('')
}
```

Функция добавления задачи подразумевает генерацию карточки для тестового случая по заранее созданному шаблону. В нее заносятся все, внесенные пользователем данные, затем генерируется текущая дата создания.

Для добавления новой задачи необходимо в приложении перейти на вкладку «Создать задачу», после чего заполнить все поля и создать карточку тестового случая. Для просмотра карточки необходимо перейти на основную доску задач, где и будет показана новая карточка.

```
const hendaleCreateTask = () => {
  const date = new Date().getTime()
  const type = 1
  const testCheck = 0
  const taskComment = ''
  setCreatedTasks([ ...createdTasks, {
    time:    date,    taskName,    taskDescription,
    taskTester, type, testCheck, taskComment,
  }])
  localStorage.setItem('createdTasks',
JSON.stringify([ ...createdTasks, {
    time:    date,    taskName,    taskDescription,
    taskTester, type, testCheck, taskComment,
  }]))
  setTaskName('')

  setTaskDescription('')
}
```

Функция рендера карточки (изменения) предполагает проверку, внесенных пользователем данных и отметок в чекбоксах при наличии карточки в одном из трех полей на доске задач.

Для изменения основных данных необходимо открыть нужную карточку, выбрать кнопку редактировать и после этого будет доступно внесение изменений с содержание карточки.

```
const renderCreatedTasks = () => {
  if (createdTasks.length !== 0) {
    return (
```

```

        createdTasks.map(el => (
        <Card
            key={ el.time }
            time={ el.time }
            taskName={ el.taskName }
            taskDescription={ el.taskDescription }
            taskTester={ el.taskTester }
            type={ el.type }
            testCheck={ el.testCheck }
            taskComment={ el.taskComment }
            setTaskInfo={ setTaskInfo }
            createdTasks={ createdTasks }
            setCreatedTasks={ setCreatedTasks }
            workTasks={ workTasks }
            setWorkTasks={ setWorkTasks }
            completedTasks={ completedTasks }
            setCompletedTasks={ setCompletedTasks }
        />
        ))
    return null
}

```

4.4 Методика испытаний программы

Функциональное тестирование предполагает проверку всех реализованных функций на предмет соответствия их техническому заданию.

После входа в приложение открывается главная доска для тестовых случаев (рисунок В.1). В правом верхнем углу есть иконка пользователя, при нажатии на нее открывается страница с информацией о пользователе (рисунок В.2). Для возвращения на основную доску необходимо в верхнем левом углу нажать на «Task Bord» (рисунок В.3) после этого откроется главная доска (см. рисунок В.1). Далее в верхнем меню выбираем кнопку «Тестировщики» и нажимаем на нее (рисунок В.4), и происходит переход на страницу для создания нового пользователя, где действующий пользователь уже существует (рисунок В.5). Для добавления нового тестировщика необходимо заполнить

					ПГУ1.090501.05.001 ПЗ	Лист
						47
Из	Лист	№ докум.	Подпись	Дата		

все поля: ФИО, пол, возраст, номер телефона, должность (рисунок В.6). Затем нажать кнопку «Добавить тестировщика», после этого форма очистится, а справа появится новый пользователь (рисунок В.7). Далее выбираем пункт меню «Создать задачу» и открывается окно создания тестового случая (рисунок В.8). Для заполнения карточки задачи необходимо в поля «Название», «Описание» внести информацию, и выбрать себя из списка тестировщиков для закрепления за данной задачей и нажать кнопку «Создать задачу» (рисунок В.9). Созданную карточку можно увидеть на главной доске в разделе «Созданные» (рисунок В.10). Для просмотра полной карточки тестового случая необходимо нажать на кнопку «Подробнее», после чего откроется подробная информация (рисунок В.11). Важно заметить, что в режиме просмотра редактировать данные не получится. При нажатии кнопки «Редактировать» становятся активными все поля карточки, после их заполнения, необходимо нажать кнопку «Сохранить» (рисунок В.12). Для того чтобы перевести тестовый случай из одного состояния в другой необходимо нажать на кнопку «Перевести в работу» после чего на главной странице приложения карточка отобразится в поле «В работе» (рисунок В.13). Для перевода карточки в режим «Готовые» необходимо снова нажать «Подробнее», а затем «Задача готова» (рисунок В.14). После создания карточки есть возможность ее удаления, для этого необходимо нажать кнопку «Удалить» (рисунок В.15). Для удаления пользователя необходимо в верхнем меню выбрать раздел «Тестировщики», открыть вкладку и нажать на крестик у иконки желаемого пользователя (рисунок В.16).

					ПГУ1.090501.05.001 ПЗ	Лист
						48
Из	Лист	№ докум.	Подпись	Дата		

5 Оценка затрат на разработку и внедрение программного приложения контроля выполнения задач

5.1 Организация производства разработки программного приложения

Разработка программного приложения осуществлена на базе собственной площади общей площадью 30 кв.м. Содержание помещения в месяц составляет 15000 рублей.

Для разработки программного продукта приобретено оборудование (вычислительная техника), лицензионное программное обеспечение, дополнительные материалы.

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблицах 13 и 14.

Таблица 13 – Потребность в оборудовании

Оборудование	Источник	Количество	Ориентировочная цена за единицу, рублей	Потребляемая мощность, Ватт
Ноутбук Huawei MateBook D 14 NbD-WDI9, 14"	ООО «Ситилинк»	3	39990	136

Таблица 14 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Microsoft Windows 10	Входит в комплект «Ноутбук Huawei MateBook D 14 NbD-WDI9, 14"»	-	бессрочная
VS Code	Интернет-магазин VS Code: https://code.visualstudio.com/download	-	Бессрочная
Антивирус Kaspersky Plus 2023	Интернет-магазин Kaspersky URL: https://www.kaspersky.ru/	1990	365
СУБД PostgreSQL	Интернет-магазин Postgresql URL: https://www.postgresql.org/download/	-	бессрочная
OpenModelSphere3.2	Интернет-магазин Open Model Sphere URL: http://www.modelsphere.com/org/download.html	-	бессрочная

Перечень дополнительных материалов, источники их приобретения, объем, и ориентировочные цены приведены в таблице 15.

Таблица 15 – Потребность в дополнительных материалах

Материал	Источник	Количество, штук	Ориентировочная цена за единицу, рублей
Стол компьютерный ДИЗАЙН ФАБРИКА Ascetic 716 Венге	ООО «Леруа Мерлен восток»	3	2000

					ПГУ1.090501.05.001 ПЗ	Лист
						50
Из	Лист	№ докум.	Подпись	Дата		

Материал	Источник	Количество, штук	Ориентировочная цена за единицу, рублей
71.59x73.2x50 см ЛДСП цвет венге			
Рабочая лампа на струбцине KD-312, цвет чёрный	ООО «Леруа Мерлен восток»	3	1150
Стул Leset Норман металл белый ткань кофейный	ООО «Леруа Мерлен восток»	3	5200
Канцтовары	Гипермаркет «Канцлер»	3	1000

Разработка программного приложения контроля выполнения задач осуществлялась командой разработчиков, представленных в таблице 16.

Таблица 16 – Состав коллектива разработчиков

Специалист	Количество, человек	Описание занятости
Руководитель проекта	1	<ul style="list-style-type: none"> - управление проектом - набор команды исполнителей - управление ресурсами проекта - контроль соблюдения требований заказчика - приемка и внедрение проекта - анализ предметной области

Специалист	Количество, человек	Описание занятости
Программист	1	<ul style="list-style-type: none"> - создание базы данных - написание, отладка и первичное тестирование программного кода - разработка и утверждение программной документации - приемка и внедрение программного кода
Тестировщик	1	<ul style="list-style-type: none"> - тестирование программного приложения

Данные по основной заработной плате разработчиков программного приложения и их количественному составу приведены в таблице 17.

Таблица 17 – Сведения о заработной плате

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	50000
Программист	1	50000
Тестировщик	1	25000

5.2 Расчет себестоимости программного обеспечения для программного приложения контроля выполнения задач

Себестоимость разработки программного приложения включает следующие расходы:

- на приобретение оборудования;
- на приобретение лицензионного программного обеспечения;
- затраты на дополнительные материалы;
- затраты на заработную плату;

- затраты на обслуживание программы.

Затраты на оборудование в период разработки составят 119970 рублей и представлены в таблице 13.

Затраты на программное обеспечение в период разработки составят 1990 рублей и представлены в таблице 14.

Затраты на дополнительные материалы в период разработки составят 28050 рублей и представлены в таблице 15.

Затраты на заработную плату включают затраты на основную заработную плату разработчиков программного приложения и отчисления на страховые взносы.

С учетом выходных дней при 5-дневной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день. На разработку всего проекта необходимо 3 месяца.

Таким образом, затраты на заработную плату разработчиков составят 125000 рублей в месяц, 375000 рублей за весь период создания проекта.

Отчисления на страховые взносы – обязательные отчисления по нормам, установленным законодательством государственного социального страхования, в Пенсионный фонд РФ, фонд обязательного медицинского страхования, фонд социального страхования РФ от затрат на оплату труда работников, включаемых в себестоимость продукции (программного обеспечения), по элементу «Затраты на оплату труда» (кроме тех видов оплаты, на которые страховые взносы не начисляются).

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации» от 14.07.2022 № 239-ФЗ.[11]

С января 2023 года введена единая базовая ставка по страховым взносам. Действующие тарифы на страховые взносы работодателей установлены статьей 425 НК РФ и составляют 30%.

Так как компания осуществляет деятельность в области IT, то она может претендовать на льготную ставку (подп. 3 п. 1 ст. 427 НК РФ) равную 7,6%.

					ПГУ1.090501.05.001 ПЗ	Лист
						53
Из	Лист	№ докум.	Подпись	Дата		

Таким образом, с выплат в пределах МРОТ (минимального размера оплаты труда) уплачиваю взносы – 30%, с суммы свыше МРОТ уплачиваются взносы – 7,6%. МРОТ на 2023 год составляет 16242 рубля.

Данные по суммам отчислений на страховые взносы приведены в таблице 18.

Таблица 18 – Отчисления на страховые взносы

Сотрудник	Заработная плата, рублей	Взнос по единой ставке, рублей	Сумма, подлежащая пониженной ставке по взносу, рублей	Взнос по пониженной ставке, рублей	Общая сумма взноса с сотрудника за месяц, рублей
Руководитель проекта	50000	4872,6	33758	2565,6	7438,2
Программист	50000	4872,6	33758	2565,6	7438,2
Тестировщик	25000	4872,6	8758	665,6	5538,2

Затраты на заработную плату и отчисления на страховые взносы представлены в таблице 19.

Таблица 19 – Затраты на заработную плату

Сотрудник	Заработная плата, рублей	Общая сумма отчислений, рублей	Всего за 1 месяц, рублей	Всего за 3 месяца, рублей
Руководитель проекта	50000	7438,2	57438,2	172314,6

Сотрудник	Заработная плата, рублей	Общая сумма отчислений, рублей	Всего за 1 месяц, рублей	Всего за 3 месяца, рублей
Программист	50000	7438,2	57438,2	172314,6
Тестировщик	25000	5538,2	30538,2	91614,6
Итого:	125000	20416,6	145414,6	436243,8

Затраты на заработную плату команды проекта по созданию программного приложения составят 436243,8 рублей из которых общая сумма отчислений на страховые взносы разработчиков составит 20416,6 рублей, а заработная плата составит 415827,2 рублей.

Затраты на обслуживание включают следующие статьи:

- амортизация оборудования, составляющая 20% от стоимости оборудования;
- обслуживание оборудования, составляющее 15% от амортизационных отчислений;
- затраты на электроэнергию, включающие затраты на электроэнергию персональных компьютеров, оргтехники и бытовых приборов;
- расходы на содержание помещения;
- накладные расходы.

Амортизация оборудования составляет 20% от стоимости оборудования. Было приобретено 3 ноутбука по 39990 рублей, Таким образом затраты на амортизацию оборудования в месяц составят:

$$\frac{(3 * 39990) * 0,2}{12} = 1999,5 \text{ руб}$$

Обслуживание оборудования составляет 15% от амортизационных отчислений.

Таким образом затраты на обслуживание оборудования составят:

$$1999,5 * 0,15 = 299,93 \text{ руб.}$$

Предположительно в день по 8 часов будут работать 3 ноутбука:

$$136 * 8 * 63 * 3 = 247 \text{ кВт.}$$

Таким образом, за 63 дня разработки программного приложения будет использовано 247 кВт электроэнергии на питание ноутбуков, а также 0,5 кВт на прочие приборы.

$$247 + 0,5 = 247,5 \text{ кВт.}$$

Накладные расходы составляют 75% то заработной платы. Таким образом, затраты на накладные расходы составляют:

$$415827,2 * 0,75 = 311870,4 \text{ руб.}$$

Результат расчетов по статьям затрат на обслуживание приведены в таблице 20.

					ПГУ1.090501.05.001 ПЗ	Лист
						56
Из	Лист	№ докум.	Подпись	Дата		

Таблица 20 – Затраты на обслуживание

Статья затрат	Показатель отчислений	Сумма отчислений в месяц, рублей	Итоговая сумма отчислений за период разработки, рублей
Амортизация оборудования	20%	1999,5	5998,5
Обслуживание оборудования	15%	299,9	899,8
Затраты на электроэнергию	2,97 на 1 кВт/ч	735,1	2205,3
Накладные расходы	75%	103956,8	311870,4
Аренда офиса и расходы на его содержание	-	15000	45000
Итого, рублей		121991,3	365973,9

Себестоимость разработки определяется как сумма затрат, включающая все статьи затрат на реализацию проекта, представленное в таблице 21.

Таблица 21 – Общая сумма затрат

Статья затрат	Сумма отчислений, рублей
Приобретение оборудования	119970
Приобретение лицензионного программного обеспечения	1990
Затраты на заработную плату	436243,8

Статья затрат	Сумма отчислений, рублей
Затраты на дополнительные материалы	28050
Затраты на обслуживание	365973,9
Итого:	952227,7

Таким образом, сумма затрат на разработку проекта составит 952227,7 рублей и будет соответствовать себестоимости программного приложения контроля выполнения задач.

5.3 Оценка эффективности программного приложения контроля выполнения задач

Необходимость и целесообразность реализации инвестиционных проектов осуществляется путем оценки их экономической эффективности.

Экономическая эффективность инвестиционных проектов характеризуется системой показателей, отражающих соотношение результатов (притока, поступлений, дохода) и затрат (оттока, выплат) от операционной, инвестиционной и финансовой деятельности.

За счет внедрения и использования программного приложения контроля выполнения задач также ожидаются следующие неисчисляемые эффекты:

- увеличение скорости фиксации работ;
- минимизация ошибок, вызванных человеческим фактором;
- повышение эффективности поиска тестовых случаев;
- упрощение формирования карточек тестовых случаев;
- снижение нагрузки на сотрудников.

5.4 Анализ рисков при разработке и внедрении программного приложения контроля выполнения задач

Риски взаимодействия между заказчиком и исполнителем – это риски, связанные с отсутствием коммуникации между исполнителем и заказчиком или их представителями.

Недостаточное обсуждение задач или архитектуры может негативно сказаться на разрабатываемом программном приложении. Для уменьшения последствий от таких рисков на начальном этапе проекта разработано соглашение и обозначена в договоре частота проведения совещаний по проекту, координаторов проекта со стороны заказчика и исполнителя, а также состав рабочей группы. На раннем этапе проекта при разработке технического задания интенсивнее привлекался заказчик к обсуждению деталей проекта.

Риски планирования – это риски, которые могут быть связаны с отсутствием навыков планирования по проекту, как менеджером, так и исполнителями, если они готовят информацию о сроках выполнения работ. Также часто на этапе планирования сроков реализации проекта слишком мало времени закладывается на этап тестирования и исправления ошибок в исходном коде, предпроектного обследования. Для минимизации риска, проектный руководитель следил за четким выполнением сроков на каждом этапе проекта.

Функциональные риски – это риски, связанные с неверным выполнением функций, возложенных на программное приложение. Для минимизации функциональных рисков необходим тщательный анализ требований заказчика, разработка моделей, а также привлечение к работе только грамотных специалистов.

Риск смены сотрудников возникает, когда проект покидают ключевые сотрудники, которые владеют максимально полной информацией о проекте. Иногда уход такого сотрудника может полностью погубить проект.

					ПГУ1.090501.05.001 ПЗ	Лист
						59
Из	Лист	№ докум.	Подпись	Дата		

Для того чтобы минимизировать последствия от ухода ключевых сотрудников проекта, руководитель проекта поддерживает высокий уровень сотрудничества и обмена информацией в коллективе, планирует работу сотрудников таким образом, чтобы они могли заменить друг друга, используя системы корпоративного обмена знаниями и документами, а также выстраивает систему внутреннего обучения. Дополнительной мотивацией сотрудников не уходить является премирование, как материальное, так и нематериальное.

Риски хищения исходного кода возникают, когда разработчики, уходя из компании, забирают с собой разрабатываемый ими проект и, немного модифицировав исходный код, могут продать его или использовать в других проектах, например, у конкурентов.

Использование компанией не лицензированного программного обеспечения сопряжено существенными рисками привлечения к ответственности и, как следствие увеличение стоимости проекта. Для минимизации риска, перед началом разработки программного обеспечения было закуплено необходимое лицензионное программное обеспечение.

Коммерческий риск – риск, который в большей мере основан на появлении новых аналогов разрабатываемого приложения. Путем минимизации коммерческого риска является предварительное проведение тщательного маркетингового исследования рынка и анализ всех конкурентов. Необходимо обратить внимание на преимущества альтернативных приложений и их недостатки.

При предпроектном анализе должны быть раскрыты все риски, а также пути их минимизации. В конце должен быть составлен обобщающий отчет по всем рискам.

					ПГУ1.090501.05.001 ПЗ	Лист
						60
Из	Лист	№ докум.	Подпись	Дата		

6 Общие принципы обеспечения безопасности на рабочем месте оператора программного приложения контроля выполнения задач

6.1 Актуальность вопросов безопасности человека на рабочем месте

С развитием и большим внедрением IT технологий в жизнь человека, большое распространение получили ПЭВМ (персональные электронные вычислительные машины). Они стали применяться и для повышения производительности труда сотрудников в разных сферах. Именно поэтому актуальным вопросом является безопасность оператора ПЭВМ, а также сохранение его здоровья и работоспособности.

Использование ЭВМ связано с распознаванием печатного и рукописного текста, анализ изображений на экране, долгим нахождением в одной позе, и др. Это может пагубно сказаться на здоровье человека, если не придерживаться нормами использования ЭВМ.

Согласно ГОСТ 12.0.003-74[11] вредные производственные факторы подразделяются на 4 категории:

- физические, к ним относятся недостаточная освещенность рабочего места, повышенная яркость света, отклонения от нормы температуры воздуха, высокая влажность воздуха, повышенный уровень шума, и др.;
- психофизиологические, к ним относятся нервно-психические и физические перегрузки, перенапряжение ушей и глаз;
- биологические, к ним относятся микроорганизмы и макроорганизмы, которые могут вызвать различные заболевания;
- химические, к ним относятся превышения в рабочей зоне аммиака, фенола, озона, оксида углерода и др.

					ПГУ1.090501.05.001 ПЗ	Лист
						61
Из	Лист	№ докум.	Подпись	Дата		

6.2 Требования СанПиН при работе с компьютером

Требования СанПиН при работе с компьютером – являются набором правил, при выполнении которых, сохраняется безопасность сотрудников.

Гигиенические требования к ЭВМ и работе с ними устанавливаются СанПиН 2.2.2/2.4.1340-03 от 30.05.2003[12]. Статьи 107 и 109 Трудового кодекса Российской Федерации [13] предписывают выделять время на отдых сотрудникам в течение рабочего дня.

На основании пункта 13.1 СанПиН 2.2.2/2.4.1340-03, те сотрудники, которые находятся на рабочем месте, за компьютером, более половины рабочего дня должны проходить медицинский осмотр. Данный вид медицинского осмотра регламентируется Приказом Минздравсоцразвития России [14] №302н. от 12.04.2011.

Также СанПиН 2.2.2/2.4.1340-03 содержит требования к помещению, в котором будет находиться сотрудник. Чтобы соответствовать нормам, помещение должно отвечать следующим требованиям:

- хорошее освещение;
- не является подвальным помещением;
- не является цокольным помещением;
- хорошая вентиляция;
- заземленное оборудование;
- не менее 4,5 м² на одного сотрудника.

Важно отметить, что правильные ЭВМ не выделяют вредные вещества в воздух, не создают сильное электромагнитное поле, не являются шумными.

6.3 Техника безопасности при работе с электроприборами

ПЭВМ – является электроприбором, следовательно, при работе с ним необходимо знать необходимые правила техники безопасности:

- для работы с электрооборудованием следует использовать заземленные розетки;

					ПГУ1.090501.05.001 ПЗ	Лист
						62
Из	Лист	№ докум.	Подпись	Дата		

- при выключении приборов из сети, нельзя тянуть за шнур и иметь мокрые руки;
- при неисправности прибора нельзя чинить его до момента отключения от сети;
- до включения прибора в сеть необходимо проверить сохранность шнура (наличие заломов и повреждения изоляции);
- не оставлять включенные приборы без присмотра.

В состав ПЭВМ входят монитор, блок питания, процессор – все они создают электромагнитные поля. Такие поля создают еще распределительные щиты, электропроводка, линии электропередач.

6.4 Влияние ПЭВМ на физиологию человека

Работа за компьютером подразумевает длительное напряжение на зрительные органы и опорно-двигательный аппарат. Если не контролировать время пребывания за ЭВМ это может привести к нарушению зрения, искривлению позвоночника, а также нарушению кровообращения в организме.

Нагрузка на зрительный аппарат может вызвать нарушения в работе ЦНС (центральной нервной системы). При длительной работе за компьютером снижается фокусировка и острота зрения, увеличивается чувствительность глаз.

Длительное нахождение человека в одной позе приводит к быстрому развитию утомляемости. При работе за компьютером стоит обратить внимание на эти вещи:

- высота рабочей зоны и стула;
- наличие опорной спинки и подлокотников;
- угол наклона головы при работе;
- угол наклона экрана компьютера;
- пространство и подставка для ног.

Большое влияние на здоровье человека оказывает и уровень шума в помещении. Согласно СанПиН 2.2.3670-20 "Санитарно-эпидемиологические требования к условиям труда", эквивалентный уровень шума не должен превышать 50 дБА. При длительном превышении уровня шума в помещении ухудшаются условия труда человека, а также оказывается вред здоровью. Происходит снижение концентрации внимания и повышается усталость, следовательно, снижается работоспособность сотрудника. Чтобы снизить уровень шума в помещении устанавливается звукоизоляция.

6.5 Техника безопасности при работе с персональным компьютером

Согласно типовой инструкции по охране труда при работе на ПК (персональном компьютере) ТОИ Р-45-084-01 [15], которая была утверждена Приказом Минсвязи Российской Федерации №162 от 02.07.2001, деятельность на ЭВМ делится на 3 группы:

- группа А – чтение информации с монитора подразумевает два перерыва по 15 минут, без учета обеденного перерыва;
- группа Б – набор текста подразумевает перерывы по 10 минут раз в час;
- группа В – творческий труд подразумевает перерывы по 15 минут раз в час.

Данное время рекомендуется уделять физической нагрузке, такой как зарядка или гимнастика для глаз.

Подводя итог, для сохранения здоровья сотрудников и ограничения от негативного влияния ПЭВМ необходимо использовать только технику обладающую сертификатами качества, а также обеспечивать сотрудникам рациональный режим труда и перерывов. Важно заметить, что сертифицированная техника – это не полная гарантия безопасности условий труда, необходимо регулярно проводить проверку всей используемой техники.

					ПГУ1.090501.05.001 ПЗ	Лист
						64
Из	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения дипломного проекта была изучена предметная область, а именно работа специалистов по тестированию. На основе ее анализа было сформулировано техническое задание для разработки программного приложения контроля выполнения задач. Затем были выбраны средства разработки, благодаря которым было разработано программное приложение.

Программное приложение контроля выполнения задач реализовано в соответствии с техническим заданием.

В разделе экономики была произведена оценка затрат на разработку и внедрение приложения.

В разделе безопасность жизнедеятельности были проанализированы и описаны общие принципы обеспечения безопасности на рабочем месте оператора программного приложения контроля выполнения задач.

Задание на дипломное проектирование выполнено в полном объеме.

					ПГУ1.090501.05.001 ПЗ	Лист
						65
Из	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Карпова И.П. – «Базы данных. Курс лекций и материалы для практических заданий». – Учебное пособие. – 2013. – 240с.
2. Ключков А.К. – «КРІ и мотивация персонала. Полный сборник практических инструментов». – 2010. – 103с.
3. Куликов С.С. – «Тестирование программного обеспечения. Базовый курс». – 2017. – 312с.
4. Официальный сайт Atlassian [Электронный ресурс] <https://www.atlassian.com/blog/announcements/atlassian-stands-with-ukraine> (дата 02.03.2022).
5. Холланд Ева, Минник Крис – «JavaScript для «чайников». – 2017. – 320с.
6. Тиленс Томас Марк – «React в действии». – 2019. – 368с.
7. Фримен Элизабет, Фриман Эрик – «Изучаем программирование на HTML5». – 2013. – 592с.
8. Макфарланд Д. – «Новая большая книга CSS». – 2016. -720с.
9. Еременко А.В. – «Проектирование баз данных с помощью OpenModelSphere». – 2011. – 73с.
10. Лузанов П., Рогов Е., Лёвшин И. – «POSTGRES первое знакомство». – 2023. – 178с.
11. ГОСТ 12.0.0003-74 ССБТ. Библиографическая запись. Опасные и вредные производственные факторы. Классификация (с Изменением № 1). Изд-во стандартов, 2002. – 5 с.
12. СанПиН 2.2.2/2.4.1340-03. Библиографическая запись. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. Изд-во стандартов, 2003. – 41 с.
13. Трудовой кодекс Российской Федерации № 197-ФЗ 2001. – 259 с.

					ПГУ1.090501.05.001 ПЗ	Лист
						66
Из	Лист	№ докум.	Подпись	Дата		

14. Приказ Минздравсоцразртия России об утверждении перечней вредных и (или) опасных производственных факторов и работ № 302. -2011.

15. Типовая инструкция по охране труда при работе на персональном компьютере. – 2001. – 3 с.

					ПГУ1.090501.05.001 ПЗ	Лист
						67
Из	Лист	№ докум.	Подпись	Дата		

Приложение А

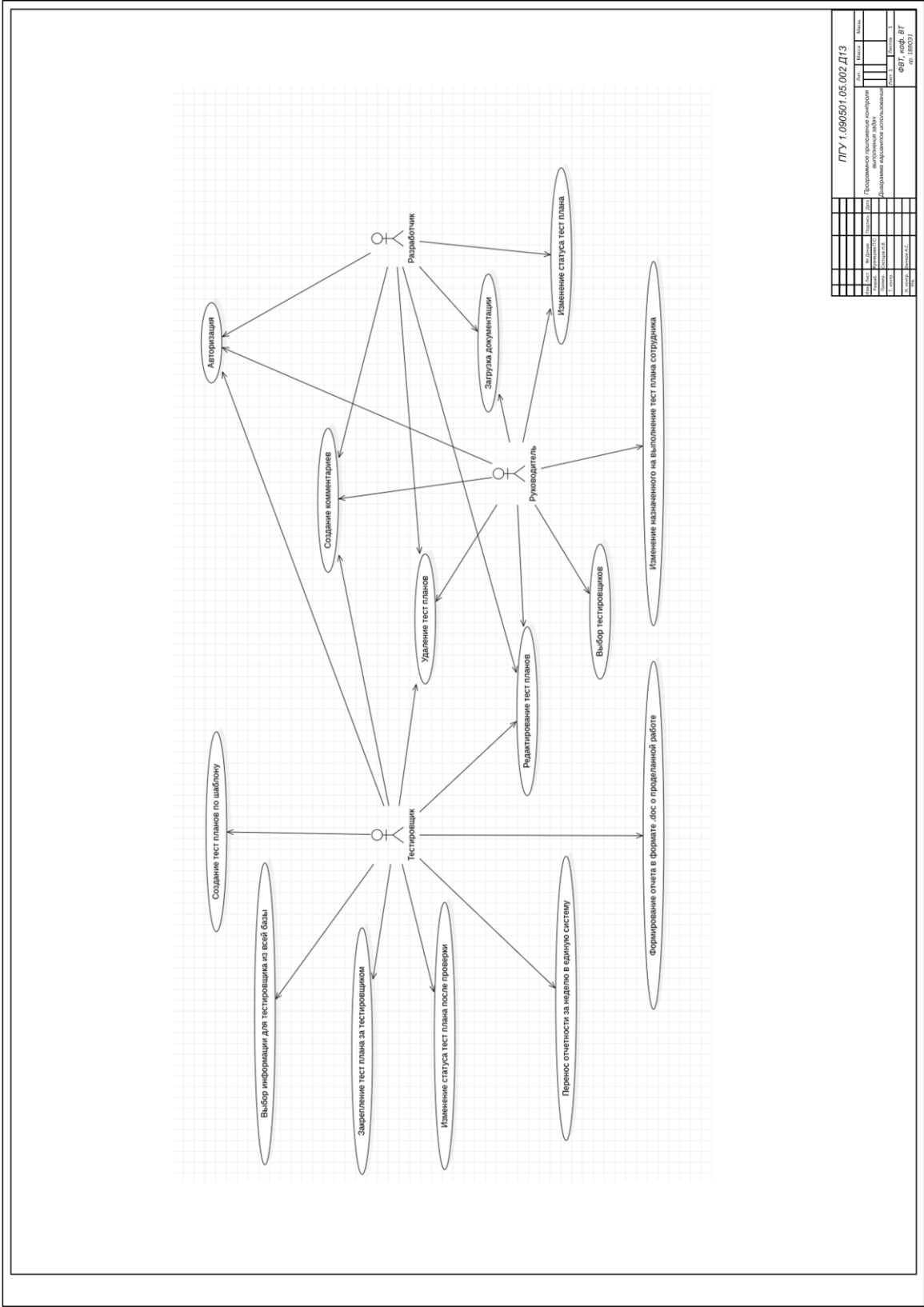
(обязательное)

Программное приложение контроля выполнения задач

UML-диаграммы

					ПГУ1.090501.05.001 ПЗ	Лист
						68
Из	Лист	№ докум.	Подпись	Дата		

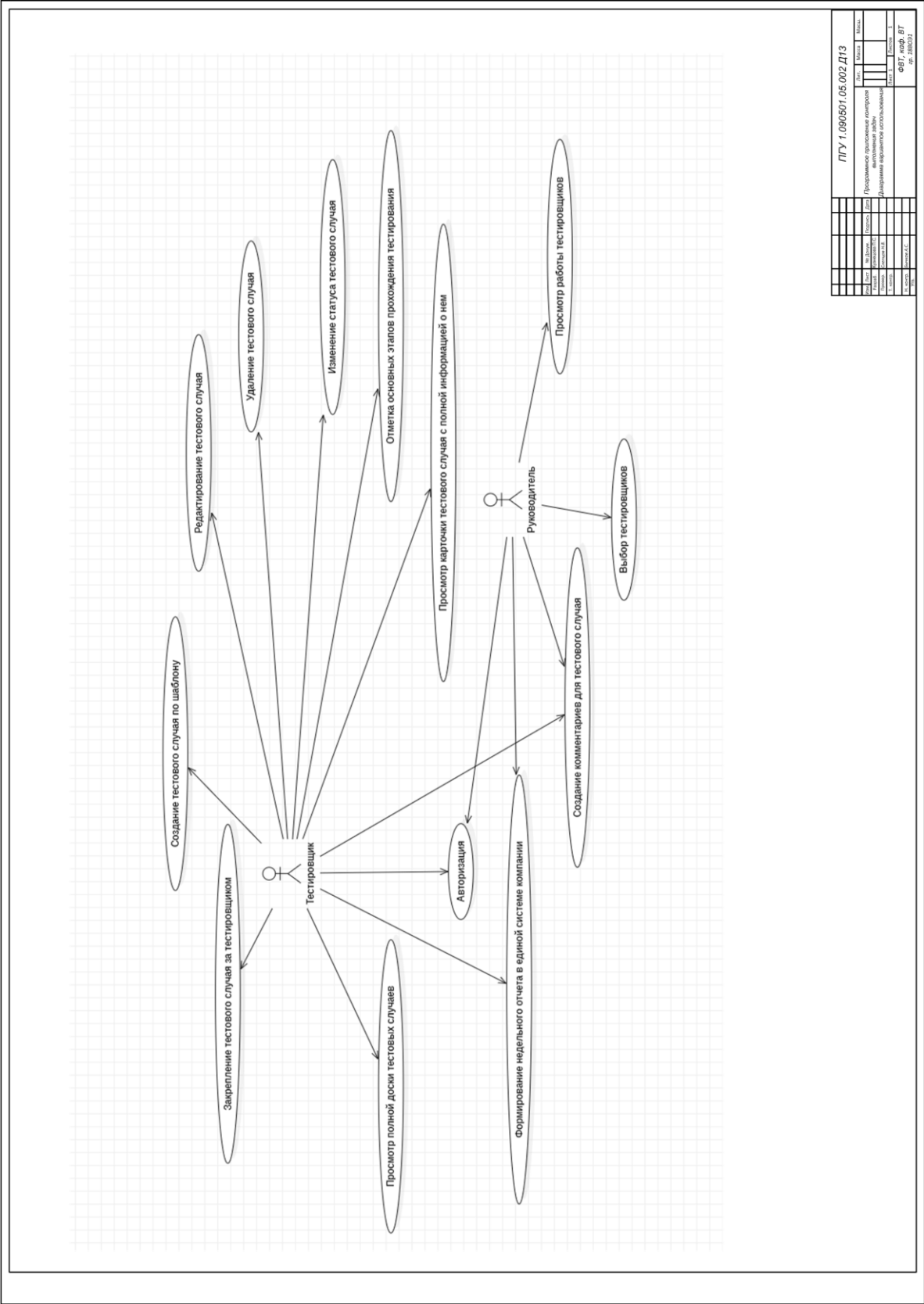
Приложение А.1 – Диаграмма вариантов использования «Как есть»



ПГУ 1.090501.05.002 Д13									
Имя	Фамилия	Инициалы	Подпись	Дата	Время	Место	Страна	Город	Улица
Подпись и печать ответственного за разработку диаграммы вариантов использования									
ФЭТ, сер. ВТ									
от 18.05.2013									

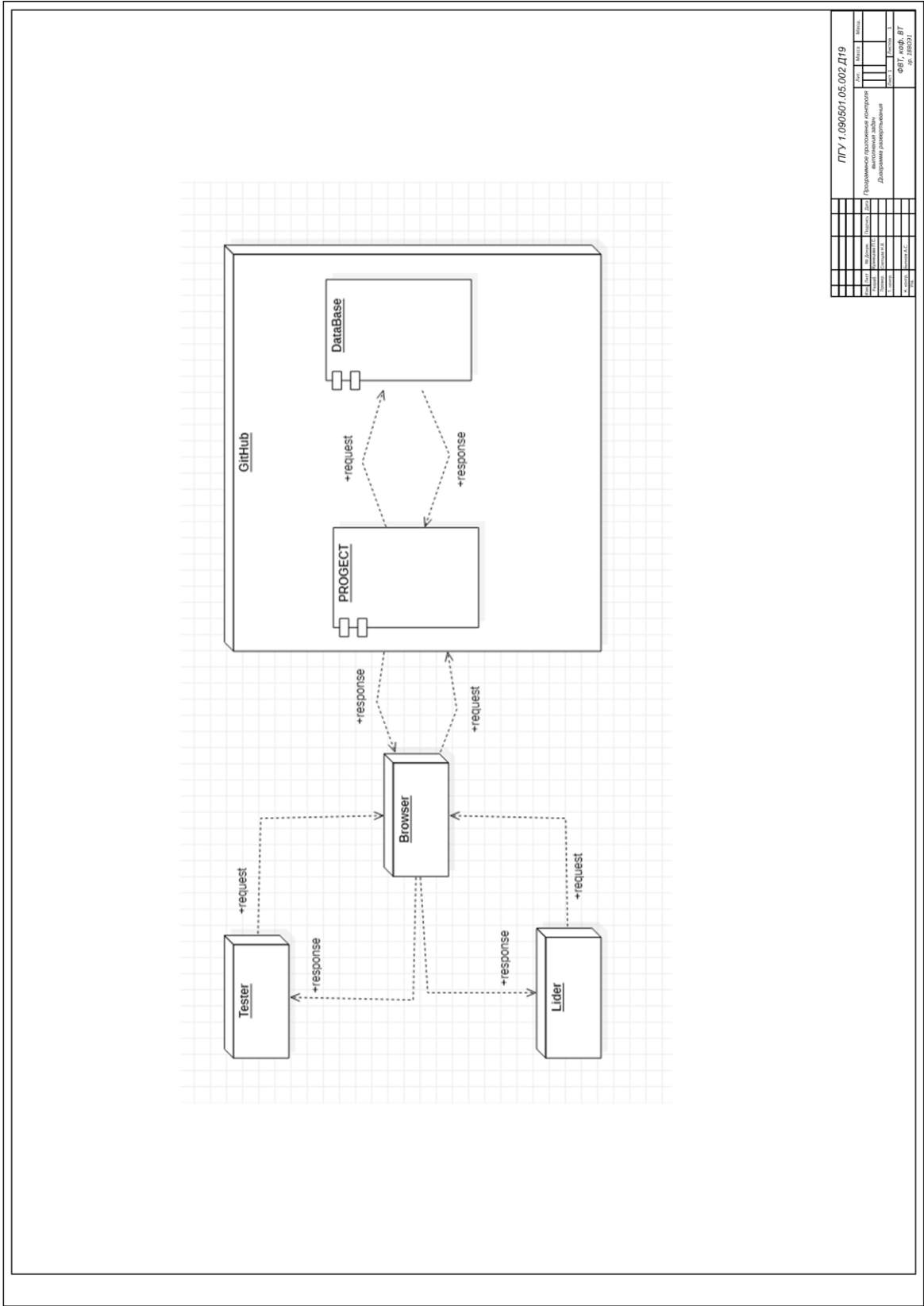
Из	Лист	№ докум.	Подпись	Дата

Приложение А.2 – Диаграмма вариантов использования «Как должно быть»



ПГУ 1.090501.05.002 Д/З									

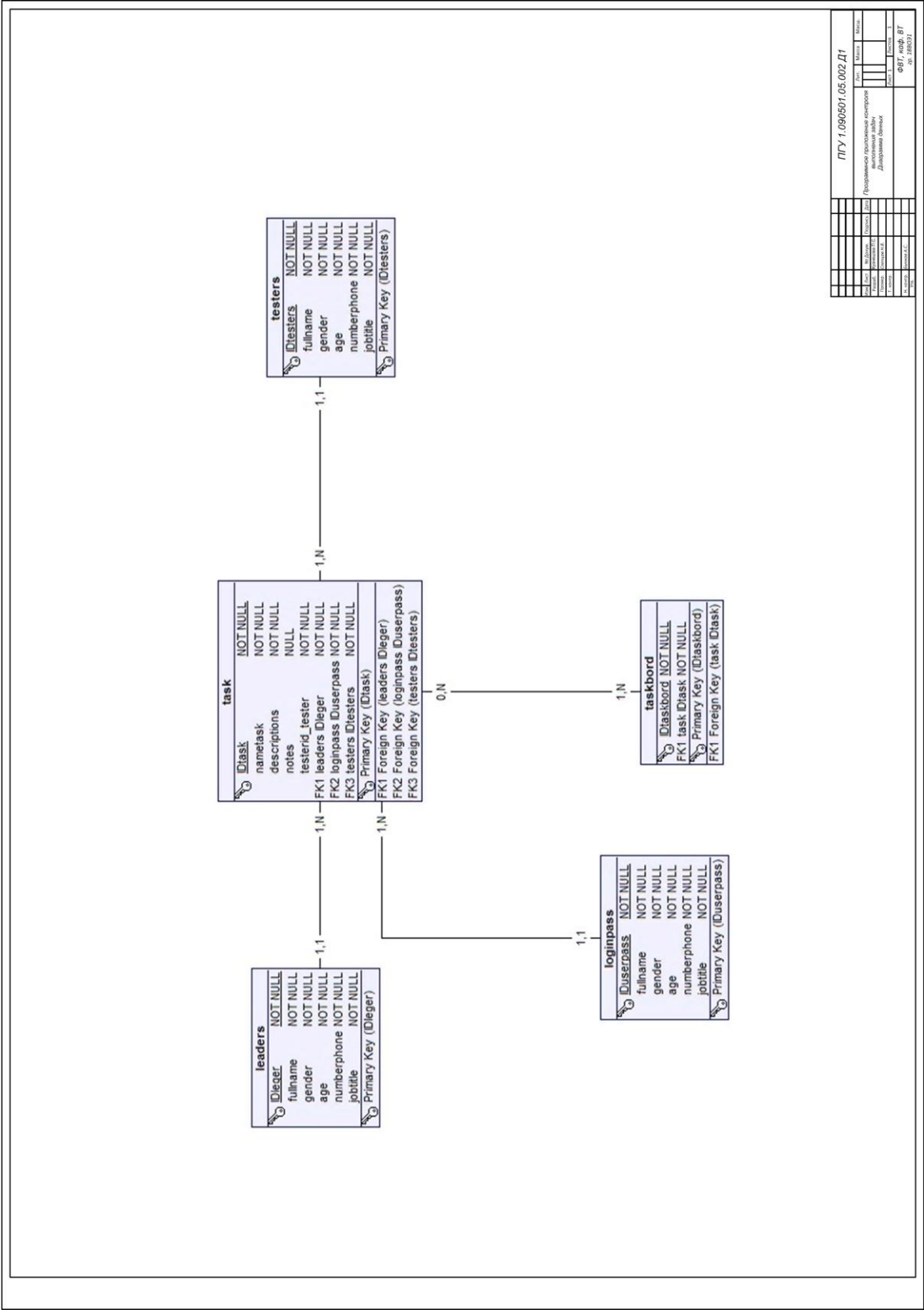
Приложение А.3 – Диаграмма развертывания



ПГУ 1.090501.05.002 Д19									
Программное обеспечение системы									
Версия 1.0									
ФВТ, код ФВТ									
ФВТ, код ФВТ									
ФВТ, код ФВТ									
ФВТ, код ФВТ									
ФВТ, код ФВТ									
ФВТ, код ФВТ									
ФВТ, код ФВТ									

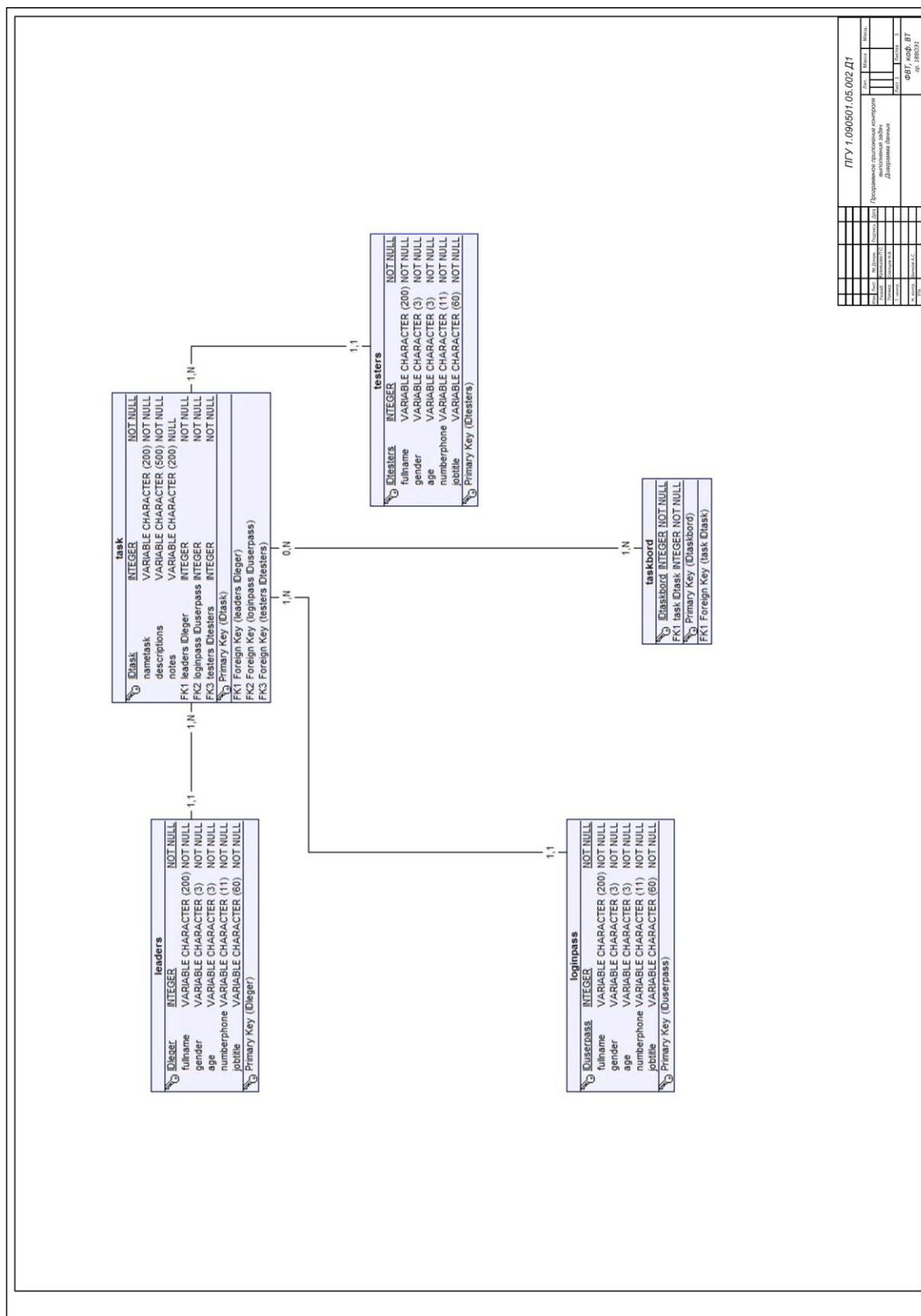
Из	Лист	№ докум.	Подпись	Дата

Приложение А.4 – Логическая модель данных



ПГУ 1.090501.05.002 Д1									
Информационно-технологическая система									
Информационная модель									
Дистрибутивная база									
ФОРМ. таб. ВГ									
до 2002021									

Приложение А.5 – Физическая модель данных



Приложение Б

(обязательное)

Программное приложение контроля выполнения задач
SQL-скрипт

					ПГУ1.090501.05.001 ПЗ	Лист
						74
Из	Лист	№ докум.	Подпись	Дата		

```

-- Table: public.leaders

-- DROP TABLE IF EXISTS public.leaders;

CREATE TABLE IF NOT EXISTS public.leaders

(
    "IDleger" integer NOT NULL GENERATED ALWAYS AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    fullname          character          varying(200)          COLLATE
pg_catalog."default" NOT NULL,

    gender character varying(3) COLLATE pg_catalog."default"
NOT NULL,

    age character varying(3) COLLATE pg_catalog."default" NOT
NULL,

    numberphone        character          varying(11)          COLLATE
pg_catalog."default" NOT NULL,

    "      jobtitle"    character          varying(60)          COLLATE
pg_catalog."default" NOT NULL,

    CONSTRAINT leaders_pkey PRIMARY KEY ("IDleger")

)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.leaders

    OWNER to postgres;

-- Table: public.loginpass

-- DROP TABLE IF EXISTS public.loginpass;

CREATE TABLE IF NOT EXISTS public.loginpass

(

```

					ПГУ1.090501.05.001 ПЗ	Лист
						75
Из	Лист	№ докум.	Подпись	Дата		

```

        "IDuserpass" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

        fullname character varying(200) COLLATE
pg_catalog."default" NOT NULL,

        gender character varying(3) COLLATE pg_catalog."default"
NOT NULL,

        age character varying(3) COLLATE pg_catalog."default" NOT
NULL,

        numberphone character varying(11) COLLATE
pg_catalog."default" NOT NULL,

        "jobtitle" character varying(60) COLLATE
pg_catalog."default" NOT NULL,

        CONSTRAINT loginpass_pkey PRIMARY KEY ("IDuserpass")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.loginpass

OWNER to postgres;

-- Table: public.task

-- DROP TABLE IF EXISTS public.task;

CREATE TABLE IF NOT EXISTS public.task

(

        "IDtask" integer NOT NULL GENERATED ALWAYS AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

        nametask character varying(200) COLLATE
pg_catalog."default" NOT NULL,

        descriptions character varying(500) COLLATE
pg_catalog."default" NOT NULL,

```

					ПГУ1.090501.05.001 ПЗ	Лист
						76
Из	Лист	№ докум.	Подпись	Дата		

```

    notes character varying(200) COLLATE pg_catalog."default"
NOT NULL,

    testerid_tester integer NOT NULL,

    leadersid_led integer NOT NULL,

    loginpassid_user integer NOT NULL,

    CONSTRAINT task_pkey PRIMARY KEY ("IDtask"),

    CONSTRAINT leder_fkey FOREIGN KEY (leadersid_led)

        REFERENCES public.leaders ("IDleger") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT loginpassid_fkey FOREIGN KEY

(loginpassid_user)

        REFERENCES public.loginpass ("IDuserpass") MATCH

SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT testers_fkey FOREIGN KEY (testerid_tester)

        REFERENCES public.testers ("IDtesters") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)

TABLESPACE pg_default;
```

					ПГУ1.090501.05.001 ПЗ	Лист
						77
Из	Лист	№ докум.	Подпись	Дата		

```

ALTER TABLE IF EXISTS public.task

    OWNER to postgres;

-- Index: fki_testers_fkey

-- DROP INDEX IF EXISTS public.fki_testers_fkey;

CREATE INDEX IF NOT EXISTS fki_testers_fkey

    ON public.task USING btree

    (testerid_tester ASC NULLS LAST)

    TABLESPACE pg_default;

-- Table: public.taskbord

-- DROP TABLE IF EXISTS public.taskbord;

CREATE TABLE IF NOT EXISTS public.taskbord

(

    "IDtaskbord" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

    taskid_task integer NOT NULL,

    CONSTRAINT taskbord_pkey PRIMARY KEY ("IDtaskbord"),

    CONSTRAINT task_fkey FOREIGN KEY (taskid_task)

        REFERENCES public.task ("IDtask") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)

```

					ПГУ1.090501.05.001 ПЗ	Лист
						78
Из	Лист	№ докум.	Подпись	Дата		

```

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.taskbord

    OWNER to postgres;

-- Table: public.testers

-- DROP TABLE IF EXISTS public.testers;

CREATE TABLE IF NOT EXISTS public.testers

(
    "IDtesters" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1
),

    fullname          character          varying(200)          COLLATE
pg_catalog."default" NOT NULL,

    gender character varying(3) COLLATE pg_catalog."default"
NOT NULL,

    age character varying(3) COLLATE pg_catalog."default" NOT
NULL,

    numberphone        character          varying(11)          COLLATE
pg_catalog."default" NOT NULL,

    "      jobtitle"    character          varying(60)          COLLATE
pg_catalog."default" NOT NULL,

    CONSTRAINT testers_pkey PRIMARY KEY ("IDtesters")

)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.testers

    OWNER to postgres;

```

					ПГУ1.090501.05.001 ПЗ	Лист
						79
Из	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Программное приложение контроля выполнения задач

Результаты испытаний

					ПГУ1.090501.05.001 ПЗ	Лист
						80
Из	Лист	№ докум.	Подпись	Дата		

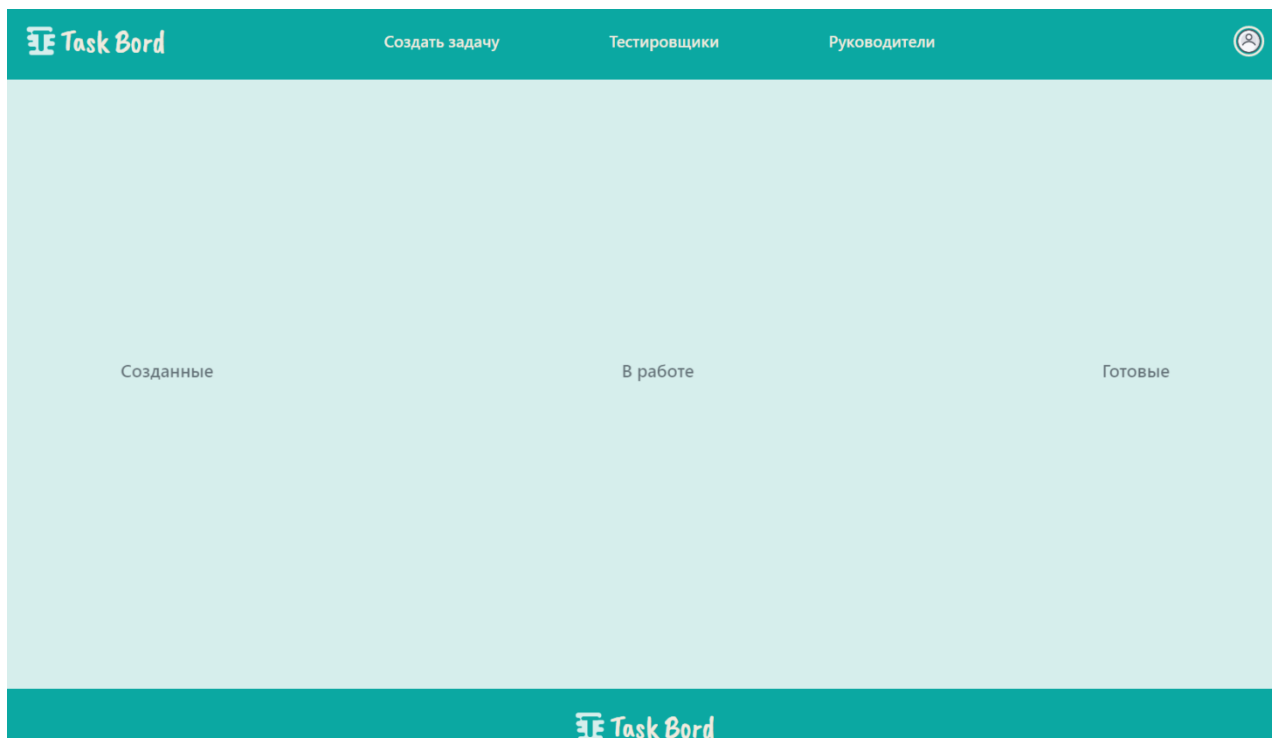


Рисунок В.1 - Главная доска для тестовых случаев

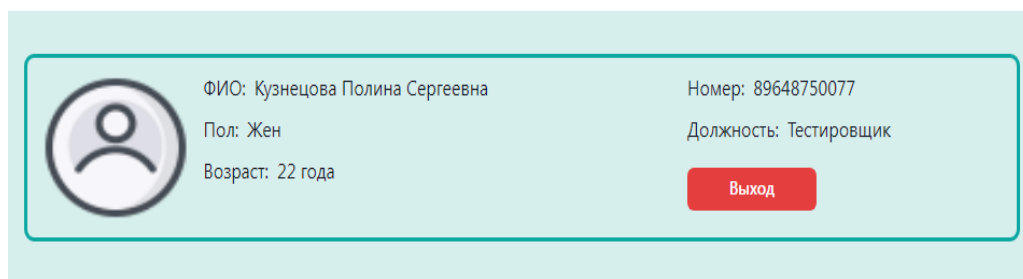


Рисунок В.2 – Данные с личной станицы пользователя



Рисунок В.3 – Кнопка перехода к основной доске приложения



Рисунок В.4 – Выбор пункта меню «Тестировщики»

					ПГУ1.090501.05.001 ПЗ	Лист
						81
Из	Лист	№ докум.	Подпись	Дата		

Task Bord

Создать задачу

Тестировщики

Руководители

Добавление нового тестировщика

ФИО

Пол

☐ Мужской
☐ Женский

Возраст

Номер телефона

Должность

Добавить тестировщика

Тестировщики

ФИО: Кузнецова Полина Сергеевна
Пол: Женский
Возраст: 22

Номер: 89648750077

Должность: Тестировщик

×

Рисунок В.5 – Пункт меню тестировщики

Добавление нового тестировщика

ФИО

Иванов Иван Иванович

Пол

☒ Мужской
☐ Женский

Возраст

30

Номер телефона

893873549090

Должность

Тестировщик

Добавить тестировщика

Рисунок В.6 – Добавление нового тестировщика

Добавление нового тестировщика

ФИО

Пол

☐ Мужской
☐ Женский

Возраст

30

Номер телефона

893873549090

Должность

Добавить тестировщика

Тестировщики

ФИО: Кузнецова Полина Сергеевна
Пол: Женский
Возраст: 22

Номер: 89648750077

Должность: Тестировщик

×

ФИО: Иванов Иван Иванович
Пол: Мужской
Возраст: 30

Номер: 893873549090

Должность: Тестировщик

×

Рисунок В.7 – Отображение добавленного пользователя

Создание задачи

Название

Описание

Тестировщик

Создать задачу

Рисунок В.8 – Экран создания новой задачи

Создание задачи

Название

Описание

Тестировщик

Создать задачу

Рисунок В.9 – Заполненная карточка добавления тестового случая

Созданные

**Некорректная работа
стартовой страницы
приложения**

Тестировщик:
Полина Кузнецова

Дата создания:
16.06.2023

Подробнее **Удалить**

Рисунок В.10 – Созданная карточка задачи

Некорректная работа стартовой страницы приложения

При открытии стартовой страницы, происходит зависание программы

Тестировщик: Полина Кузнецова

Статус: Создана

Дата создания: 16.06.2023

☐ Проверка совпадения интерфейса проверяемой функции техническому заданию
 ☐ Проверка функциональных характеристик функции на основе технического задания
 ☐ Анализ полученных результатов

Комментарий к задаче:

Перевести в работу

Редактировать

Рисунок В.11 – Открыта подробная информация о тестовом случае

Некорректная работа стартовой страницы приложения

При открытии стартовой страницы, происходит зависание программы

Тестировщик:

Полина Кузнецова

Статус: Создана

Дата создания: 16.06.2023

☒ Проверка совпадения интерфейса проверяемой функции техническому заданию
 ☒ Проверка функциональных характеристик функции на основе технического задания
 ☒ Анализ полученных результатов

Комментарий к задаче:

Пройден 1 этап тестирования

Сохранить

Рисунок В.12 – Окно редактирования задачи

Task Bord

Создать задачу

Тестировщики

Руководители

Созданные

В работе

Готовые

Некорректная работа стартовой страницы приложения

Тестирующая: Полина Кузнецова

Дата создания: 16.06.2023

Изменить

Закрыть

Рисунок В.13 – Изменение статуса на «В работе»

					ПГУ1.090501.05.001 ПЗ	Лист
						84
Из	Лист	№ докум.	Подпись	Дата		

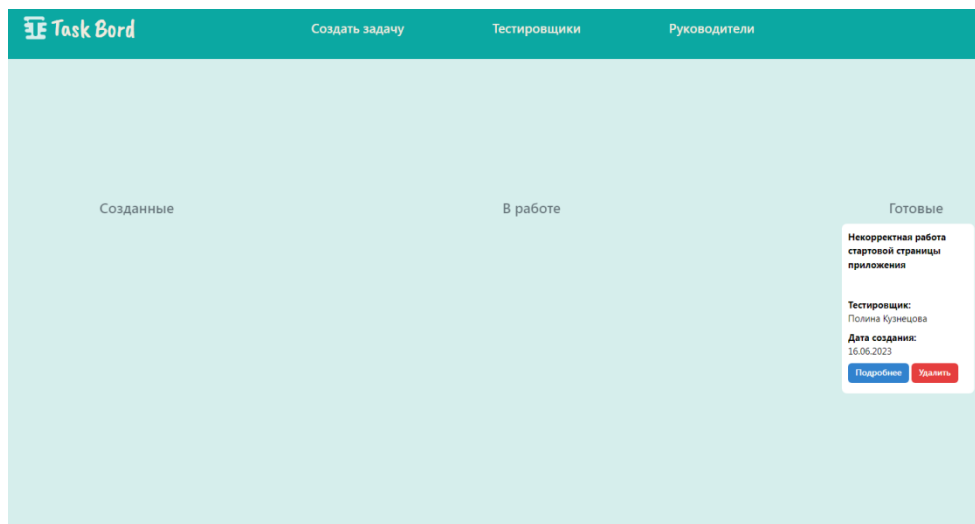


Рисунок В.14 – Переход задачи в режим «Готовые»

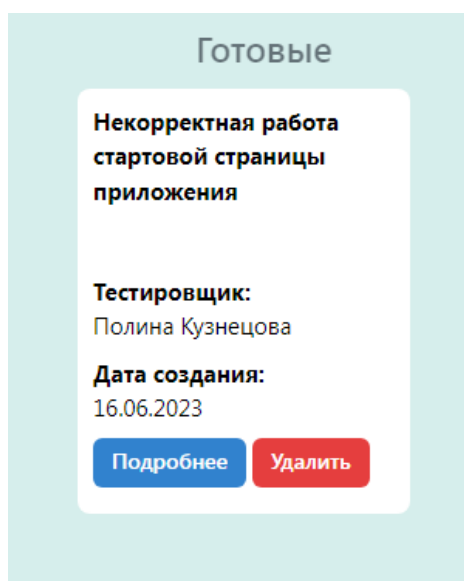


Рисунок В.15 – Удаление карточки задачи

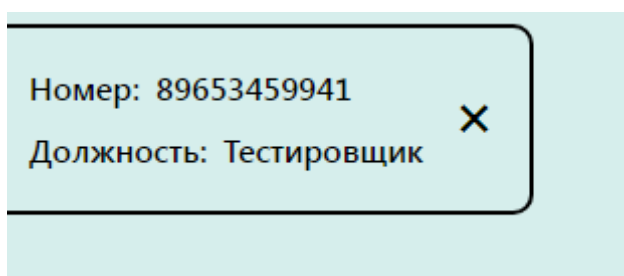


Рисунок В.16 – Удаление пользователя

Приложение Г

(обязательное)

Программное приложение контроля выполнения задач

Листинг программы

					ПГУ1.090501.05.001 ПЗ	Лист
						86
Из	Лист	№ докум.	Подпись	Дата		

```

import React, { FC, useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import styles from '../styles/index.module.css'
import Card from '../../components/Card';

interface Props {
  createdTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  workTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  completedTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  setCreatedTasks: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string
    type: number,
    testCheck: number,
    taskComment: string,
  }) => void,
  setWorkTasks: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string
    type: number,
    testCheck: number,
    taskComment: string,
  }) => void,
  setCompletedTasks: (params:{

```

					ПГУ1.090501.05.001 ПЗ	Лист
						87
Из	Лист	№ докум.	Подпись	Дата		

```

        time: number,
        taskName: string,
        taskDescription: string,
        taskTester: string
        type: number,
        testCheck: number,
        taskComment: string,
    }[]) => void,
    setTaskInfo: (params:{
        time: number,
        taskName: string,
        taskDescription: string,
        taskTester: string,
        type: number,
        testCheck: number,
        taskComment: string,
    }) => void,
}

const Home: FC<Props> = ({
    createdTasks,
    workTasks,
    completedTasks,
    setTaskInfo,
    setCreatedTasks,
    setWorkTasks,
    setCompletedTasks,
}) => {
    const renderCreatedTasks = () => {
        if (createdTasks.length !== 0) {
            return (
                createdTasks.map(el => (
                    <Card
                        key={ el.time }
                        time={ el.time }
                        taskName={ el.taskName }
                        taskDescription={ el.taskDescription }
                        taskTester={ el.taskTester }
                        type={ el.type }
                        testCheck={ el.testCheck }
                        taskComment={ el.taskComment }
                        setTaskInfo={ setTaskInfo }
                        createdTasks={ createdTasks }
                        setCreatedTasks={ setCreatedTasks }
                        workTasks={ workTasks }
                        setWorkTasks={ setWorkTasks }
                        completedTasks={ completedTasks }
                        setCompletedTasks={ setCompletedTasks }
                    />
                ))
            )
        }
    }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						88
Из	Лист	№ докум.	Подпись	Дата		

```

    return null
  }

  const renderWorkTasks = () => {
    if (workTasks.length !== 0) {
      return (
        workTasks.map(el => (
          <Card
            key={ el.time }
            time={ el.time }
            taskName={ el.taskName }
            taskDescription={ el.taskDescription }
            taskTester={ el.taskTester }
            type={ el.type }
            testCheck={ el.testCheck }
            taskComment={ el.taskComment }
            setTaskInfo={ setTaskInfo }
            createdTasks={ createdTasks }
            setCreatedTasks={ setCreatedTasks }
            workTasks={ workTasks }
            setWorkTasks={ setWorkTasks }
            completedTasks={ completedTasks }
            setCompletedTasks={ setCompletedTasks }
          />
        ))
      )
    }
    return null
  }

  const renderCompletedTasks = () => {
    if (completedTasks.length !== 0) {
      return (
        completedTasks.map(el => (
          <Card
            key={ el.time }
            time={ el.time }
            taskName={ el.taskName }
            taskDescription={ el.taskDescription }
            taskTester={ el.taskTester }
            type={ el.type }
            testCheck={ el.testCheck }
            taskComment={ el.taskComment }
            setTaskInfo={ setTaskInfo }
            createdTasks={ createdTasks }
            setCreatedTasks={ setCreatedTasks }
            workTasks={ workTasks }
            setWorkTasks={ setWorkTasks }
            completedTasks={ completedTasks }
            setCompletedTasks={ setCompletedTasks }
          />
        ))
      )
    }
  }

```

```

    )
  }
  return null
}

return (
  <div className={ styles.wrapper_all }>
    <div className={ styles.wrapper_title_app }>
      <p className={ styles.title_all_move }>Созданные</p>
      <p className={ styles.title_all_move }>В работе</p>
      <p className={ styles.title_all_move }>Готовые</p>
    </div>
    <section className={ styles.wrapper_moveList }>
      <div className={ styles.wrapper_posters }>
        <div className={ styles.wrapper_card }>
          {renderCreatedTasks()}
        </div>
      </div>
      <div className={ styles.wrapper_posters }>
        <div className={ styles.wrapper_card }>
          {renderWorkTasks()}
        </div>
      </div>
      <div className={ styles.wrapper_posters }>
        <div className={ styles.wrapper_card }>
          {renderCompletedTasks()}
        </div>
      </div>
    </section>
  </div>
);
}

export default Home;
-----

import React, { FC, useEffect, useState } from 'react';
import PropTypes, { bool, number } from 'prop-types';
import { observer } from 'mobx-react'
import {
  Button, Input, Select, Textarea,
  Checkbox,
} from '@chakra-ui/react';
import styles from './styles/index.module.css'

interface Props {
  testers:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberTester: number,

```

					ПГУ1.090501.05.001 ПЗ	Лист
						90
Из	Лист	№ докум.	Подпись	Дата		

```

    post: string,
  }[],
  taskInfo:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  },
  createdTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  workTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  completedTasks:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[],
  setCreatedTasks: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }) => void,
  setWorkTasks: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string

```

					ПГУ1.090501.05.001 ПЗ	Лист
						91
Из	Лист	№ докум.	Подпись	Дата		

```

        type: number,
        testCheck: number,
        taskComment: string,
    }[]) => void,
    setCompletedTasks: (params:{
        time: number,
        taskName: string,
        taskDescription: string,
        taskTester: string
        type: number,
        testCheck: number,
        taskComment: string,
    }[]) => void,
    }

const TaskPage: FC<Props> = observer(({
    taskInfo: {
        time,
        taskName,
        taskDescription,
        taskTester,
        type,
        testCheck,
        taskComment,
    },
    testers,
    createdTasks,
    workTasks,
    completedTasks,
    setCreatedTasks,
    setWorkTasks,
    setCompletedTasks,
}) => {
    const [ taskEdit, setTaskEdit ] = useState<boolean>(false)
    const [ newTaskName, setNewTaskName ] =
    useState<string>(taskName)
    const [ newTaskDescription, setNewTaskDescription ] =
    useState<string>(taskDescription)
    const [ newTaskTester, setNewTaskTester ] =
    useState<string>(taskTester)
    const [ newTestCheck, setNewTestCheck ] =
    useState<number>(testCheck)
    console.log('🔍 ~ newTestCheck:', newTestCheck)
    const [ newTaskComment, setNewTaskComment ] =
    useState<string>(taskComment)

    let status = ''

    if (type === 1) {
        status = 'Создана'
    }
    if (type === 2) {

```

					ПГУ1.090501.05.001 ПЗ	Лист
						92
Из	Лист	№ докум.	Подпись	Дата		

```

        status = 'В работе'
    }
    if (type === 3) {
        status = 'Готова'
    }

    const date = new Date(time)
    const dateString = date.toLocaleDateString()

    const handleTaskWork = () => {
        setWorkTasks([
            time,
            taskName,
            taskDescription,
            taskTester,
            type: 2,
            testCheck: newTestCheck,
            taskComment: newTaskComment,
        ])
        localStorage.setItem('workTasks', JSON.stringify([
...workTasks, {
            time,
            taskName,
            taskDescription,
            taskTester,
            type: 2,
            testCheck: newTestCheck,
            taskComment: newTaskComment,
        }]))

        const newCreatedTasks = createdTasks.filter(el => el.time !==
time)
        setCreatedTasks(newCreatedTasks)
        localStorage.setItem('createdTasks',
JSON.stringify(newCreatedTasks))
    }

    const handleCompletedTasks = () => {
        setCompletedTasks([
            time,
            taskName,
            taskDescription,
            taskTester,
            type: 3,
            testCheck: newTestCheck,
            taskComment: newTaskComment,
        ])
        localStorage.setItem('completedTasks', JSON.stringify([
...completedTasks, {
            time,
            taskName,
            taskDescription,

```

					ПГУ1.090501.05.001 ПЗ	Лист
						93
Из	Лист	№ докум.	Подпись	Дата		

```

        taskTester,
        type: 3,
        testCheck: newTestCheck,
        taskComment: newTaskComment,
    ]]))

    const newWorkTasks = workTasks.filter(el => el.time !== time)
    setWorkTasks(newWorkTasks)
    localStorage.setItem('workTasks',
JSON.stringify(newWorkTasks))
    }

    const setTaskEditSave = () => {
        const idxCreatedTasks = createdTasks.findIndex(el => el.time
=== time)
        console.log('🔍 ~ idxCreatedTasks:', idxCreatedTasks)
        console.log('🔍 ~ newTaskComment:', newTaskComment)
        if (idxCreatedTasks >= 0) {
            const newCreatedTasks = createdTasks.slice()
            newCreatedTasks[idxCreatedTasks] = {
                time,
                taskName: newTaskName,
                taskDescription: newTaskDescription,
                taskTester: newTaskTester,
                type,
                testCheck: newTestCheck,
                taskComment: newTaskComment,
            }

            setCreatedTasks(newCreatedTasks)
            localStorage.setItem('createdTasks',
JSON.stringify(newCreatedTasks))
            setTaskEdit(false)
            return
        }

        const idxWorkTasks = workTasks.findIndex(el => el.time ===
time)

        if (idxWorkTasks >= 0) {
            const newWorkTasks = workTasks.slice()

            newWorkTasks[idxWorkTasks] = {
                time,
                taskName: newTaskName,
                taskDescription: newTaskDescription,
                taskTester: newTaskTester,
                type,
                testCheck: newTestCheck,
                taskComment: newTaskComment,
            }
            setWorkTasks(newWorkTasks)
        }
    }

```

```

        localStorage.setItem('workTasks',
JSON.stringify(newWorkTasks))

        setTaskEdit(false)
        return
    }

    const idxCompletedTasks = completedTasks.findIndex(el =>
el.time === time)

    if (idxCompletedTasks >= 0) {
        const newCompletedTasks = completedTasks.slice()

        newCompletedTasks[idxCompletedTasks] = {
            time,
            taskName: newTaskName,
            taskDescription: newTaskDescription,
            taskTester: newTaskTester,
            type,
            testCheck: newTestCheck,
            taskComment: newTaskComment,
        }

        setCompletedTasks(newCompletedTasks)
        localStorage.setItem('completedTasks',
JSON.stringify(newCompletedTasks))
        setTaskEdit(false)
    }
}

const testCheckStart = (stage:number) => {
    console.log('🕒 ~ stage:', stage)
    if (stage === 1) {
        if (newTestCheck === 0 || newTestCheck === 2 || newTestCheck
=== 3) {
            setNewTestCheck(1)
        } else if (newTestCheck === 1) {
            setNewTestCheck(0)
        }
    } else if (stage === 2) {
        if (newTestCheck === 0 || newTestCheck === 1 || newTestCheck
=== 3) {
            setNewTestCheck(2)
        } else if (newTestCheck === 2) {
            setNewTestCheck(1)
        }
    } else if (stage === 3) {
        if (newTestCheck === 0 || newTestCheck === 1 || newTestCheck
=== 2) {
            setNewTestCheck(3)
        } else if (newTestCheck === 3) {
            setNewTestCheck(2)
        }
    }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						95
Из	Лист	№ докум.	Подпись	Дата		

```

    }
  }
}

const renderCheckTest = (flag:boolean) => (
  <div className={ styles.wrapper_checkbox }>
    <Checkbox
      isDisabled={ flag }
      isInvalid
      defaultChecked={ !!newTestCheck }
      onChange={ () => testCheckStart(1) }
    >
      Проверка совпадения интерфейса проверяемой функции
      технического заданию
    </Checkbox>
    <Checkbox
      isDisabled={ flag }
      isInvalid
      defaultChecked={ newTestCheck >= 2 }
      onChange={ () => testCheckStart(2) }
    >
      Проверка функциональных характеристик функции на основе
      технического задания
    </Checkbox>
    <Checkbox
      isDisabled={ flag }
      isInvalid
      defaultChecked={ newTestCheck === 3 }
      onChange={ () => testCheckStart(3) }
    >
      Анализ полученных результатов
    </Checkbox>
  </div>
)

const renderButton = () => {
  if (type === 1) {
    return (
      <div>
        <Button colorScheme="blue" onClick={ () =>
handleTaskWork() }>Перевести в работу</Button>
        <Button className={ styles.btn_edit } colorScheme="blue"
onClick={ () => setTaskEdit(true) }>Редактировать</Button>
        { /* <button>Задача готова</button> */ }
      </div>
    )
  }
  if (type === 2) {
    return (
      <div>
        { /* <button>Вернуть в созданные</button> */ }
      </div>
    )
  }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						96
Из	Лист	№ докум.	Подпись	Дата		

```

        <Button colorScheme="blue" onClick={ () =>
handleCompletedTasks() }>Задача готова</Button>
        <Button className={ styles.btn_edit } colorScheme="blue"
onClick={ () => setTaskEdit(true) }>Редактировать</Button>
    </div>
    )
}
if (type === 3) {
    return (
        <div>
            { /* <Button colorScheme="blue">Вернуть в
созданные</Button> */}
            { /* <Button colorScheme="blue">Вернуть в работу</Button>
*/}
            <Button className={ styles.btn_edit } colorScheme="blue"
onClick={ () => setTaskEdit(true) }>Редактировать</Button>
        </div>
    )
}
return null
}

if (taskEdit) {
    return (
        <div>
            <section className={ styles.wrapper_move_page }>
                <div className={ styles.wrapper_content }>
                    <Input
                        value={ newTaskName }
                        onChange={ e => setNewTaskName(e.target.value) }
                    />
                    <Textarea
                        value={ newTaskDescription }
                        onChange={ e =>
setNewTaskDescription(e.target.value) }
                    />
                    <p>
                        Тестировщик:&ensp;
                    </p>
                    <Select placeholder="-" onChange={ e =>
setNewTaskTester(e.target.value) }>
                        { testers.map(el => (
                            <option key={ el.id } value={ el.name
}>{el.name}</option>
                        ))}
                    </Select>
                    <p>
                        Статус:&ensp;
                        {status}
                    </p>
                    <p>
                        Дата создания:&ensp;

```

					ПГУ1.090501.05.001 ПЗ	Лист
						97
Из	Лист	№ докум.	Подпись	Дата		

```

        {dateString}
      </p>
      { renderCheckTest(false) }
      <strong>Комментарий к задаче:</strong>
      <Textarea
        value={ newTaskComment }
        onChange={ e => setNewTaskComment(e.target.value) }
      />
      <Button className={ styles.btn_edit }
colorScheme="blue" onClick={ () => setTaskEditSave()
}>Сохранить</Button>
    </div>
  </section>
</div>
)
}

return (
  <div>
    <section className={ styles.wrapper_move_page }>
      <div className={ styles.wrapper_content }>
        <strong className={ styles.title
}>{newTaskName}</strong>
        <p>{newTaskDescription}</p>
        <p>
          Тестировщик:&ensp;
          {newTaskTester}
        </p>
        <p>
          Статус:&ensp;
          {status}
        </p>
        <p>
          Дата создания:&ensp;
          {dateString}
        </p>
        { renderCheckTest(true) }
        <p>Комментарий к задаче:</p>
        <p>{newTaskComment}</p>
        { renderButton() }
      </div>
    </section>
  </div>
)
})

export default TaskPage;
-----

import React, { FC, useEffect, useState } from 'react';
import {
  Button, Input, Select, Textarea,

```

```

} from '@chakra-ui/react'

import styles from '../styles/index.module.css'

const LABELS = {
  CREATE_TASK: 'Создание задачи',
}

interface Props {
  setCreatedTasks: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }) => void,
  testers:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberTester: number,
    post: string,
  }[],
  createdTasks:[]
}

const CreateTask: FC<Props> = ({ setCreatedTasks, createdTasks,
testers }) => {
  const [ taskName, setTaskName ] = useState<string>('')
  const [ taskDescription, setTaskDescription ] =
useState<string>('')
  const [ taskTester, setTaskTester ] = useState<string>('')

  const hendaleCreateTask = () => {
    const date = new Date().getTime()
    const type = 1
    const testCheck = 0
    const taskComment = ''

    setCreatedTasks([ ...createdTasks, {
      time: date, taskName, taskDescription, taskTester, type,
testCheck, taskComment,
    }])
    localStorage.setItem('createdTasks', JSON.stringify([
...createdTasks, {
  time: date, taskName, taskDescription, taskTester, type,
testCheck, taskComment,
}]))
    setTaskName('')
  }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						99
Из	Лист	№ докум.	Подпись	Дата		

```

        setTaskDescription('')
    }

    return (
        <div>
            <section className={ styles.wrapper_moveList }>
                <div className={ styles.wrapper_posters }>
                    <p className={ styles.wrapper_title
}>{LABELS.CREATE_TASK}</p>
                    <div className={ styles.wrapper_content }>
                        <p>Название</p>
                        <Input
                            placeholder="Название"
                            value={ taskName }
                            onChange={ e => setTaskName(e.target.value) }
                        />
                        <p>Описание</p>
                        <Textarea
                            placeholder="Описание"
                            value={ taskDescription }
                            onChange={ e => setTaskDescription(e.target.value) }
                        />
                        <p>Тестировщик</p>
                        <Select placeholder="-" onChange={ e =>
setTaskTester(e.target.value) }>
                            { testers.map(el => (
                                <option key={ el.id } value={ el.name
}>{el.name}</option>
                            ))}
                        </Select>
                        <Button colorScheme="blue" onClick={ () =>
hendaleCreateTask() }>
                            Создать задачу
                        </Button>
                    </div>
                </div>
            </section>
        </div>
    );
}

export default CreateTask;
----

// @ts-nocheck
import React, { FC, useEffect, useState } from 'react';

import {
    Button, Input,
} from '@chakra-ui/react';
import iconTester from '../image/icon_tester.png'
import styles from './styles/index.module.css'

```

					ПГУ1.090501.05.001 ПЗ	Лист
						100
Из	Лист	№ докум.	Подпись	Дата		

```

interface Props {
  managers:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberManager: number,
    post: string,
  }[],
  setManagers: (params:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberManager: number,
    post: string,
  }[]) => void,
}

const Managers: FC<Props> = ({
  managers,
  setManagers,
}) => {
  const [ name, setName ] = useState<string>('')
  const [ floor, setFloor ] = useState<string>('')
  const [ age, setAge ] = useState<number>()
  const [ numberManager, setNumberManager ] = useState<number>()
  const [ post, setPost ] = useState<string>('')

  const hendaleCreateManagers = () => {
    const id = new Date().getTime()

    setManagers([ ...managers, {
      id, name, floor, age, numberManager, post,
    }])
    localStorage.setItem('managers', JSON.stringify([ ...managers,
    {
      id, name, floor, age, numberManager, post,
    }]))
    setName('')
    setFloor('')
    setAge(0)
    setNumberManager(0)
    setPost('')
  }

  const deleteCard = id => {
    const newManagers = managers.filter(e => e.id !== id)

    setManagers(newManagers)
    localStorage.setItem('managers', JSON.stringify(newManagers))
  }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						101
Из	Лист	№ докум.	Подпись	Дата		

```

    }

    const renderManagers = () => {
      if (managers.length !== 0) {
        return (
          managers.map(el => (
            <div className={ styles.wrapper_card_managers } key={
el.id }>
              <img alt="user_logo" src={ iconTester } className={
styles.wrapper_user_logo } />
              <div className={ styles.fio_info }>
                <p>
                  ФИО:&ensp;
                  {el.name}
                </p>
                <p>
                  Пол:&ensp;
                  {el.floor}
                </p>
                <p>
                  Возраст:&ensp;
                  {el.age}
                </p>
              </div>
              <div className={ styles.number_info }>
                <p>
                  Номер:&ensp;
                  {el.numberManager}
                </p>
                <p>
                  Должность:&ensp;
                  {el.post}
                </p>
              </div>
              <div className={ styles.button_edit }>
                { /* <button>&#9998;</button> */ }
                <button onClick={ () => deleteCard(el.id)
}>&#10060;</button>
              </div>
            </div>
          ))
        )
      }
      return null
    }

    return (
      <div>
        <section className={ styles.wrapper_moveList }>
          <div className={ styles.wrapper_add_managers }>
            <strong>Добавление нового руководителя</strong>
            <p>ФИО</p>

```

					ПГУ1.090501.05.001 ПЗ	Лист
						102
Из	Лист	№ докум.	Подпись	Дата		

```

    <Input
      value={ name }
      onChange={ e => setName(e.target.value) }
    />
    <p>Пол</p>
    <Input
      value={ floor }
      onChange={ e => setFloor(e.target.value) }
    />
    <p>Возраст</p>
    <Input
      value={ age }
      onChange={ e => setAge(+e.target.value) }
    />
    { /* <NumberInput >
      <NumberInputField />
      <NumberInputStepper >
        <NumberIncrementStepper />
        <NumberDecrementStepper />
      </NumberInputStepper>
    </NumberInput> */ }
    <p>Номер телефона</p>
    <Input
      value={ numberManager }
      onChange={ e => setNumberManager(+e.target.value) }
    />
    <p>Должность</p>
    <Input
      value={ post }
      onChange={ e => setPost(e.target.value) }
    />
    <Button colorScheme="blue" onClick={ () =>
hendaleCreateManagers() }>
      Добавить руководителя
    </Button>
  </div>
  <div className={ styles.wrapper_managers }>
    <strong>Руководители</strong>
    {renderManagers()}
  </div>
</section>
</div>
);
}

export default Managers;
----

// @ts-nocheck
import React, { FC, useEffect, useState } from 'react';

import { Button, Input } from '@chakra-ui/react';

```

					ПГУ1.090501.05.001 ПЗ	Лист
						103
Из	Лист	№ докум.	Подпись	Дата		

```

import iconTester from '../../image/icon_tester.png'
import styles from './styles/index.module.css'

interface Props {
  testers:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberTester: number,
    post: string,
  }[],
  setTesters: (params:{
    id: number,
    name: string,
    floor: string,
    age: number,
    numberTester: number,
    post: string,
  }[]) => void,
}

const Testers: FC<Props> = ({
  testers,
  setTesters,
}) => {
  const [ name, setName ] = useState<string>('')
  const [ floor, setFloor ] = useState<string>('')
  const [ age, setAge ] = useState<number>()
  const [ numberTester, setNumberTester ] = useState<number>()
  const [ post, setPost ] = useState<string>('')

  const hendaleCreateTesters = () => {
    const id = new Date().getTime()

    setTesters([ ...testers, {
      id, name, floor, age, numberTester, post,
    }])
    localStorage.setItem('testers', JSON.stringify([ ...testers, {
      id, name, floor, age, numberTester, post,
    }]))
    setName('')
    setFloor('')
    setAge(0)
    setNumberTester(0)
    setPost('')
  }

  const deleteCard = id => {
    const newTesters = testers.filter(e => e.id !== id)

    setTesters(newTesters)
  }
}

```

					ПГУ1.090501.05.001 ПЗ	Лист
						104
Из	Лист	№ докум.	Подпись	Дата		

```

    localStorage.setItem('testers', JSON.stringify(newTesters))
  }

  const renderTesters = () => {
    if (testers.length !== 0) {
      return (
        testers.map(el => (
          <div className={ styles.wrapper_card_tester } key={
el.id }>
            <img alt="user_logo" src={ iconTester } className={
styles.wrapper_user_logo } />
            <div className={ styles.fio_info }>
              <p>
                ФИО:&ensp;
                {el.name}
              </p>
              <p>
                Пол:&ensp;
                {el.floor}
              </p>
              <p>
                Возраст:&ensp;
                {el.age}
              </p>
            </div>
            <div className={ styles.number_info }>
              <p>
                Номер:&ensp;
                {el.numberTester}
              </p>
              <p>
                Должность:&ensp;
                {el.post}
              </p>
            </div>
            <div className={ styles.button_edit }>
              { /* <button>&#9998;</button> */ }
              <button onClick={ () => deleteCard(el.id)
}>&#10060;</button>
            </div>
          </div>
        ))
      )
    }
    return null
  }

  return (
    <div>
      <section className={ styles.wrapper_moveList }>
        <div className={ styles.wrapper_add_testers }>
          <strong>Добавление нового тестировщика</strong>

```

```

        <p>ФИО</p>
        <Input
          value={ name }
          onChange={ e => setName(e.target.value) }
        />
        <p>Пол</p>
        <Input
          value={ floor }
          onChange={ e => setFloor(e.target.value) }
        />
        <p>Возраст</p>
        <Input
          value={ age }
          onChange={ e => setAge(+e.target.value) }
        />
        <p>Номер телефона</p>
        <Input
          value={ numberTester }
          type="tel"
          onChange={ e => setNumberTester(+e.target.value) }
        />
        <p>Должность</p>
        <Input
          value={ post }
          onChange={ e => setPost(e.target.value) }
        />
        <Button colorScheme="blue" onClick={ () =>
hendaleCreateTesters() }>
          Добавить тестировщика
        </Button>
      </div>
      <div className={ styles.wrapper_testers }>
        <strong>Тестировщики</strong>
        {renderTesters()}
      </div>
    </section>
  </div>
);
}

export default Testers;
----

import React, { FC, useEffect, useState } from 'react';
import PropTypes, { number } from 'prop-types';
import { observer } from 'mobx-react';
import Card from '../components/Card';
import userLogo from '../image/icon_user.jpg'

import styles from './styles/index.module.css'

interface Props {

```

					ПГУ1.090501.05.001 ПЗ	Лист
						106
Из	Лист	№ докум.	Подпись	Дата		

```

cardFavorites: {
  idFavorit: number,
  posterFavorit: string,
  shortDescriptionFavorit: string,
}[],
addFavorites: (id:number, poster: string,
shortDescription:string)=>void,
getAllInfoOneMove: (id:string)=>void,
}

const LABELS = {
  TITLE_USER_PAGE: 'Личная страница',
  NAME_USER: 'Полина Кузнецова',
  GEN_USER: 'Жен',
  AGE_USER: '23 года',
  NUMBER: '8965451233',
  POST: 'Тестировщик',
}

const UserPage: FC<Props> = ({ cardFavorites, addFavorites,
getAllInfoOneMove }) => (
  <div>
    <section className={ styles.wrapper_usersr_page }>
      <div className={ styles.wrapper_user_card }>
        <div className={ styles.wrapper_user_info }>
          <img alt="user_logo" src={ userLogo } className={
styles.wrapper_user_logo } />
          <div className={ styles.fio_info }>
            <p>
              ФИО:&ensp;
              {LABELS.NAME_USER}
            </p>
            <p>
              Пол:&ensp;
              {LABELS.GEN_USER}
            </p>
            <p>
              Возраст:&ensp;
              {LABELS.AGE_USER}
            </p>
          </div>
          <div className={ styles.number_info }>
            <p>
              Номер:&ensp;
              {LABELS.NUMBER}
            </p>
            <p>
              Должность:&ensp;
              {LABELS.POST}
            </p>
          </div>
        </div>
      </div>
    </div>
  )

```

					ПГУ1.090501.05.001 ПЗ	Лист
						107
Из	Лист	№ докум.	Подпись	Дата		

```
        </div>
    </section>
</div>
)

export default UserPage;
```

					ПГУ1.090501.05.001 ПЗ	Лист
						108
Из	Лист	№ докум.	Подпись	Дата		

Приложение Д

(обязательное)

Программное приложение контроля выполнения задач

Презентация

					ПГУ1.090501.05.001 ПЗ	Лист
						109
Из	Лист	№ докум.	Подпись	Дата		

Программное приложение контроля выполнения задач

Автор работы:
студент гр. 18ВОЭ1 Кузнецова Полина Сергеевна
Научный руководитель:
к.т.н., доцент Слепцов Николай Владимирович

Пенза, 2023

Цель работы и объекты

Цель работы – разработать программное приложение контроля выполнения задач

Объектом проекта является программное приложение, выполняющее следующие функции:

- Работа в режиме тестирования
- Работа в режиме руководителя
- Работа в создании тестовых случаев

Задачи приложения

Реализовать приложение с последующей его загрузкой на GitHub

Данные пользователей должны храниться в базе данных

Данные внесенные пользователем должны отображаться на главной странице приложения в виде карточек

Из	Лист	№ докум.	Подпись	Дата

Сравнение аналогов

Функции	Jira	Flowlu	Kaiten	Штаб
Создание и редактирование задач	+	+	+	+
Изменение статуса задач	+	-	-	+
База данных	+	+	+	-
Шаблон для задач	+	+	+	+
Информация о пользователе	+	-	-	-
Доступность для команды проекта	+	+	+	+

Аналоги обладают недостатками:

- отсутствие возможности изменять статус созданных задач;
- невозможность хранения информации в базе данных;
- отсутствует минимальная информация о пользователях;
- вся команда имеет редактируемый доступ к задачам.

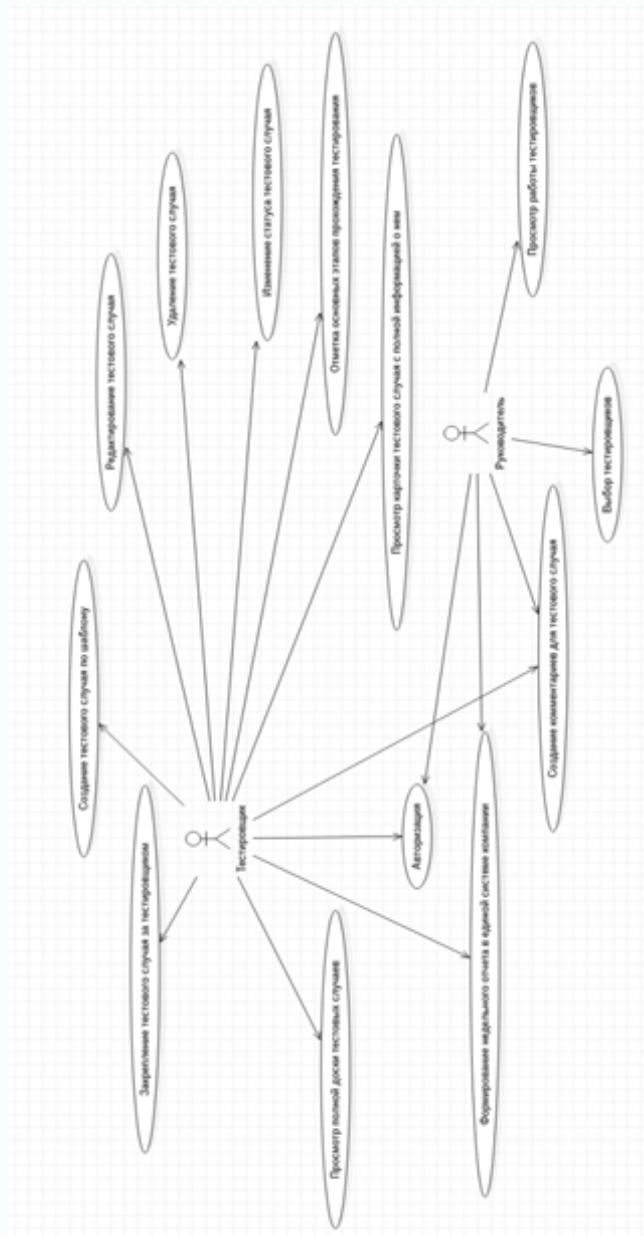
Аналогов для работы непосредственно специалистов по тестированию найдено не было.



Стек технологий

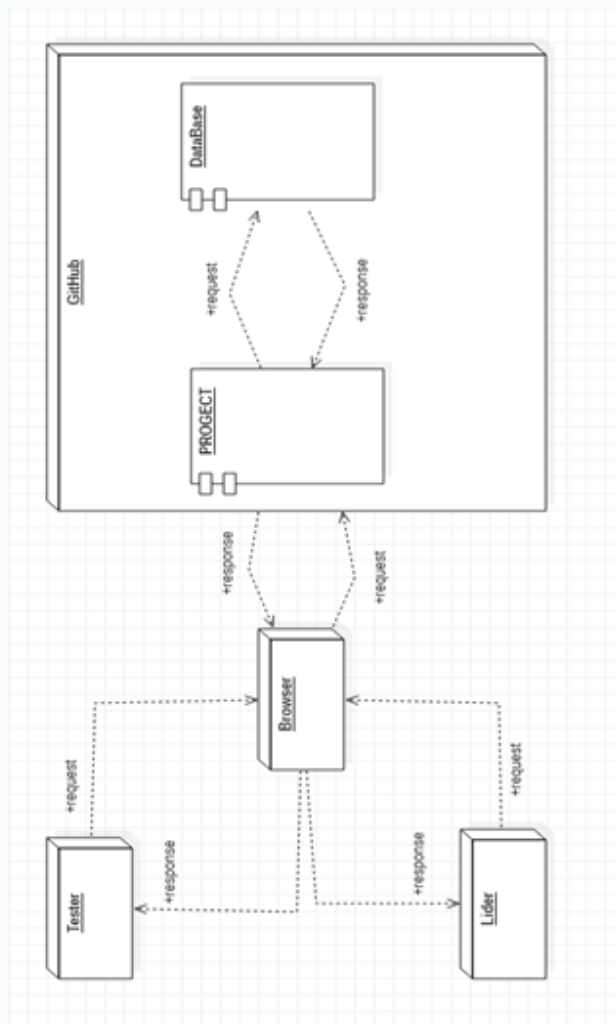


Основные функции приложения

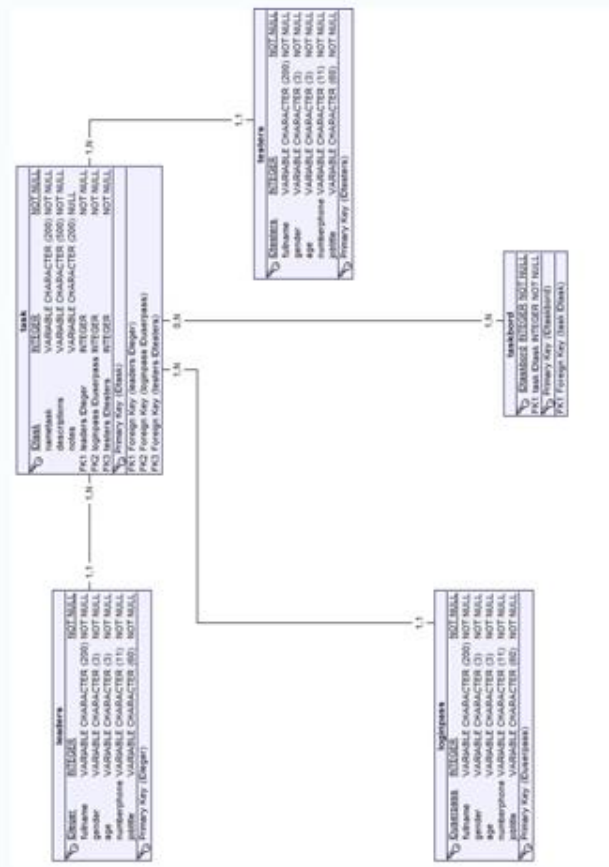


Из	Лист	№ докум.	Подпись	Дата

Архитектура приложения



База данных приложения



Скриншоты запущенного приложения

Task Board

Создать задачуТестированиеРуководство

Добавление нового тестирования

Имя

Фамилия

Пол

Возраст

Номер телефона

Должность

Добавить тестировщика

Создание

В работе

Готовые

Добавление нового тестирования

ФИО
Иванов Иван Иванович

Пол
☒ Мужской ☐ Женский

Возраст
30

Номер телефона
893873545090

Должность

Тестировщик

Добавить тестировщика

Готовые

Некорректная работа стартовой страницы приложения

Тестировщик:
Полина Кузнецова

Дата создания:
16.06.2023

Подробнее

Удалить

Task Board

Создать задачуТестированиеРуководство

Добавление нового тестирования

Имя

Фамилия

Пол

Возраст

Номер телефона

Должность

Добавить тестировщика

Создание

В работе

Готовые

ФИО: Кузнецова Полина Сергеевна
Пол: Жен
Возраст: 22 года

Номер: 89648750077
Должность: Тестировщик

Выход

Создание задачи

Название
Некорректная работа стартовой страницы приложения

Описание
При открытии стартовой страницы, происходит зависание программы

Тестировщик
Полина Кузнецова

Создать задачу

Результаты работы

- Разработано программное приложение полностью соответствующее техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!

студент гр. 18ВОЭ1 Кузнецова Полина Сергеевна

					ПГУ1.090501.05.001 ПЗ	Лист
						120
Из	Лист	№ докум.	Подпись	Дата		