

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

ПРОГРАММА ДЛЯ ИНТЕРАКТИВНОГО ТЕСТИРОВАНИЯ
IT-СПЕЦИАЛИСТОВ

Студент Исаева 18.09.2023 Исаева Виктория Евгеньевна
(подпись, дата) (ФИО полностью)

Руководитель Слепцов 19.09.2023 Слепцов Н.В.
(подпись, дата) (фамилия, инициалы)

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

Рецензент Авдеев (подпись, дата) (фамилия, инициалы)
(должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14)

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

Работа защищена с отметкой хорошо
(протокол заседания ГЭК от 21.09.2023 № 9)

Секретарь ГЭК Гурин Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«26» апреля 2023 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Исаева Виктория Евгеньевна гр. 18ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Программа для интерактивного тестирования IT-специалистов».

Тема утверждена приказом ПГУ № 458/0 от 26.04.2023 г.

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

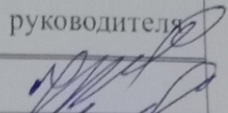
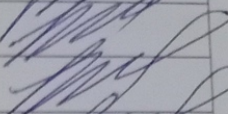
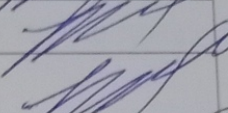
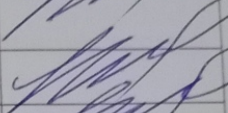
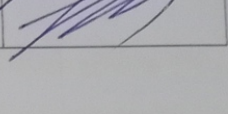
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Разработать программу для интерактивного тестирования IT специалистов. Программа должна выполнять следующие функции: предоставлять возможность для администратора (сотрудника HR отдела) создавать новые тесты; предоставлять возможность прохождения выбранных тестов; хранить в базе данных и отображать на экран результаты пройденных пользователями тестов; выполнять оценку результатов пройденного тестирования. Программа для интерактивного тестирования IT специалистов должна 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 инструменты реализации программы и развертывания приложения в облачной среде исполнения. Необходимо разработать алгоритм работы программы, разработать UML диаграммы 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	26.04.23-01.05.23	
2.	Проектирование программы	20	02.05.23-08.05.23	
3.	Разработка базы данных	10	09.05.23-14.05.23	
4.	Отладка и тестирование	10	15.05.23-20.05.23	
5.	Разработка экономической и БЖД части	15	21.05.23 - 25.05.23	
6.	Оформление графической части	10	26.05.23-28.05.23	
7.	Оформление выпускной работы	30	29.05.23-07.06.23	

Дата выдачи задания « 26 » 04. 2023 года.

Руководитель дипломного проекта _____ Сlepцов Н.В.
(подпись, дата)

Задание к исполнению принял студент _____ Исаева В.Е.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

Реферат

Пояснительная записка содержит 135 листов, 30 рисунков, 26 таблиц и 4 приложения

ПРОГРАММА, КОНТРОЛЬ СПЕЦИАЛИСТОВ, ВЕБ-РАЗРАБОТКА, БАЗА ДАННЫХ, ТЕСТЫ

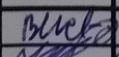
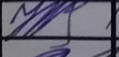
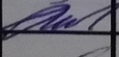
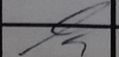
Объектом исследования является тестирование *IT*-специалистов на предприятии.

Цель работы – разработка программы для тестирования *IT*-специалистов на предприятии.

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

В результате проделанной работы разработана программа для интерактивного тестирования *IT* специалистов, состоящая из базы данных и веб-приложения для работы с ней.

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

					ПГУ 1.090501.04.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программа для интерактивного тестирования IT специалистов Пояснительная записка	Лит.	Лист	Листов
Разраб.		Исаева В.Е.						
Провер.		Слепцов Н.В.		19.03.18			2	135
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.				гр. 18ВОЭ1		

Содержание

Перечень сокращений и обозначений.....	6
Введение.....	7
1 Анализ предметной области.....	8
1.1 Описание предметной области.....	8
1.2 Анализ существующих аналогов.....	11
1.2.1 Социальная платформа « <i>HackerRank</i> ».....	11
1.2.2 Онлайн-платформа « <i>Codility</i> ».....	12
1.2.3 Веб-приложение « <i>CoderPad</i> ».....	13
1.2.4 Онлайн-платформа « <i>CodinGame</i> ».....	14
1.2.5 Платформа « <i>DevSkiller</i> ».....	15
1.3. Цели и задачи разрабатываемой программы.....	16
1.4 Анализ целевой аудитории.....	17
1.5 Проектирование разрабатываемой программы.....	17
2 Техническое задание.....	20
2.1 Назначение разработки.....	20
2.2 Требования к программе.....	20
2.2.1 Требования к функциональным характеристикам.....	20
2.2.2 Требования к надежности.....	20
2.2.3 Требования к параметрам технических средств.....	21
2.3 Требования к программной документации.....	21
2.4 Стадии и этапы разработки.....	21
2.5 Порядок контроля и приемки.....	22
3 Обоснование выбора инструментов разработки.....	24
3.1 Текстовый редактор <i>Visual Studio Code</i>	24
3.2. Язык разработки <i>JavaScript</i>	26
3.3 Библиотека <i>React</i>	28
3.4 Язык разработки <i>HTML5</i>	29
3.5 Язык разработки <i>CSS</i>	30

3.6 Сборщик модулей <i>Webpack</i>	31
3.7 СУБД <i>PostgreSQL</i>	32
3.8 CASE-средство <i>OpenModelSphere 3.2</i>	35
4 Разработка программы.....	37
4.1 Логическое и физическое проектирование базы данных.....	37
4.2 Описание функций программы.....	44
4.3 Описание работы с базой данных.....	59
5 Оценка затрат на разработку и внедрение программы для интерактивного тестирования <i>IT</i> специалистов.....	64
5.1 Организация производства разработки программы для интерактивного тестирования <i>IT</i> специалистов.....	64
5.2 Расчет себестоимости программного обеспечения для интерактивного тестирования <i>IT</i> специалистов.....	66
5.3 Оценка эффективности программы для интерактивного тестирования <i>IT</i> специалистов.....	71
5.4 Анализ рисков при разработке и внедрении программы для интерактивного тестирования <i>IT</i> специалистов.....	71
6 Общие принципы обеспечения безопасности на рабочем месте оператора программного приложения для интерактивного тестирования <i>IT</i> специалистов.....	73
6.1 Актуальность вопросов безопасности на рабочем месте.....	73
6.2 Требования СанПиН при работе с компьютером.....	74
6.3 Требования безопасности при работе с электроприборами.....	75
6.4 Влияние ПЭВМ на физиологию человека.....	76
6.5 Техника безопасности при работе с персональным компьютером.....	77
Заключение.....	79
Список использованных источников.....	80
Приложение А – <i>UML</i> -диаграммы.....	81
Приложение А.1 – Диаграмма вариантов использования.....	82
Приложение А.2 – Диаграмма последовательности.....	83

Приложение А.3 – Диаграмма развертывания.....	84
Приложение А.4 – Логическая модель данных.....	85
Приложение А.5 – Физическая модель данных.....	86
Приложение Б – <i>SQL</i> -скрипт.....	87
Приложение В – Листинг программы.....	90
Приложение Г – Презентация.....	124

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ГОСТ – государственный стандарт

ОС – операционная система

НФ – нормальная форма

ПЭВМ – персональная электронно-вычислительная машина

ЭВМ – электронно-вычислительная машина

CASE (Computer-Aided Software Engineering) - методы и технологии, которые позволяют проектировать различные информационные системы и автоматизировать их создание

DOM (Document Object Model) – объектная модель документа

IT (Information Technology) – информационные технологии

HR (Human Resources) - отдел в организации, который занимается управлением человеческими ресурсами и заботится о персонале компании

FK (Foreign Key) – внешний ключ

PK (Primary Key) первичный ключ

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ 1.090501.04.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящее время сфера информационных технологий стала неотъемлемой частью жизни, и сейчас невозможно представить комфортное существование человека без компьютеров, смартфонов, и других ЭВМ, как это было всего век назад. Теперь же многие функции, выполняемые ранее вручную, перекладываются на «плечи» машин, что позволяет облегчить и усовершенствовать жизнь населения, делая ее более комфортной и простой.

Информационные технологии проникли в каждую сферу жизни, начиная от заказа еды с доставкой на дом и заканчивая банковским и даже военным делами. Бизнес, коммерция, наука и многое другое – все это развивается быстрее и успешнее с применением компьютерных технологий и ЭВМ.

Кроме того, в компьютерах и компьютерных сетях хранится огромное количество данных, что это дает людям возможность получать знания, следить за новостями, обмениваться информацией и оставаться на связи.

Таким образом, можно сделать вывод, что в наши дни внедрение ЭВМ повсеместно и развитие компьютерных технологий – процесс важный, необходимый и необратимый. А для того, чтоб электронный рынок и современный электронный мир не потерпел крах, *IT*-специалисты, занимающиеся этим, должны быть достаточно квалифицированными и образованными.

					ПГУ 1.090501.04.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Описание предметной области

В данный период жизни в России происходит особо активное развитие сферы *IT* - программное обеспечение, ранее почти без ограничений поставляемое другими странами, необходимо заменить российскими аналогами. Для выполнения этой задачи нужны способные разработчики.

Ежегодно Российские университеты выпускают 75 тысяч *IT*-специалистов, из которых значительная часть планирует работать по профессии в различных *IT* компаниях. Однако чаще всего для устройства на работу недостаточно только диплома, поэтому компании используют определенные средства для проверки квалификации их потенциальных сотрудников или сотрудников, уже работающих, но желающих получить более высокую должность.

Одним из способов собеседования может быть тестирование, которое проводится либо вручную, либо с использованием специальной программы. Использование такой специальной программы помогает оптимизировать процедуру найма новых сотрудников, а также отслеживать квалификацию уже существующих работников компании, чтоб определить, какие из сотрудников уже имеют высокий потенциал для продвижения. Программа для тестирования способствует принятию верного решения и помешает допустить ошибки в отборе своих «новобранцев».

Программа для интерактивного тестирования *IT* специалистов – инструмент, позволяющий сотруднику *HR*-отдела *IT*-компании проверить квалификацию *IT*-специалиста. В зависимости от результата тестирования сотрудник *HR*-отдела сможет решить, принимать ли нового специалиста в компанию либо повышать ли должность уже состоящего в компании специалиста.

Хорошим решением будет являться реализация программы для тестирования *IT*-специалистом в виде веб-приложения по нескольким перечисленным ниже причинам.

					ПГУ 1.090501.04.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Веб-приложения могут быть доступны в любое время и из любого места с использованием любого устройства с доступом в Интернет. Это удобно для *IT*-специалистов, которые могут проходить тесты из офиса, дома или даже в пути, используя свой смартфон или ноутбук.

Также веб-приложения могут легко масштабироваться и обрабатывать большое количество одновременных пользователей. Это позволяет эффективно тестировать множество *IT*-специалистов одновременно без затруднений и задержек.

Веб-приложения подстраиваются под современные требования и тенденции в области *IT*, такие как использование облачных технологий, мобильная доступность и интеграция с другими важными инструментами и платформами. Это позволяет обеспечить современный и интуитивно понятный пользовательский интерфейс, а также легко интегрироваться с существующими системами и процессами в организации.

Таким образом, веб-приложения для тестирования *IT*-специалистов обеспечивают удобство, эффективность и точность при проведении тестов, что делает их хорошим выбором для оценки компетенции и навыков *IT*-специалистов.

На рисунке 1 представлена диаграмма вариантов использования. На диаграмме вариантов использования представлены две сущности: пользователь и администратор.

В возможности пользователя входит:

- создание аккаунта (регистрация в системе);
- авторизация;
- выбор целевого теста;
- прохождение теста по целевому направлению.

В возможности администратора входит:

- создание аккаунта (регистрация в системе);
- авторизация;
- просмотр результатов тестирования пользователя;
- создание тестов по требующимся критериям;

					ПГУ 1.090501.04.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

- анализ результатов тестов;
- формирование и отправка пользователю обратной связи по тесту.



Рисунок 1 – Диаграмма вариантов использования

Пользователь на представленной диаграмме — это *IT*-специалист, являющийся либо кандидатом на должность, проходящим собеседование, либо уже работающим в компании сотрудником и потенциальным кандидатом на повышение или понижение в должности.

Администратор — сотрудник *HR*-отдела, задача которого оценить квалификацию кандидата/сотрудника посредством проведения тестирования с использованием разрабатываемой программы для тестирования *IT*-специалистов.

При первом использовании программы сначала необходимо осуществить регистрацию аккаунта под одним из видов пользователей, при следующих входах в систему повторная регистрация не требуется.

Когда пользователь авторизуется и входит в систему, ему предоставляется возможность выбрать категорию теста и его тему (сообщается пользователю администратором), а также пройти тестирование. Результаты пройденного теста для пользователя не доступны.

Когда администратор авторизуется и входит в систему, ему предоставляется возможность создания нового теста. Создание теста включает в себя определение названия теста, составление и добавление вопросов, а также вариантов

ответов на каждый вопрос. Также администратор имеет доступ к результатам пройденных пользователями тестов, на основе которых он делает анализ их квалификации и, по необходимости, формирует и отправляет обратную связь.

1.2 Анализ существующих аналогов

Существует несколько популярных систем для тестирования IT-специалистов. Ниже приведены некоторые из них.

1.2.1 Социальная платформа «HackerRank»

HackerRank это платформа для подготовки и тестирования навыков программирования. Она предоставляет широкий набор задач и заданий на различные языки программирования, а также проводит соревнования среди программистов.

HackerRank также предлагает возможность участвовать в собеседованиях и пройти техническое собеседование с различными компаниями, а также создавать свои задачи для участия в соревнованиях или обучения других людей.

Платформа *HackerRank* широко используется в учебных заведениях, компаниях и программистами во всем мире для обучения, рекрутинга и оценки навыков программирования.

На рисунке 2 представлена главная страница платформы *HackerRank*.

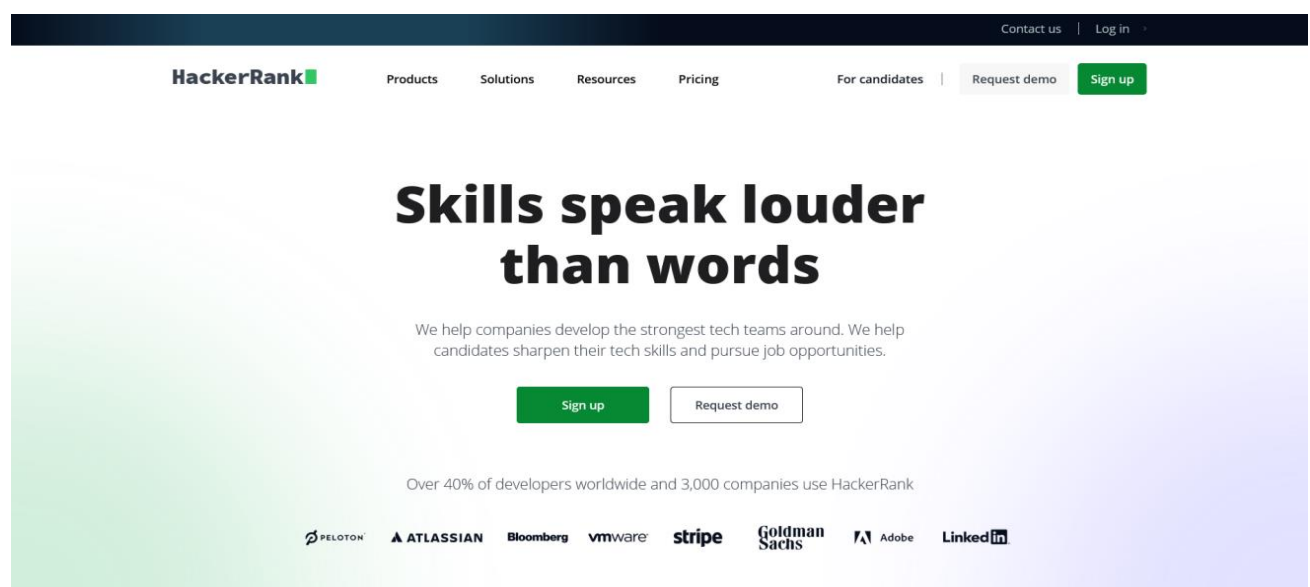


Рисунок 2 – Платформа «HackerRank»

					ПГУ 1.090501.04.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

1.2.2 Онлайн-платформа «Codility»

Codility - это онлайн-платформа для оценки технических навыков программистов. Она предоставляет компаниям возможность проводить технические собеседования и оценку профессиональных знаний кандидатов в области программирования.

Codility предлагает широкий набор задач и тестов, которые охватывают различные области программирования, такие как алгоритмы, структуры данных, языки программирования, базы данных и другие. Каждая задача оценивается на основе правильности решения, эффективности, качества кода и других параметров.

Платформа *Codility* предоставляет компаниям инструменты для создания и настройки тестовых сценариев, отслеживания прогресса кандидатов, а также генерации отчетов и статистики. Кроме того, платформа *Codility* также поддерживает множество популярных языков программирования, таких как *C++*, *Java*, *Python*, *Ruby* и другие, что позволяет кандидатам использовать знакомые им инструменты.

Использование платформы *Codility* помогает компаниям определить технические навыки и способности кандидатов в программисты, что позволяет им принимать квалифицированных профессионалов и повышать эффективность процесса найма.

Главная страница данной платформы представлена на рисунке 3.

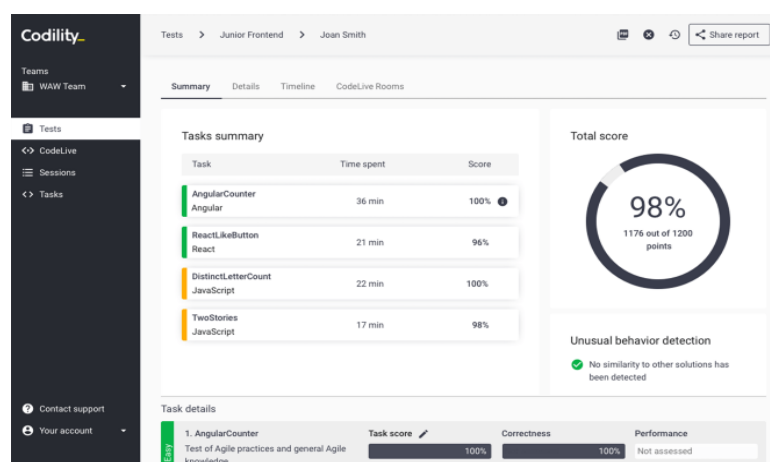


Рисунок 3 – Платформа «Codility»

1.2.3 Веб-приложение «*CoderPad*»

CoderPad - это платформа, которая позволяет разработчикам удаленно сотрудничать и проводить технические собеседования. Платформа обеспечивает среду совместного кодирования в режиме реального времени, в которой несколько пользователей могут вместе писать и редактировать код.

С помощью *CoderPad* интервьюеры могут создавать вопросы по программированию или упражнения по решению проблем, а также делиться уникальной ссылкой с кандидатами для проведения собеседований по программированию. И интервьюер, и кандидат могут видеть и изменять код в режиме реального времени, что упрощает совместную работу и совместное решение проблем.

Помимо собеседований по программированию, *CoderPad* также предлагает дополнительные функции, такие как использование доски, совместное использование экрана и видеоконференции. Это обеспечивает более интерактивное собеседование, где интервьюеры и кандидаты могут обсуждать и решать проблемы в режиме реального времени.

Итак, можно сказать, что *CoderPad* предназначен для оптимизации и улучшения процесса собеседования по программированию, позволяя интервьюерам и кандидатам эффективно сотрудничать, даже если они находятся в разных физических местах.

Главная страница данной платформы представлена на рисунке 4.

					ПГУ 1.090501.04.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

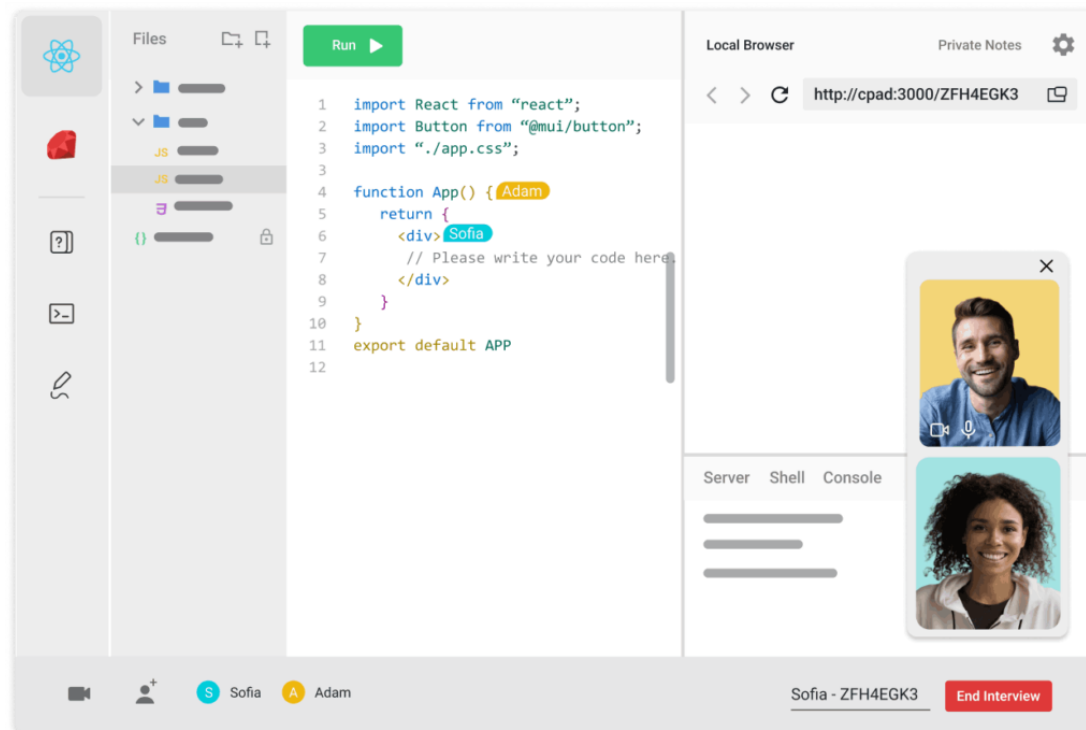


Рисунок 4 – Собеседование кандидатов с платформой «*CoderPad*»

1.2.4 Онлайн-платформа «*CodinGame*»

CodinGame - это онлайн-платформа, созданная для развития навыков программирования через решение интерактивных головоломок и создание игр.

На платформе представлены различные языки программирования, включая *Python*, *JavaScript*, *Java*, *C++*, *C#* и многие другие. Пользователи могут выбирать задачи, которые им интересны, и решать их, используя любой из доступных языков программирования.

Одним из основных преимуществ *CodinGame* является то, что пользователи могут оттачивать свои навыки программирования, работая над реальными проектами. Платформа предлагает различные задачи и головоломки разного уровня сложности, чтобы пользователи могли прокачать свои навыки от начинающего до продвинутого уровня.

CodinGame также предлагает возможность участия в соревнованиях и битвах с другими программирующими участниками. Пользователи могут соревноваться в различных дисциплинах, таких как разработка искусственного интеллекта, программирование для игр и других задачах.

					ПГУ 1.090501.04.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

Платформа предлагает различные инструменты для обучения программированию, включая обучающие видео, учебники и подсказки по решению задач. Благодаря этому, даже начинающие программисты могут улучшить свои навыки и освоить новые языки программирования.

CodinGame также позволяет разработчикам создавать свои собственные задачи и игры, которые другие пользователи могут решать. Это отличная возможность для разработчиков получить обратную связь и поделиться своими проектами с сообществом.

В целом, *CodinGame* - это платформа, которая помогает учиться программированию через решение задач и игр, участие в соревнованиях и разработку проектов. Это отличный инструмент для программистов всех уровней, от начинающих до опытных разработчиков.

Главная страница платформы представлена на рисунке 5.



Рисунок 5 – Платформа «*CodinGame*»

1.2.5 Платформа «*DevSkiller*»

DevSkiller - это платформа для привлечения, оценки и найма ИТ-специалистов. Эта платформа предлагает ряд инструментов и функций для создания и проведения технических интервью и тестов по программированию.

Платформа *DevSkiller* позволяет работодателям создавать свои собственные тесты и задания, основанные на реальных проектах и выполнении реальных

задач, чтобы оценить навыки кандидатов. Кроме того, платформа предлагает готовые наборы задач и тестов, которые можно использовать в процессе найма.

Пользователи могут создавать тесты на различные языки программирования, а также проводить тесты на различные аспекты, такие как алгоритмы, базы данных, фронтенд и многое другое. Тесты могут быть на разных уровнях сложности - от начального до продвинутого.

DevSkiller также предлагает аналитику и отчеты, позволяющие работодателям оценивать навыки кандидатов и принимать обоснованные решения при найме. Все результаты тестирования и интервью хранятся в облаке, что позволяет работодателям легко просматривать и анализировать данные.

DevSkiller - это удобная и эффективная платформа для оценки и найма технических специалистов, которая помогает сократить время и усилия в процессе найма, а также повысить качество набора персонала.

Функциональные возможности представленных аналогов не удовлетворяют потребностям заказчика. Заказчику необходимо программное обеспечение, которое обладает следующими свойствами:

- открытый исходный код;
- возможность формирования и загрузки собственных тестов;
- простота технической поддержки;
- реализация на бесплатном и свободно распространяемом инструментарии.

Программное обеспечение, реализованное в дипломном проекте, полностью удовлетворяет описанным выше требованиям.

1.3 Цели и задачи разрабатываемой программы

Цель – проведение объектного тестирования кандидатов на должность *IT-специалистов*.

Задачами данного проекта являются:

- регистрация и авторизация пользователей в системе;
- возможность формирования и загрузка тестов;
- проведение пользователям тестирования;

					ПГУ 1.090501.04.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

- фиксация результатов проведения тестирования;
- возможность просмотра результатов тестирования;
- формирование обратной связи для тестируемого.

Цели реализации программы тестирования *IT*-специалистов – увеличение скорости отбора потенциальных кандидатов на работу в *IT*-компании. В результате внедрение по планируется достичь следующее:

- повысить скорость отбора потенциальных кандидатов для работы в *IT*-компании;
- улучшить и облегчить функционирование отдела кадров;
- получить список потенциальных кандидатов на различные *IT*-позиции в организации.

1.4 Анализ целевой аудитории

Целевая аудитория программного обеспечения – является потенциальные кандидаты на позицию *IT*-специалиста в профильную организацию. Разработка программного приложения актуальна как для частных, так и для государственных компания, у которых есть штат специалистов по информационным технологиям.

1.5 Проектирование разрабатываемой программы

Проектирование программы выполнено с помощью *UML*-диаграмм. *UML* (*Unified Modeling Language*) - это язык графического описания и моделирования программных систем, который используется для анализа, проектирования и описания различных видов систем.

UML диаграммы позволяют визуализировать и представить различные аспекты системы, ее структуру и взаимодействие между компонентами. Существует несколько типов *UML* диаграмм, каждая из которых предназначена для моделирования определенных аспектов системы.

Например, классовые диаграммы позволяют описать классы и их

					ПГУ 1.090501.04.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

взаимодействие, диаграммы последовательности показывают последовательность выполнения операций, диаграммы состояний описывают различные состояния объекта и переходы между ними, а диаграммы компонентов и пакетов позволяют показать структуру системы и зависимости между компонентами.

UML диаграммы могут быть полезными для коммуникации и запоминания деталей системы, понимания взаимосвязей между компонентами, а также для облегчения процесса разработки программного обеспечения. Они широко используются в инженерии программного обеспечения и системном анализе.

На рисунке 6 представлена диаграмма последовательности функционирования программы при входе в систему от имени обычного пользователя (кандидата).

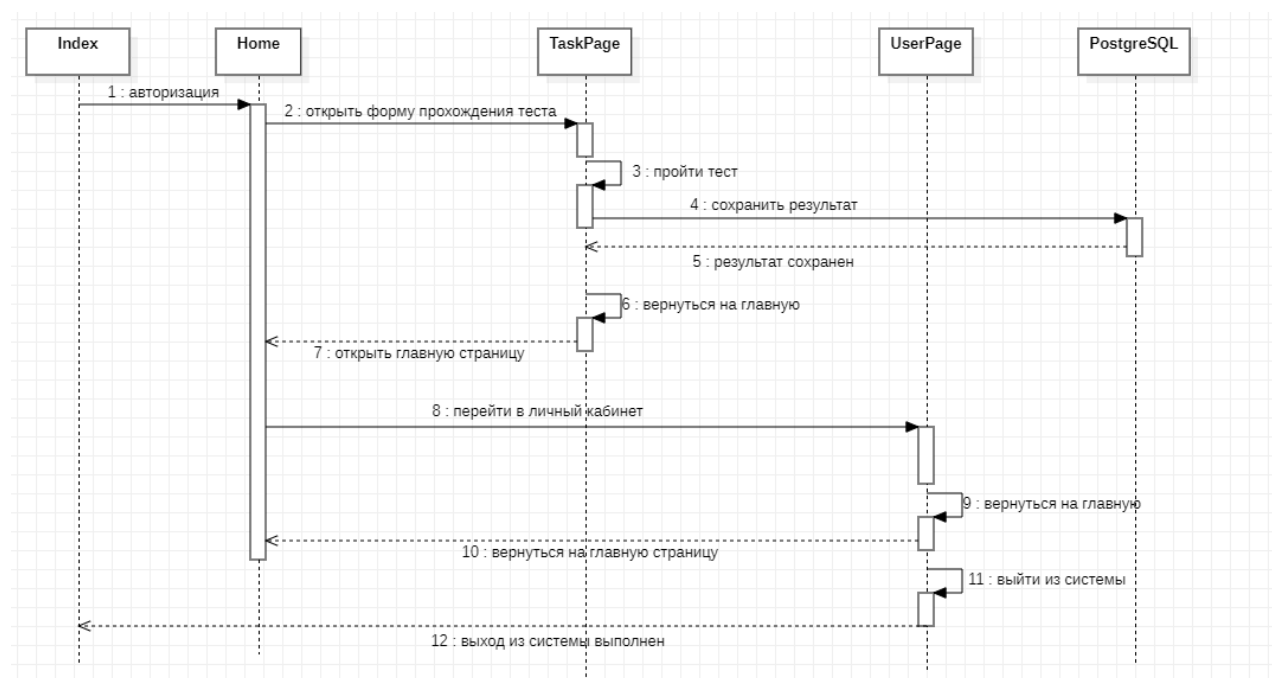


Рисунок 6 – Диаграмма последовательности функционирования программы при входе в систему от имени обычного пользователя

На представленной диаграмме последовательности показаны следующие шаги:

- авторизация;
- открытие формы прохождения теста;

- прохождение тестирования;
- переход в личный кабинет;
- выход из системы.

На рисунке 7 представлена диаграмма последовательности функционирования программы при входе в систему от имени администратора (сотрудника *HR*-отдела).

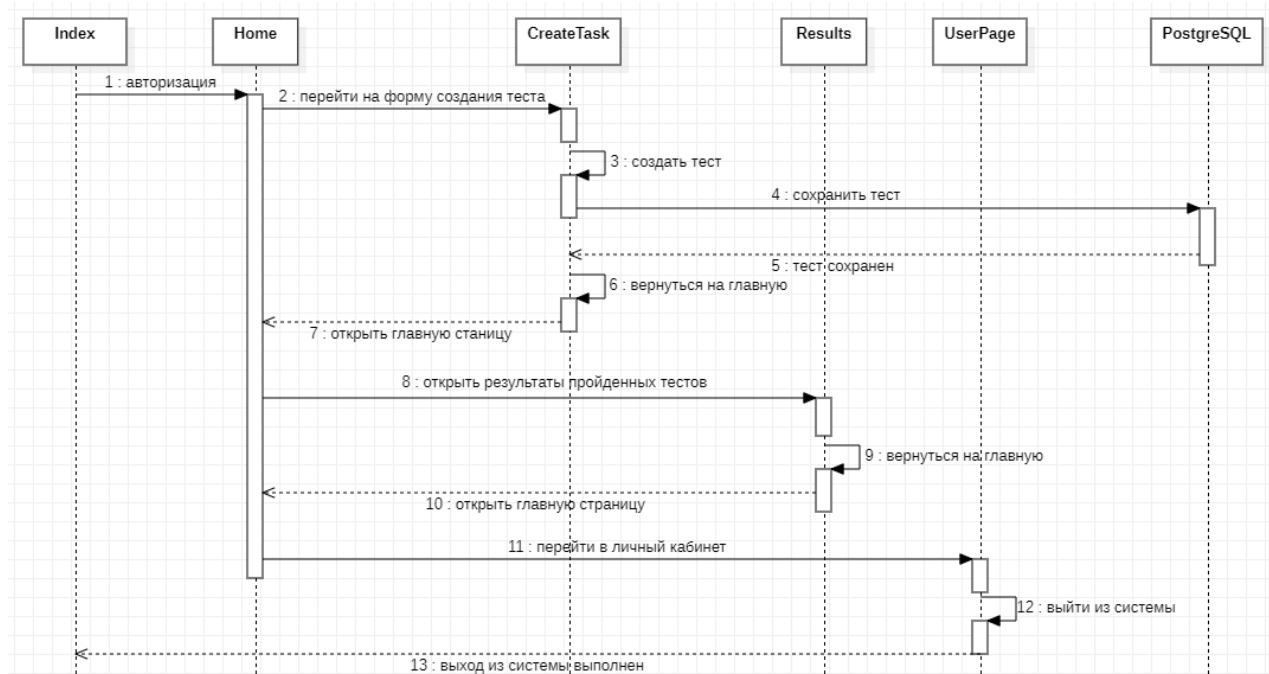


Рисунок 7 - Диаграмма последовательности функционирования программы при входе в систему от имени обычного пользователя

На представленной диаграмме последовательности показаны следующие шаги:

- авторизация;
- открытие формы создания теста;
- создание теста;
- открытие формы с результатами пройденных тестов;
- просмотр результатов;
- переход в личный кабинет;
- выход из системы.

2 Техническое задание

2.1 Назначение разработки

Программа для тестирования *IT*-специалистов должно быть предназначено для автоматизации и упрощения задач сотрудников *HR*-отдела по найму специалистов и контроля квалификации сотрудников.

Цель данного проекта – разработка базы данных и программного веб-приложения для тестирования *IT*-специалистов на предприятии.

2.2 Требования к программе

2.2.1 Требования к функциональным характеристикам

Программа для тестирования *IT*-специалистов должна обеспечивать выполнение следующих функций:

Программа для интерактивного должна обеспечивать следующие функции:

- регистрация пользователей;
- выбор категории (направления) тестирования;
- прохождения и сохранение выбранного теста;
- регистрация администратора;
- создание новых категорий и новых тестов для пользователей;
- просмотр результатов тестирования;
- возможности редактирования и удаления существующих категорий и тестов.

2.2.2 Требования к надежности

Программа должна работать без прерываний и ошибок. В случае, если пользователь совершает неверные действия, система должна сообщить ему об этом.

Программа должна обеспечивать контроль вводимых данных. Если были введены некорректные данные или данные, являющиеся обязательными, не были введены, система должна информировать об этом пользователя.

					ПГУ 1.090501.04.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

При вводе входных данных должен осуществляться ее контроль на допустимость вводимых значений по следующим правилам:

- вводимые данные не должны противоречить заявленному типу данных;
- вводимые данные не должны противоречить законам логики.

Конечный пользователь не должен иметь доступ к исходным данным приложения. Для модификации программной части системы пользователю необходимо обратиться к разработчику данной программы.

2.2.3 Требования к параметрам технических средств

Разрабатываемое приложение предназначено для работы на персональных компьютерах специалистов по тестированию, с минимальная конфигурация технических средств:

- ОС *Windows* 10 и выше;
- процессор *AMD* 5 или *INTEL* 11 поколения и выше;
- свободное место на жестком диске 1 ГБ и выше;
- оперативную память 4 ГБ и выше;
- монитор с диагональю не менее 27 дюймов;
- периферия: мышь, клавиатура.

2.3 Требования к программной документации

Программа для тестирования ИТ-специалистов должна сопровождаться документацией, содержащей:

- техническое задание;
- описание проектирования БД;
- описание программы и методика испытаний;
- описание применения.

2.4 Стадии и этапы разработки

Стадии и этапы разработки программы для тестирования ИТ-специалистов представлены в таблице 1.

					ПГУ 1.090501.04.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1 – Стадии и этапы разработки

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
1 Анализ предметной области	Анализ предметной области	10		Исаева В.Е.
2 Проектирование программы	- концептуальное проектирование БД - логическое проектирование БД - физическое проектирование БД	20		Исаева В.Е.
3 Разработка базы данных	Разработка базы данных	10		Исаева В.Е.
4 Отладка и тестирование	Отладка и тестирование	10		Исаева В.Е.
5 Разработка экономической части	Оценка затрат на разработку и внедрение программы для тестирования ИТ-специалистов	10		Исаева В.Е.
6 Разработка раздела БЖД	Оценка тяжести труда эксплуататора программы	5		Исаева В.Е.
7 Оформление графической части	Оформление графической части	10		Исаева В.Е.
8 Оформление выпускной работы	Оформление пояснительной записки	30		Исаева В.Е.

2.5 Порядок контроля и приемки

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

					ПГУ 1.090501.04.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

3 Обоснование выбора инструментов разработки

На основании анализа требований к функциональным характеристикам программы для тестирования ИТ-специалистов были выбраны наиболее актуальные средства разработки.

3.1 Текстовый редактор *Visual Studio Code*

В качестве редактора исходного кода для разработки настоящей программы используется среда *Visual Studio Code*. Это бесплатный, «легкий» и расширяемый редактор кода для создания веб-приложений, а также настольных и мобильных приложений с использованием любого языка программирования. Изначально данный редактор кода поддерживает *JavaScript* и *TypeScript*, предлагая расширения для кодирования на таких языках, как *Python*, *Java*, *C/C++*, *C#*, *Go*, *Rust*, *PHP* и множество других.

Visual Studio Code имеет встроенную поддержку управления исходных кодом *Git* (распределенная система управления версиями) и мощную интеграцию с *GitHub* [1] (самый крупный веб-сервис для хостинга ИТ-проектов и их совместной разработки), встроенным отладчиком (компьютерная программа для автоматизации процесса отладки: поиска ошибок в других программах, ядрах операционных систем, *SQL*-запросах и других видах кода), а также интеллектуальное завершение кода с помощью *IntelliSense* и *IntelliCode* на основе искусственного интеллекта. *IntelliSense* и *IntelliCode* представляют собой технологию автодополнения *Microsoft* и набор возможностей разработки с помощью ИИ для повышения производительности разработчиков нового поколения соответственно.

В таблице 2 представлено сравнение функционала редакторов *Visual Studio Code* и *Eclipse*.

					ПГУ 1.090501.04.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 2 - Сравнение функционала редакторов *Visual Studio Code* и *Eclipse*.

Характеристика	<i>Visual Studio Code</i>	<i>Eclipse</i>	Пояснение
Легкость использования	+	-	<i>Visual Studio Code (VS Code)</i> обычно считается более простым в использовании, поскольку он имеет более простой пользовательский интерфейс и ориентирован на разработчиков, работающих с языками программирования, такими как <i>JavaScript</i> и <i>TypeScript</i> . <i>Eclipse</i> , с другой стороны, может быть более сложным в освоении для новичков.
Расширяемость	+	+	Обе <i>IDE</i> являются очень расширяемыми благодаря наличию плагинов. Однако у <i>VS Code</i> более активное сообщество разработчиков, что означает большой выбор плагинов и экосистема сторонних инструментов.
Обширный спектр поддерживаемых языков программирования	-	+	<i>Eclipse</i> обладает более широким спектром поддерживаемых языков программирования, включая <i>Java</i> , <i>C++</i> , <i>Python</i> , <i>PHP</i> и т. д. <i>VS Code</i> также имеет впечатляющую поддержку различных языков, но начинал свою историю как текстовый редактор для веб-разработки и <i>JavaScript</i> .
Размер и производительность	+	-	<i>Eclipse</i> является более тяжеловесным Интерфейсом, который может потреблять больше ресурсов компьютера, особенно для крупных проектов. <i>VS Code</i> , являясь более легким, может работать более отзывчиво, особенно при работе с небольшими или средними проектами.

Характеристика	<i>Visual Studio Code</i>	<i>Eclipse</i>	Пояснение
Сообщество и поддержка	+	-	Оба <i>IDE</i> имеют большое сообщество пользователей, но сообщество <i>VS Code</i> , поскольку это более новый инструмент, обычно является более активным и предлагает более частые обновления и исправления ошибок.

3.2 Язык разработки *JavaScript*

Основным языком разработки в данном проекте выбран язык разработки *JavaScript*.

JavaScript - это высокоуровневый язык программирования, который широко используется для разработки веб-приложений. Он является одним из основных технологических компонентов для создания интерактивных элементов на веб-страницах.

JavaScript обеспечивает функциональность, позволяющую изменять содержимое и стиль веб-страницы, обрабатывать события, валидировать формы, делать запросы к серверам и многое другое. Он позволяет создавать динамический и интерактивный пользовательский интерфейс, управлять поведением элементов и добавлять анимацию.

Основные характеристики языка *JavaScript* включают:

- легкость обучения: *JavaScript* имеет простой и интуитивно понятный синтаксис, который понятен даже новичкам в программировании;
- интерпретируемость: *JavaScript* выполняется непосредственно в браузере без необходимости компиляции, что ускоряет процесс разработки;
- объектно-ориентированность: *JavaScript* поддерживает объектно-ориентированную парадигму, позволяя создавать классы, объекты и использовать наследование;

- возможность манипулирования *DOM*: *JavaScript* позволяет легко взаимодействовать с деревом объектов документа (*DOM*) и изменять его структуру и содержимое;

- масштабируемость: *JavaScript* является мощным языком программирования, который может быть использован для создания сложных и масштабируемых приложений;

- поддержка стандартами: *JavaScript* активно развивается и соответствует стандартам *ECMA International*, что обеспечивает совместимость с различными браузерами.

JavaScript является одним из наиболее популярных языков программирования, используемых для разработки веб-приложений, и продолжает быть востребованным в сфере веб-разработки.

Ниже представлено сравнение языка *JavaScript* с языками *PHP* и *Python*.

JavaScript - это язык программирования, который распространен в веб-разработке, поскольку выполняется непосредственно в браузере. Он может использоваться для создания интерактивных веб-страниц с динамическим содержанием и веб-приложений, а также имеет широкую поддержку и множество библиотек и фреймворков, которые облегчают разработку. *JavaScript* также используется для разработки серверной части с использованием платформы *Node.js*.

PHP - это язык программирования, который широко применяется в веб-разработке для создания динамических веб-страниц и серверных приложений. Он интегрируется с *HTML* и может использоваться для обработки данных форм, работы с базами данных и создания динамического контента. *PHP* широко используется в системе управления контентом, таких как *WordPress*, *Joomla* и *Drupal*. *PHP* также обладает большой базой разработчиков и обширной документацией, что делает его популярным выбором для создания веб-приложений.

Python - это язык программирования общего назначения, который может быть использован в различных областях, включая веб-разработку. *Python* обладает простым и читаемым синтаксисом, что делает его очень популярным

					ПГУ 1.090501.04.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

среди начинающих программистов. Он имеет мощные и богатые библиотеки, такие как *Django* и *Flask*, которые обеспечивают обширные возможности веб-разработки и создания веб-фреймворков. *Python* также широко используется в научных исследованиях, анализе данных и искусственном интеллекте благодаря своим мощным библиотекам, таким как *NumPy*, *Pandas* и *TensorFlow*.

Исходя из вышеперечисленного, можно сделать вывод, что язык разработки *JavaScript* наиболее подходящий для использования в процессе разработки заданной программы.

3.3 Библиотека *React*

React - это *JavaScript* библиотека, разработанная компанией *Facebook* для создания пользовательских интерфейсов. Она позволяет разработчикам строить компоненты интерфейса, которые обрабатывают изменения данных и обновляют только ту часть интерфейса, которая действительно изменилась. В основе *React* лежит концепция виртуального *DOM* (*Virtual DOM*), которая позволяет увеличить производительность и эффективность обновлений интерфейса. Вместо того, чтобы напрямую обновлять *HTML*-элементы при каждом изменении данных, *React* строит виртуальное представление интерфейса, которое сравнивает с текущим состоянием и обновляет только изменившиеся части дерева *DOM*.

React также предоставляет множество функциональных возможностей, таких как жизненный цикл компонентов, система условного рендеринга, обработка событий и многое другое. Компоненты *React* могут быть многоразовыми модулями, которые могут быть использованы повторно на разных страницах и приложениях.

Данная библиотека является мощным инструментом для разработки пользовательских интерфейсов, который позволяет создавать масштабируемые и производительные веб-приложения. Он пользуется широкой популярностью и поддержкой сообщества разработчиков.

В таблице 3 представлен сравнительный анализ библиотек *React* и *Vue*.

					ПГУ 1.090501.04.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 3 - Сравнительный анализ библиотек *React* и *Vue*

Характеристика	<i>React</i>	<i>Vue</i>
Синтаксис и обучаемость	<i>JSX</i> – комбинация <i>JavaScript</i> и <i>XML</i> , требует изучения особого синтаксиса <i>JSX</i> .	Использует привычный <i>HTML</i> -подобный шаблонный синтаксис, легче для понимания и изучения.
Компонентный подход	Использует компонентный подход, разделение интерфейса на небольшие, переиспользуемые компоненты.	Также использует компонентный подход с возможностью создавать высокоуровневые компоненты с помощью <i>Single File Components (SFC)</i> .
Экосистема и сообщество	Имеет очень большую и активную экосистему, которая включает в себя множество сторонних библиотек, фреймворков и инструментов.	Хотя экосистема <i>Vue</i> не так обширна, как у <i>React</i> , она все равно предлагает множество ресурсов и плагинов для удобной разработки.
Размер бандла и производительность	Генерирует более объемный <i>JavaScript</i> -бандл, что может влиять на начальную загрузку и производительность.	Использует виртуальный <i>DOM</i> и умные обновления, что может привести к более эффективной работе с <i>DOM</i> и лучшей производительности.
Интеграция с другими инструментами	Легко интегрируется с другими сторонними инструментами и фреймворками, такими как <i>Redux</i> или <i>Next.js</i> .	Также обладает хорошей интеграцией, однако имеет более сглаженную кривую обучения и удобное интегрированное решение для управления состоянием - <i>VueX</i> .

3.4 Язык разработки *HTML5*

HTML5 (*Hypertext Markup Language 5*) является последней версией стандарта языка разметки гипертекста. Это основной язык для создания и оформления веб-страниц [2].

Основные характеристики *HTML5* имеет пять основных преимуществ:

- семантическая разметка. *HTML5* включает новые элементы разметки, которые позволяют лучше структурировать содержимое веб-страницы. Новые элементы, такие как `<header>`, `<nav>`, `<section>`, `<article>` и `<footer>`, позволяют создавать более читабельный и логический код;

- расширенная графика и мультимедиа. *HTML5* включает новые элементы и *API* для добавления графики и мультимедиа на веб-страницы без необходимости использования плагинов. Это позволяет использовать элементы `<canvas>` для создания интерактивных графических элементов, а также тег `<video>` и `<audio>` для вставки видео и звука на страницу;

- улучшенные формы. *HTML5* предоставляет новые типы полей для форм, такие как поле для ввода *email*, номер телефона или даты. Кроме того, *HTML5* включает новые атрибуты и *API* для проверки форм, автозаполнения и отправки данных на сервер без обновления страницы;

- локальное хранилище. *HTML5* включает возможности локального хранения данных на стороне клиента. Это позволяет веб-приложениям сохранять данные на компьютере пользователя и обрабатывать их без постоянного соединения с интернетом;

- улучшенная поддержка для мобильных устройств. *HTML5* включает множество функций, которые упрощают создание адаптивных и мобильных веб-приложений. Такие функции, как геолокация, доступ к камере и микрофону, позволяют создавать более интерактивные и функциональные приложения для мобильных устройств.

HTML5 является открытым стандартом и поддерживается всеми современными браузерами. Он предоставляет разработчикам широкий набор возможностей для создания современных и интерактивных веб-страниц и веб-приложений.

3.5 Язык разработки *CSS*

CSS (Cascading Style Sheets) – это язык, используемый для описания внешнего вида элементов на веб-странице [3]. Он определяет, как элементы *HTML* должны быть отображены на экране, на печати или других устройствах.

					ПГУ 1.090501.04.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

Ниже перечислены основные характеристики CSS:

- разделение содержания и представления. CSS позволяет разделить структуру *HTML*-документа от его визуального представления. Это позволяет легко изменять внешний вид без необходимости изменения *HTML*-кода;

- каскадность. Название CSS отражает его основную концепцию – каскадирование. CSS-стили могут быть определены в различных файлах и подключены к одному и тому же *HTML*-документу. При этом стили могут наследоваться, объединяться и перезаписываться другими стилями в зависимости от их специфичности и порядка объявления;

- наследование. CSS может наследовать стили от родительских элементов. Например, если вы задаете цвет текста для элемента *<body>*, все дочерние элементы будут иметь тот же цвет текста, если явно не задано другое значение;

- поддержка разных медиа-устройств. CSS позволяет эффективно адаптировать внешний вид веб-страницы для разных устройств и медиа-типов, таких как экраны компьютеров, печать, мобильные устройства и т.д. Это достигается путем определения различных стилей для каждого медиа-типа;

- модульность. CSS состоит из различных модулей, каждый из которых определяет определенный аспект визуального представления. Это позволяет разработчикам использовать только нужные модули и упрощает поддержку и обновление стилей.

CSS обеспечивает гибкость и контроль над внешним видом элементов на веб-странице, что делает его важной технологией для веб-разработки.

3.6 Сборщик модулей *Webpack*

Сборщик модулей *Webpack* - это инструмент, который позволяет объединять различные модули (файлы) веб-приложения в один или несколько бандлов (объединенных файлов). Он является одним из наиболее популярных инструментов для сборки фронтенд-приложений.

Ниже перечислены основные характеристики сборщика модулей *Webpack*:

					ПГУ 1.090501.04.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

- модульность. *Webpack* позволяет организовать код вашего приложения в виде небольших модулей, каждый из которых содержит свою функциональность. Это позволяет легко поддерживать и разрабатывать большие проекты;

- зависимости. *Webpack* автоматически анализирует зависимости между модулями и создает граф зависимостей. Это позволяет ему оптимизировать сборку, удалять неиспользуемый код и обрабатывать только те модули, которые действительно нужны для работы приложения;

- лоадеры. *Webpack* позволяет использовать различные лоадеры, которые расширяют его функциональность. Лоадеры позволяют обрабатывать различные типы файлов (например, *JavaScript*, *CSS*, изображения) и применять к ним различные преобразования (например, компиляцию, минификацию, оптимизацию и т. д.). Это делает *Webpack* очень гибким инструментом;

- плагины. *Webpack* также поддерживает использование плагинов, которые предоставляют дополнительные функциональные возможности. Плагины позволяют автоматизировать задачи, такие как генерация *HTML*-страниц, создание кэшей или подключение сторонних библиотек. Они помогают упростить и автоматизировать процесс разработки;

- конфигурация. *Webpack* основывается на конфигурационном файле, в котором задаются параметры сборки. Конфигурация позволяет настроить различные аспекты сборки, такие как точка входа, выходной файл, настройки лоадеров и плагинов и так далее.

Webpack предоставляет мощные возможности для сборки, оптимизации и упрощения разработки веб-приложений. Он позволяет собирать и объединять модули, обрабатывать различные типы файлов, оптимизировать код и автоматизировать задачи. Это делает *Webpack* одним из наиболее популярных инструментов для сборки фронтенд-приложений.

3.7 СУБД *PostgreSQL*

В данном проекте в качестве СУБД используется СУБД *PostgreSQL*.

PostgreSQL - это объектно-реляционная система управления базами

					ПГУ 1.090501.04.001 ПЗ	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

данных (СУБД), которая является одной из самых мощных и популярных СУБД в мире. Она основана на языке *SQL (Structured Query Language)* и поддерживает полный набор возможностей, необходимых для разработки и управления большими и сложными базами данных.

PostgreSQL предоставляет множество возможностей для разработки дополнительных функций и типов данных, что позволяет адаптироваться к различным потребностям и задачам.

Данная СУБД поддерживает параллельное выполнение запросов, репликацию данных и разделение данных по нескольким серверам, что позволяет обрабатывать большие объемы данных и высокие нагрузки.

PostgreSQL обладает высокой степенью надежности и оптимальной работой даже в случае сбоев системы. Она поддерживает *ACID (atomicity, consistency, isolation, durability)* - набор свойств, обеспечивающих целостность данных.

СУБД *PostgreSQL* поддерживает различные типы данных (текстовые, числовые, даты и времени, географические и другие), а также множество операторов и функций для обработки и анализа данных.

PostgreSQL предоставляет множество возможностей для обеспечения безопасности данных, включая авторизацию, аутентификацию, шифрование и журналирование. Также *PostgreSQL* имеет активное сообщество разработчиков и пользователей, которое обеспечивает постоянное развитие и поддержку СУБД. Существует множество ресурсов, документации и форумов, где можно получить помощь и поддержку.

СУБД *PostgreSQL* является мощным и надежным инструментом для работы с базами данных и находит применение в различных областях, включая разработку приложений, аналитику данных, корпоративные системы и многое другое.

В таблице 4 представлен сравнительный анализ СУБД *PostgreSQL*, *MySQL* и *Microsoft SQL*.

Таблица 4 - Сравнительный анализ СУБД *PostgreSQL*, *MySQL* и *Microsoft SQL*

Характеристика	<i>PostgreSQL</i>	<i>MySQL</i>	<i>Microsoft SQL</i>	Пояснение
Открытый исходный код	+	+	-	<i>PostgreSQL</i> и <i>MySQL</i> являются СУБД с открытым исходным кодом, что означает, что они бесплатны для использования, распространения и модификации. <i>Microsoft SQL</i> , с другой стороны, является коммерческим продуктом и требует оплаты для его использования.
Совместимость	<i>SQL ANSI</i> , обработка транзакций, поддержку процедур, функций и триггеров, множественные типы данных.		<i>SQL ANSI</i> , собственные уникальные функции и расширения	<i>PostgreSQL</i> и <i>MySQL</i> поддерживают стандарты <i>SQL ANSI</i> . Они имеют широкие возможности для работы с базами данных. <i>Microsoft SQL</i> также поддерживает стандарты <i>SQL ANSI</i> , но также имеет свои собственные уникальные функции и расширения.
Производительность и масштабируемость	Небольшие и средние проекты		Крупные проекты	<i>PostgreSQL</i> и <i>MySQL</i> оба обеспечивают хорошую производительность и масштабируемость для небольших и средних проектов. <i>Microsoft SQL</i> имеет более высокую производительность и масштабируемость, и поэтому преимущественно используется для крупных предприятий с высокими нагрузками на базу данных.

Характеристика	<i>PostgreSQL</i>	<i>MySQL</i>	<i>Microsoft SQL</i>	Пояснение
Системы операционных баз данных	<i>Windows, Linux, macOS</i> и другие	Различные ОС с некоторыми ограничениями	<i>Windows</i>	<i>PostgreSQL</i> может работать на разных операционных системах, включая <i>Windows, Linux, macOS</i> и другие. <i>MySQL</i> также может работать на разных ОС, но его состояние развития и функциональность на разных платформах могут отличаться. <i>Microsoft SQL</i> полностью интегрирован и оптимизирован для <i>Windows</i> , но также имеет ограниченную поддержку для <i>Linux</i> .
Сообщество и поддержка	+	+	-	<i>PostgreSQL</i> и <i>MySQL</i> имеют активные сообщества разработчиков и пользователей, что обеспечивает широкую поддержку и доступ к документации, форумам и ресурсам. <i>Microsoft SQL</i> имеет ограниченное сообщество сравнительно с открытыми СУБД, но предоставляет высококачественную платную поддержку и документацию.

3.8 CASE-средство *OpenModelSphere 3.2*

Для разработки базы данных используются специальные CASE-средства, позволяющие разрабатывать концептуальные, логические и реляционные модели.

OpenModelSphere (OMS) - это инструмент с открытым исходным кодом для создания и управления моделями данных. *OMS* предоставляет возможность создавать графические модели и метамодели, а также генерировать код на различных языках программирования [4].

Ниже перечислены преимущества *OpenModelSphere*:

- открытый исходный код. *OMS* разработан на основе открытых стандартов и имеет активное сообщество разработчиков. Все исходные коды доступны для свободного использования и изменения, что способствует развитию данного инструмента;

- гибкость и удобство использования. *OpenModelSphere* предоставляет простой и понятный пользовательский интерфейс, который позволяет легко создавать и редактировать модели данных. Он также поддерживает различные типы моделей и метамodelей, что делает его гибким инструментом для работы с различными предметными областями;

- генерация кода. *OMS* позволяет генерировать код на различных языках программирования, что позволяет автоматизировать процесс разработки. Это может значительно сократить время и усилия, затраченные на создание программного обеспечения.

Недостатки *OpenModelSphere* следующие:

- ограниченные возможности. *OpenModelSphere* не имеет такого большого набора функций и возможностей, как коммерческие инструменты для моделирования данных. Это может быть ограничивающим фактором для некоторых проектов с более сложными требованиями;

- отсутствие коммерческой поддержки и ограниченная адаптация.

4 Разработка программы

Программа для тестирования *IT*-специалистов состоит из трех частей: серверная, клиентская и схема базы данных. Диаграмма развертывания представлена на рисунке 8.

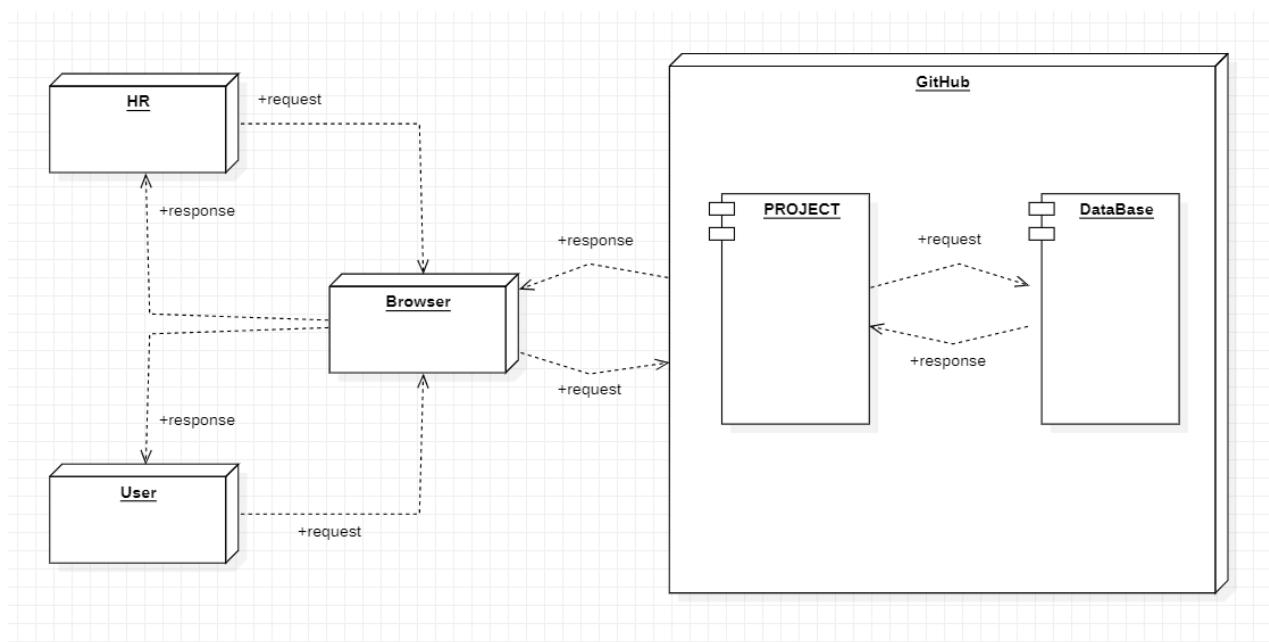


Рисунок 8 – Диаграмма развертывания

4.1 Логическое и физическое проектирование базы данных

Реализованная база данных, приведенная на рисунке 8 является реляционной. Реляционная БД позволяет: избежать избыточных, дублирующихся и недопустимых данных, структуру базы данных легко изменять и поддерживать, есть возможность масштабировать базу данных в целом и таблицы в частности.

Логическая модель базы данных приведена на рисунке 9 и содержит шесть сущностей.

Описание атрибутов сущностей приведено в таблицах 5 – 10.

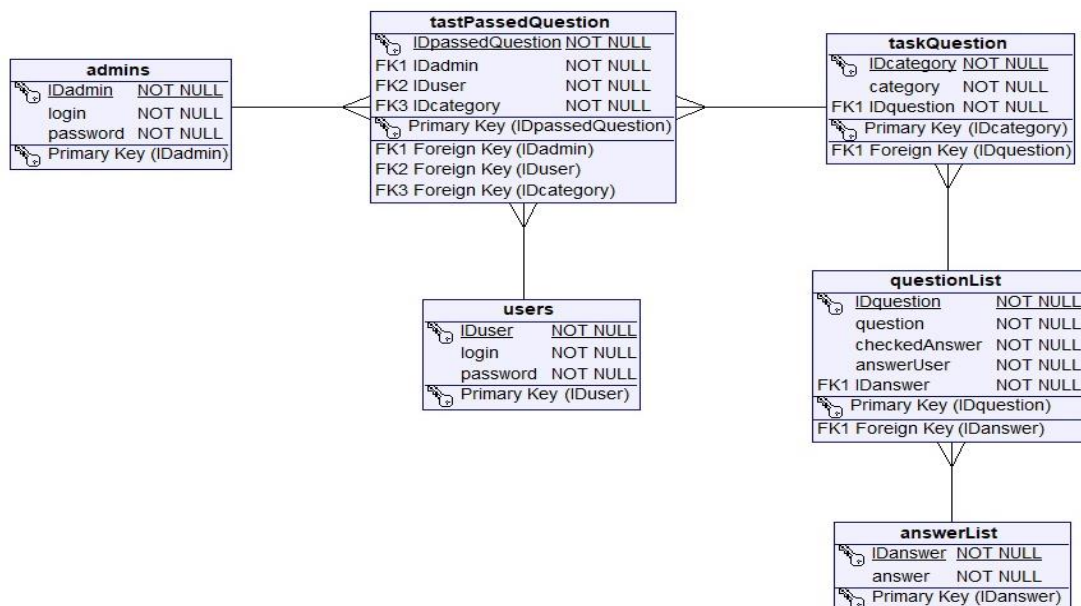


Рисунок 9 – Логическая модель данных

Таблица 5 – Описание атрибутов сущности «*admins*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDadmin</i>	ID администратора	<i>PK</i>	
<i>login</i>	Логин администратора		
<i>password</i>	Пароль администратора		

Таблица 6 – Описание атрибутов сущности «*users*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDuser</i>	ID пользователя	<i>PK</i>	
<i>login</i>	Логин пользователя		
<i>password</i>	Пароль пользователя		

Таблица 7 – Описание атрибутов сущности «*taskPassedQuestion*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDpassedQuestion</i>	ID вопроса	<i>PK</i>	
<i>admins IDadmin</i>	ID администратора		<i>FK1</i>
<i>users IDuser</i>	ID пользователя		<i>FK2</i>
<i>taskQuestion IDcategory</i>	ID категории теста		<i>FK3</i>

Таблица 8 – Описание атрибутов сущности «*taskQuestion*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDcategory</i>	<i>ID</i> категории теста	<i>PK</i>	
<i>category</i>	Названии категории теста		
<i>questionList</i>	<i>ID</i> вопроса		<i>FK</i>

Таблица 9 – Описание атрибутов сущности «*questionList*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDquestion</i>	<i>ID</i> вопроса	<i>PK</i>	
<i>question</i>	Текст вопроса		
<i>checkedAnswer</i>	Правильный ответ		
<i>answerUser</i>	Ответ пользователя		
<i>answerList IDanswer</i>	<i>ID</i> ответа		<i>FK</i>

Таблица 10 – Описание атрибутов сущности «*answerList*»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
<i>IDanswer</i>	<i>ID</i> ответа	<i>PK</i>	
<i>answer</i>	Текст ответа		

Процесс нормализации – это последовательный процесс приведения базы данных к эталонному виду, то есть переход от одной нормальной формы к следующей [5].

Отношение находится в 1НФ, если все строки таблицы различны и все атрибуты являются атомарными, то есть на пересечении одной строки и одного столбца находится единственное значение.

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

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

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

- сущность «*admins*» имеет ключевой атрибут «*IDadmin*». Атрибуты «*login*», «*password*» данной сущности являются простыми и неделимыми;
- сущность «*users*» имеет ключевой атрибут «*IDuser*». Атрибуты «*login*», «*password*» данной сущности являются простыми и неделимыми;
- сущность «*taskPassedQuestion*» имеет ключевой атрибут «*IDpassedQuestion*». Атрибуты «*admins IDadmin*», «*users IDuser*», «*taskQuestion IDcategory*» данной сущности являются простыми и неделимыми;
- сущность «*taskQuestion*» имеет ключевой атрибут «*IDcategory*». Атрибуты «*category*» и «*questionList*» данной сущности являются простыми и неделимыми;
- сущность «*questionList*» имеет ключевой атрибут «*IDquestion*». Атрибуты «*question*», «*checkedAnswer*», «*answerUser*», «*answerList IDanswer*» данной сущности являются простым и неделимым;
- сущность «*answerList*» имеет ключевой атрибут «*IDanswer*». Атрибут «*answer*» данной сущности является простым и неделимым.

Отношения реляционной модели данных, представленной на рисунке 9, находятся во 2НФ, так как их неключевые атрибуты функционально полно зависят от первичных ключей.

Отношение «*admins*» находится во 2НФ, так как имеет единственный первичный ключ «*IDadmin*». Неключевые атрибуты «*login*» и «*password*» данного отношения функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Отношение «*users*» находится во 2НФ, так как имеет единственный первичный ключ «*IDuser*». Неключевые атрибуты «*login*» и «*password*» данного отношения функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Отношение «*taskPassedQuestion*» находится во 2НФ, так как имеет единственный первичный ключ «*IDpassedQuestion*». Неключевые атрибуты «*admins IDadmin*», «*users IDuser*» и «*taskQuestion IDcategory*» данного

					ПГУ 1.090501.04.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

отношения функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Отношение «*taskQuestion*» находится во 2НФ, так как имеет единственный первичный ключ «*IDcategory*». Неключевые атрибуты «*category*» и «*questionList*» данного отношения функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Отношение «*questionList*» находится во 2НФ, так как имеет единственный первичный ключ «*IDquestion*». Неключевые атрибуты «*question*», «*checkedAnswer*», «*answerUser*» и «*answerList IDanswer*» данного отношения функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Отношение «*answerList*» находится во 2НФ, так как имеет единственный первичный ключ «*IDanswer*». Неключевой атрибут «*answer*» данного отношения функционально полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данное отношение находится во 2НФ.

Сущности реляционной модели, представленной на рисунке 7, находятся в 3НФ, так как ни в одной из сущностей не существует транзитивной зависимости.

В сущности «*admins*» атрибуты «*login*» и «*password*» не зависят друг от друга. Следовательно, можно сделать вывод о том, что данное отношение находится в 3НФ.

В сущности «*users*» атрибуты «*login*» и «*password*» не зависят друг от друга. Следовательно, можно сделать вывод о том, что данное отношение находится в 3НФ.

В сущности «*taskPassedQuestion*» атрибуты «*admins IDadmin*», «*users IDuser*» и «*taskQuestion IDcategory*» не зависят друг от друга. Следовательно, можно сделать вывод о том, что данное отношение находится в 3НФ.

В сущности «*taskQuestion*» атрибуты «*category*» и «*questionList*» не зависят друг от друга. Следовательно, можно сделать вывод о том, что данное

					ПГУ 1.090501.04.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

отношение находится в 3НФ.

В сущности «questionList» атрибуты «question», «checkedAnswer», «answerUser» и «answerList IDanswer» не зависят друг от друга. Следовательно, можно сделать вывод о том, что данное отношение находится в 3НФ.

В сущности «answerList» имеется один неключевой атрибут «answer». Следовательно, можно сделать вывод о том, что данное отношение находится в 3НФ. Таким образом, была разработана логическая модель данных.

На основании разработанной логической модели данных была построена физическая модель данных. Физическая модель данных разработана для СУБД PostgreSQL и представлена на рисунке 10.

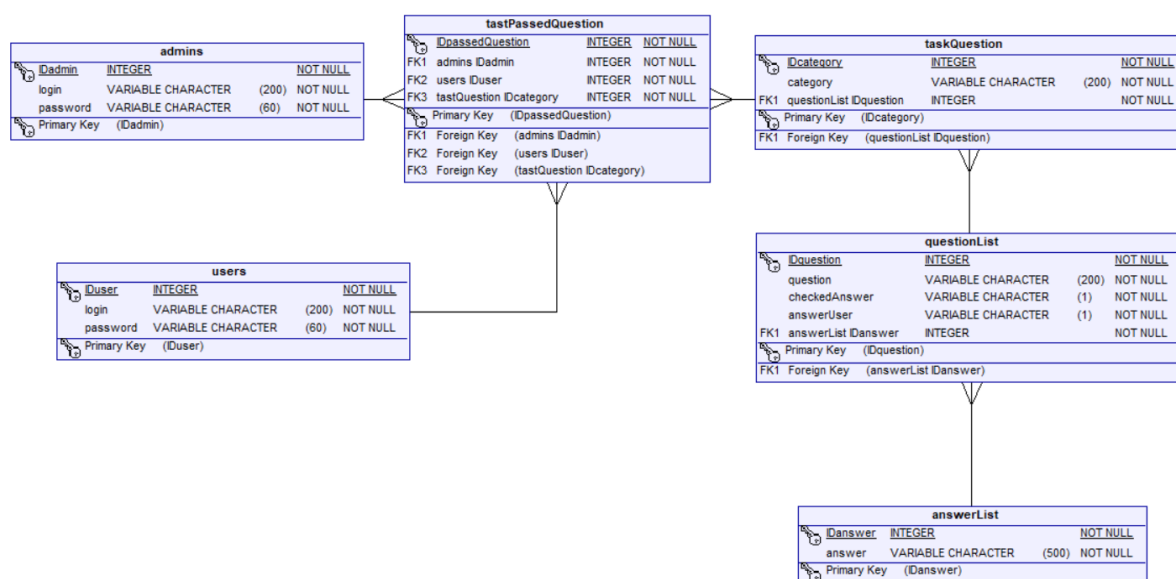


Рисунок 10 – Физическая модель данных

Физическая модель данных содержит шесть таблиц. Описание таблиц физической модели приведено в таблицах 11-16.

Таблица 11 – Описание полей таблицы «admins»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDadmin</i>	<i>IDadmin</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>login</i>	<i>login</i>	Не допустимо		<i>VARCHAR(200)</i>
<i>password</i>	<i>password</i>	Не допустимо		<i>VARCHAR(60)</i>

Таблица 12 – Описание полей таблицы «users»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDuser</i>	<i>IDuser</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>login</i>	<i>login</i>	Не допустимо		<i>VARCHAR(200)</i>
<i>password</i>	<i>password</i>	Не допустимо		<i>VARCHAR(60)</i>

Таблица 13 – Описание полей таблицы «taskPassedQuestion»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDpassedQuestion</i>	<i>IDpassedQuestion</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>admins IDadmin</i>	<i>admins IDadmin</i>	Не допустимо	<i>FK1</i>	<i>INTEGER</i>
<i>users IDuser</i>	<i>users IDuser</i>	Не допустимо	<i>FK2</i>	<i>INTEGER</i>
<i>taskQuestion IDcategory</i>	<i>taskQuestion IDcategory</i>	Не допустимо	<i>FK3</i>	<i>INTEGER</i>

Таблица 14 – Описание полей таблицы «taskQuestion»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDcategory</i>	<i>IDcategory</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>category</i>	<i>category</i>	Не допустимо		<i>VARCHAR(200)</i>
<i>questionList</i>	<i>questionList</i>	Не допустимо	<i>FK</i>	<i>INTEGER</i>

Таблица 15 – Описание полей таблицы «questionList»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDquestion</i>	<i>IDquestion</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>question</i>	<i>question</i>	Не допустимо		<i>VARCHAR(200)</i>
<i>checkedAnswer</i>	<i>checkedAnswer</i>	Не допустимо		<i>VARCHAR(1)</i>
<i>answerUser</i>	<i>answerUser</i>	Не допустимо		<i>VARCHAR(1)</i>
<i>answerList IDanswer</i>	<i>answerList IDanswer</i>	Не допустимо	<i>FK</i>	<i>INTEGER</i>

Таблица 16 – Описание полей таблицы «*answerList*»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
<i>IDanswer</i>	<i>IDanswer</i>	Не допустимо	<i>PK</i>	<i>INTEGER</i>
<i>answer</i>	<i>answer</i>	Не допустимо		<i>VARCHAR(200)</i>

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

4.2 Описание функций программы

Основной функционал программы для интерактивного тестирования *IT* специалистов реализован в нескольких модулях, представленных в виде веб-страниц. Модули программы приведены в таблице 17.

Таблица 17 – Модули программы

Модуль	Назначение
<i>Home</i>	Модуль домашней страницы
<i>Entry</i>	Модуль для авторизации или регистрации
<i>CreateTask</i>	Модуль для создания теста
<i>TaskPage</i>	Модуль для прохождения теста
<i>Results</i>	Модуль для просмотра результатов пройденных тестов
<i>UserPage</i>	Страница пользователя

Модуль *Home* представляет собой домашнюю страницу, на которой пользователь может зарегистрироваться и войти в систему.

В модуле *Home* генерируются следующие перечисленные ниже элементы управления.

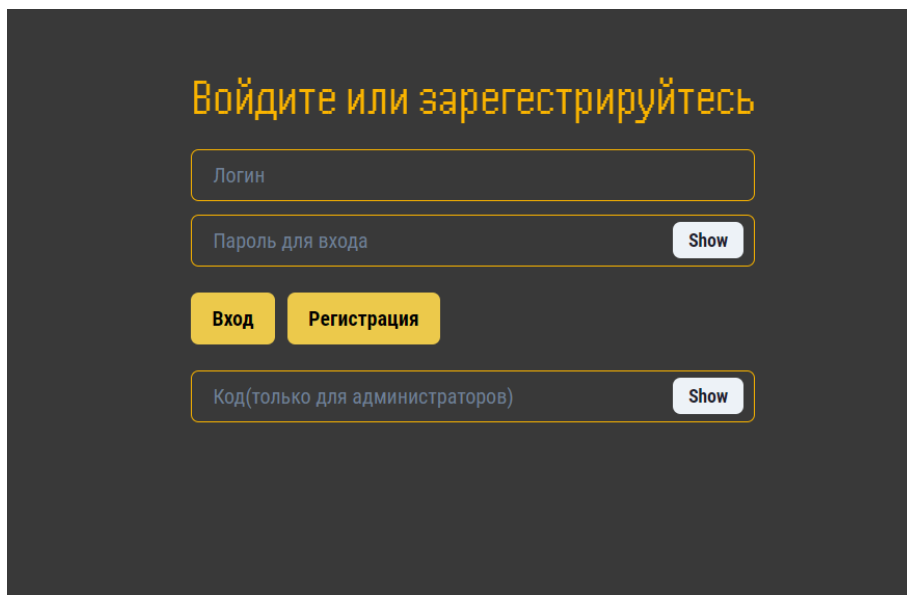
Текстовые поля ввода:

- поле ввода «Логин» для ввода логина;
- поле ввода «Пароль для входа» для ввода пароля;
- поле ввода «Код (только для администраторов)».

Кнопки:

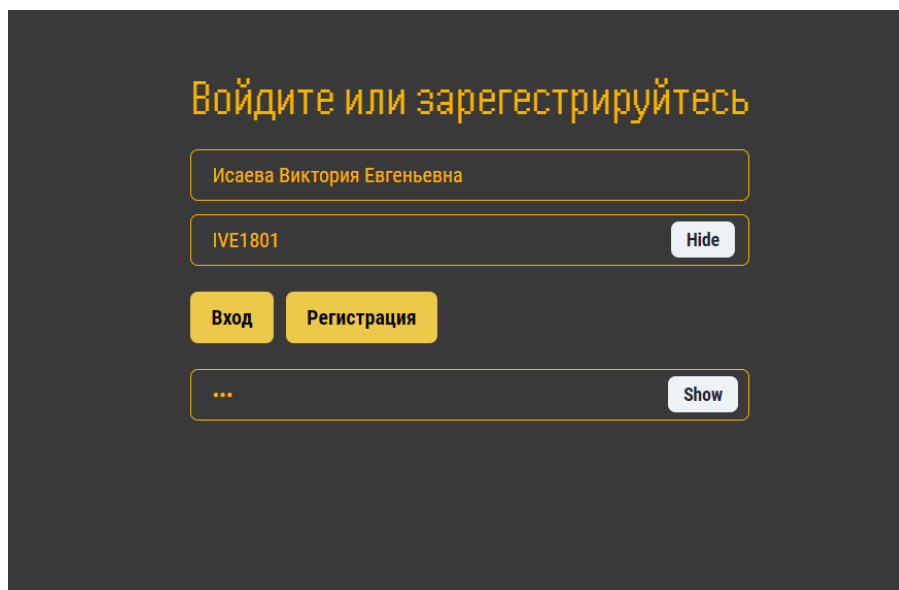
- кнопка «Вход» для выполнения входа в систему;
- кнопка «Регистрация» для регистрации в система;
- две кнопки «*Show/Hide*» для отображения и скрывтия введенных данных соответственно.

Графический интерфейс модуля *Home* в пустыми и заполненными полями ввода представлены на рисунках 11 и 12 соответственно.



The screenshot shows a dark-themed login interface. At the top, the text "Войдите или зарегистрируйтесь" is displayed in orange. Below it are four input fields: "Логин", "Пароль для входа" (with a "Show" button), "Вход" (yellow button), "Регистрация" (yellow button), and "Код(только для администраторов)" (with a "Show" button).

Рисунок 11 – Модуль *Home*. Домашняя страница



The screenshot shows the same login interface as Figure 11, but with the input fields filled. The "Логин" field contains "Исаева Виктория Евгеньевна", the "Пароль для входа" field contains "IVE1801" (with a "Hide" button), and the "Код(только для администраторов)" field contains "...". The "Вход" and "Регистрация" buttons remain yellow.

Рисунок 12 – Модуль *Home*. Домашняя страница с введенными данными

					ПГУ 1.090501.04.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

Функции данного модуля реализуются следующим кодом:

```
const Home: FC<Props> = ({ taskQuestion }) => {
  const handleCategory = (idCategory:number) => {
    localStorage.setItem('idCategory', JSON.stringify(idCategory))
  }
  const renderCard = () => {
    if (taskQuestion.length !== 0) {
      return (
        taskQuestion.map(el => (
          <Card key={ el.idCategory } className={ styles.card_color
        }>
          <CardHeader>
            <Heading size="md">{el.category}</Heading>
          </CardHeader>
          <CardBody>
            { /* <Text>View a summary of all your customers over
the last month.</Text> */ }
          </CardBody>
          <CardFooter>
            <Link to="/taskPage">
              <Button colorScheme="yellow" onClick={ () => {
handleCategory(el.idCategory) } }>Пройти тест</Button>
            </Link>
          </CardFooter>
        </Card>
      ))
    )
  }
  return null
}
```

Для регистрации в системе пользователь должен ввести логин и пароль в соответствующие поля ввода, затем нажать кнопку «Регистрация». Если пользователь является администратором, он должен также ввести код администратора в соответствующее поле ввода.

					ПГУ 1.090501.04.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Для входа в систему пользователь должен выполнить описанные выше действия и вместо кнопки «Регистрация» нажать кнопку «Вход».

После регистрации или авторизации в системе происходит проверка правильности ввода данных и вход в систему, реализованный в модуле *Entry*:

```
const handleEntry = () => {
  if (codeAdmin !== CODE) {
    const protectLogin = users.find(el => el.login === userLogin)
    if (protectLogin?.password === userPassword) {
      navigate('/home');
      serCurrentUser({ ...protectLogin })
      setFlagAlarmError(false)

      setCodeAdmin('')
    } else {
      setFlagAlarmError(true)
    }
  } else {
    const protectLogin = admins.find(el => el.login === userLogin)
    if (protectLogin?.password === userPassword) {
      navigate('/home');
      serCurrentUser({ ...protectLogin })
      setFlagAlarmError(false)
    } else {
      setFlagAlarmError(true)
    }
  }
}
```

Если данные введены верно, то выполняется переход на главную страницу, показанную на рисунке 13.

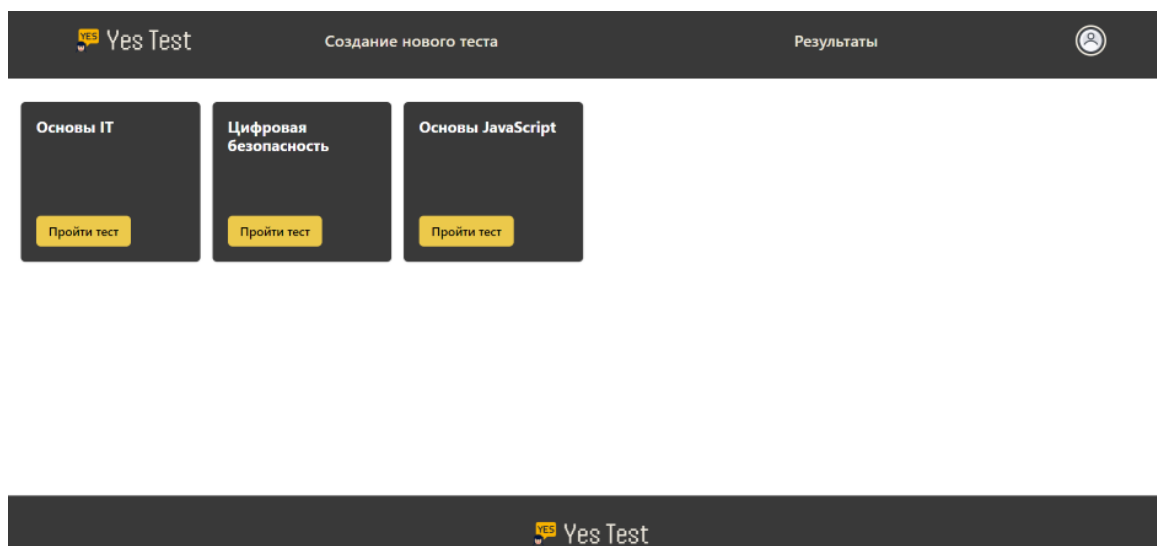


Рисунок 13 – Модуль *Home*. Главная страница
при входе от имени администратора

На главной странице программы при входе от имени администратора используются перечисленные ниже элементы управления.

Интерактивный текст:

– интерактивный текст «Создание нового теста» для перехода на страницу создания нового теста;

– интерактивный текст «Результаты» для перехода на страницу с просмотром результатов пройденных тестов.

Иконка, при нажатии на которую выполняется переход в личный кабинет.

Карточки различных тестов, содержащие кнопки «Пройти тест», при нажатии на которые выполняется переход к прохождению теста.

При входе от имени обычного интерактивный текст на главной странице отсутствует (рисунок 14).

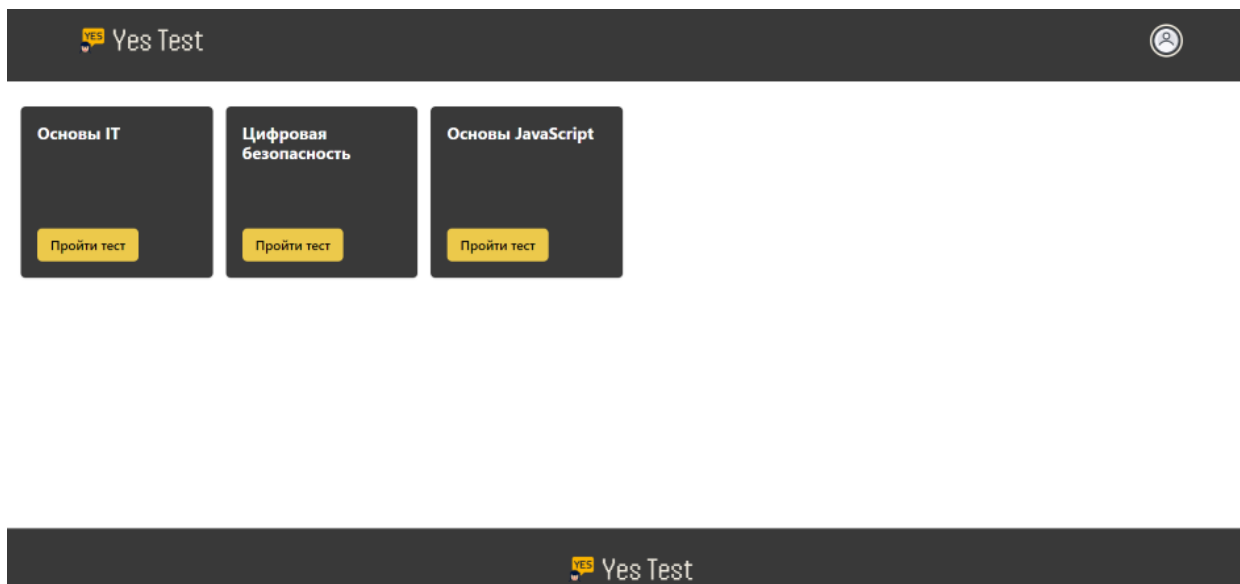


Рисунок 14 – Модуль *Home*. Главная страница
при входе от имени обычного пользователя

В случае, если поля ввода заполнены неверно или являются пустыми, пользователь видит сообщение об ошибке, представленное на рисунке 15.

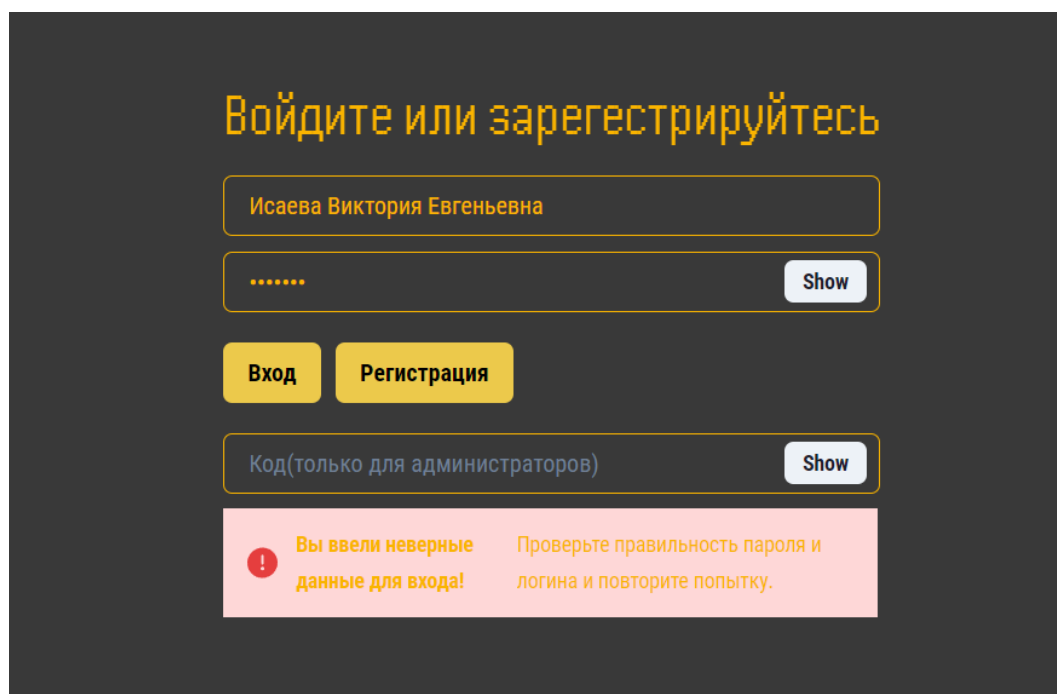


Рисунок 15 - Модуль *Entry*. Сообщение об ошибке

Модуль *CreateTask* представляет собой веб-страницу, на которой пользователь может создать новый текст. Данная функция программы доступна только администраторам и реализованная следующим образом:

					ПГУ 1.090501.04.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

```

const renderItem = answerListItem => (
  answerListItem.map((el, index) => (
    <div className={ styles.radioBlock } key={ el.idAnswer }>
      <Radio isDisabled className={ styles.radio } color-
Scheme="green" value={ `${index + 1}` }>{el.answer}</Radio>
    </div>
  )))

const renderQuestion = () => {
  if (questionList.length !== 0) {
    return (
      questionList.map(el => (
        <div key={ el.idQuestion }>
          <p>{ el.question }</p>
          <RadioGroup defaultValue={ el.checkedAnswer }>
            <Stack>
              { renderItem(el.answerList) }
            </Stack>
          </RadioGroup>
        </div>
      ))
    )
  }
  return null
}

```

Графический интерфейс модуля *CreateTask* включает в себя перечисленные ниже элементы управления.

Интерактивный текст:

– интерактивный текст «Создание нового теста» для перехода на страницу создания нового теста;

– интерактивный текст «Результаты» для перехода на страницу с просмотром результатов пройденных тестов.

Иконка, при нажатии на которую выполняется переход в личный кабинет.

Поля ввода данных:

					ПГУ 1.090501.04.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

- поле ввода «Категория» для ввода названия категории теста;
- поле ввода для ввода текста вопроса;
- поля ввода «Вариант ответа» для ввода текста вариантов ответа.

Группа радиокнопок для отметки верного варианта ответа.

Кнопки:

- кнопки «X» для удаления варианта ответа;
- кнопка «Добавить вариант» для добавления нового варианта ответа (можно добавить до 5 вариантов);
- кнопка «Подтвердить создание вопроса» для того, чтоб закончить создание вопроса и перейти к созданию следующего вопроса;
- кнопка «Подтвердить создание теста» для завершения и сохранения теста.

Иконка «Yes Test», при нажатии на которую выполняется переход на главную страницу.

Графический интерфейс модуля *CreateTask* представлен на рисунках 16 и 17.

Рисунок 16 – Модуль *CreateTask*

Создание нового теста

Результаты

Создание нового теста

Категория

Оснoвы C++

Чему равны переменные a и x после выполнения следующего кода? int a=5; long x=3; a+=x; x=-x;

☐ 8 и 7
☒ 8 и 2
☐ 6 и 2

Вопрос

```
int a,b,c;
cin>>a>>b>>c;
cout<<a<<b<<c;
```

Варианты ответа

☐ 10 234.5 8822
☐ 10234.58822
☒ 102340

Добавить вариант

Рисунок 17 – Модуль *CreateTask*. Процесс создания теста

Модуль *TaskPage* представляет собой веб-страницу, на которой пользователь может пройти тест.

```
const handlePassed = (valuePassed, idQuestion) => {
  const passedQuestion = questionList.find(el => el.idQuestion
=== idQuestion)// ищем объект в начальном массиве
  const idxPassedQuestion = questionList.findIndex(el =>
el.idQuestion === idQuestion)// ищем индекс оюъекта из начального
массива
  if (passedTask[idxPassedQuestion]?.answerUser) {
    const newPassedQuestion = passedTask.slice()
    newPassedQuestion[idxPassedQuestion] = {
      idQuestion: passedQuestion.idQuestion,
      question: passedQuestion.question,
      checkedAnswer: passedQuestion.checkedAnswer,
      answerUser: valuePassed,
      answerList: passedQuestion.answerList,
    }
    setPassedTask(newPassedQuestion)
  } else {
    setPassedTask([ ...passedTask, {
      idQuestion: passedQuestion.idQuestion,
      question: passedQuestion.question,
```

```

        checkedAnswer: passedQuestion.checkedAnswer,
        answerUser: valuePassed,
        answerList: passedQuestion.answerList,
    ]])
}
}

const hendaleEndTest = () => {
    setSuccsesTestFlag(true)
    const localIdCategory = Number(localStorage.getItem('idCategory'))
    const date = new Date().getTime()
    if (questionList.length === passedTask.length) {
        setFlagAlarmError(false)
        setTaskPassedQuestion([ ...taskPassedQuestion, {
            idPassed: date,
            idCategory: localIdCategory,
            loginUser: currentUser.login,
            category,
            questionList: passedTask,
        },
    ])
        localStorage.setItem('taskPassedQuestion', JSON.stringify([
...taskPassedQuestion, {
            idPassed: date,
            idCategory: localIdCategory,
            loginUser: currentUser.login,
            category,
            questionList: passedTask,
        }]))
    } else {
        setFlagAlarmError(true)
    }
}

```

Графический интерфейс модуля *TaskPage* представлен на рисунке 18 и включает в себя перечисленные элементы управления.

					ПГУ 1.090501.04.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Интерактивный текст:

– интерактивный текст «Создание нового теста» для перехода на страницу создания нового теста;

– интерактивный текст «Результаты» для перехода на страницу с просмотром результатов пройденных тестов.

Интерактивный тест на странице прохождения теста отображается только при входе в систему от имени администратора.

Иконка «Yes Test», при нажатии на которую выполняется переход на главную страницу.

Иконка, при нажатии на которую выполняется переход в личный кабинет.

Несколько групп радиокнопок, количество которых зависит от количества вопросов в тесте. Радиокнопки используются для отметки правильных по мнению тестируемого ответов теста.

Кнопка «Завершить тест» для завершения тестирования.

Yes Test

Основы JavaScript

Какие функции выполняет JS? (В вопросе не идет речь про платформу Node JS)

- ☐ Создает разметку на странице сайта
- ☐ Выполняет работу с сервером
- ☐ Отвечает за работу с базами данных
- ☒ Отвечает за функции на стороне клиента
- ☐ Создает стиливое оформление сайта

Где верно указано имя переменной?

- ☒ var num_1;
- ☐ var num
- ☐ var num-1;
- ☐ ver num;
- ☐ var 2num;

В чем разница между confirm и prompt?

- ☒ prompt вызывает диалоговое окно с полем для ввода, confirm - окно с подтверждением
- ☐ confirm вызывает диалоговое окно с полем для ввода, prompt - окно с подтверждением
- ☐ Они ничем не отличаются

Что будет выведено в консоль? var b = 10; if (b % 3 == 0) { var l = 10; } console.log(l);

- ☐ Ошибку
- ☐ Значение NaN
- ☐ Число 10
- ☒ Значение пустой строки
- ☐ Значение undefined

Завершить тест

Yes Test

Рисунок 18 – Модуль *TaskPage*. Прохождение тестирования при входе в систему от имени обычного пользователя

					ПГУ 1.090501.04.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

После завершения прохождения теста пользователь видит экран с подтверждением прохождения и информацией об обратной связи, представленный на рисунке 19.

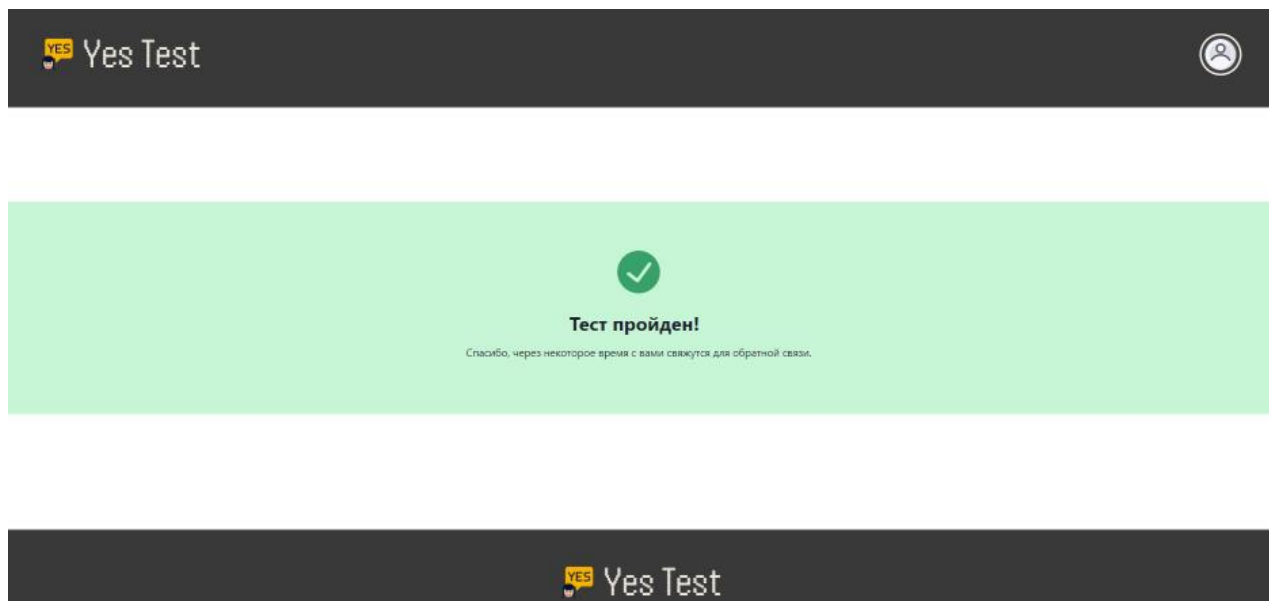


Рисунок 19 – Модуль *TaskPage*. Подтверждение прохождения пользователем теста

Администратор может результаты прохождений тестов всеми пользователями. Данный функционал реализован в модуле *Results* следующим кодом:

```
const renderUser = () => {
  if (taskPassedQuestion.length > 0) {
    return (
      taskPassedQuestion.map(el => (
        <div key={ el.idPassed } className={ styles.wrapper_per_user_card }>
          <img alt="user_logo" src={ iconTester } className={ styles.wrapper_user_logo } />
          <div>
            <strong>Логин:</strong>
            <p>{el.loginUser}</p>
          </div>
          <div>
            <strong>Категория:</strong>
            <p>{el.category}</p>
          </div>
        </div>
      ))
    )
  }
}
```

					ПГУ 1.090501.04.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

```

</div>
<div>
    <strong>Результат:</strong>
    <p>{calcAnswer(el.questionList)}</p>
</div>
<Button colorScheme="yellow" onClick={ () => handleCurrentTest(el.idPassed) }>
    Подробнее
</Button>
</div>
))
)
}
return <p>Никто пока не прошёл тест</p>
}

```

Графический интерфейс модуля *Results* представлен на рисунке 20 и состоит из окна, в котором отображаются логины (ФИО) пользователей, прошедших тестирования, а также их результаты.

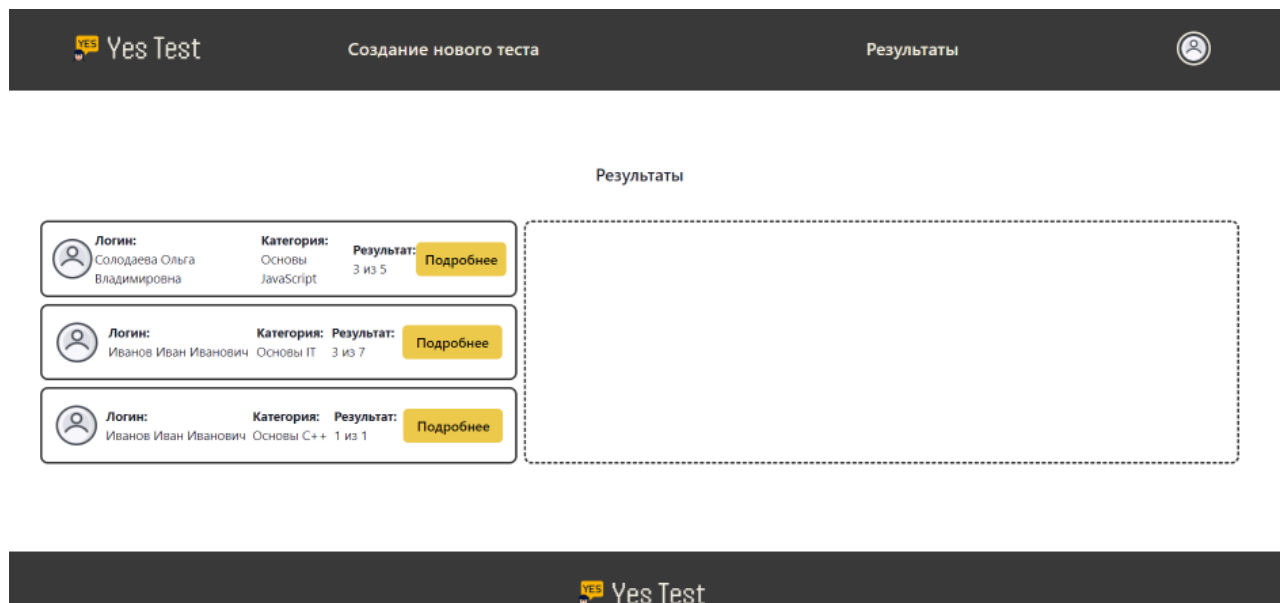


Рисунок 20 – Модуль *Results*. Просмотр результатов пройденных пользователями тестирований

На веб-странице присутствуют следующие элементы управления.

Интерактивный текст:

– интерактивный текст «Создание нового теста» для перехода на страницу создания нового теста;

– интерактивный текст «Результаты» для перехода на страницу с просмотром результатов пройденных тестов.

Интерактивный тест на странице прохождения теста отображается только при входе в систему от имени администратора.

Иконка, при нажатии на которую выполняется переход в личный кабинет.

Иконка «Yes Test», при нажатии на которую выполняется переход на главную страницу.

Кнопки «Подробнее», при нажатии на которые администратор может увидеть более подробную информацию о результатах пройденных тестов, а именно – какие ответы были даны пользователем на вопросы (рисунок 21).

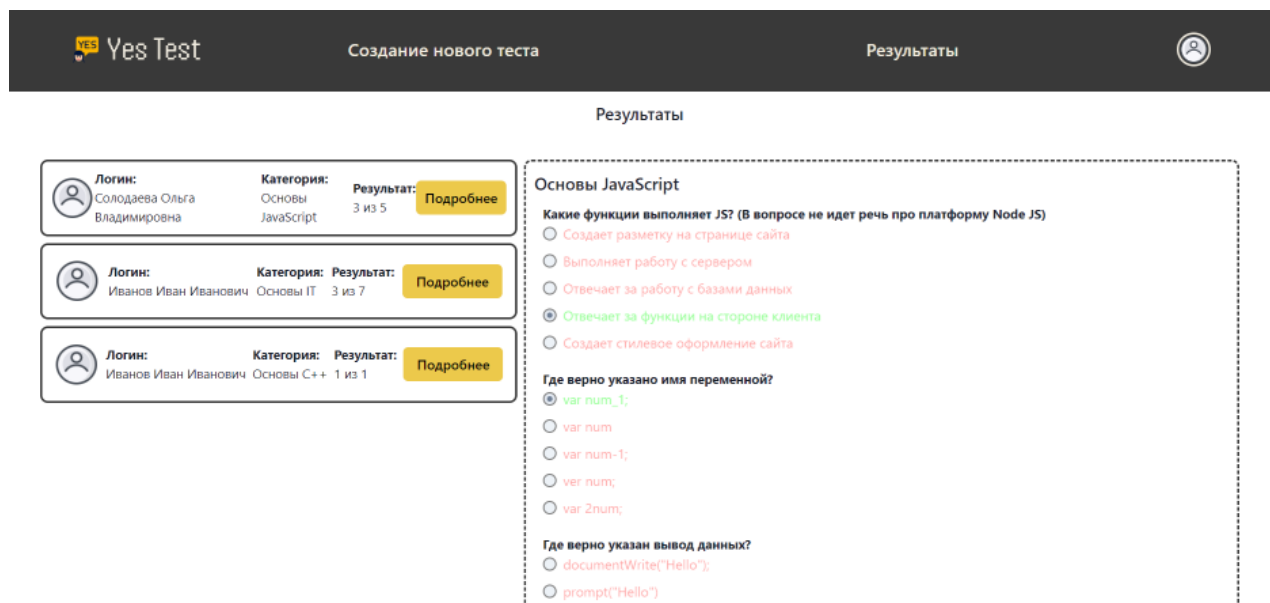


Рисунок 21 – Модуль *Results*. Отображение подробностей о результате пройденного пользователем теста

Модуль *UserPage* – это личный кабинет пользователя, на котором отображаются его личные данные. Реализация модуля *UserPage*:

```
interface Props {  
  currentUser: {
```

					ПГУ 1.090501.04.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

```

    id: number,
    login: string,
    password: string,
  },
}

const UserPage: FC<Props> = ({ currentUser }) => (
  <div>
    <section className={ styles.wrapper_user_page }>
      <div className={ styles.wrapper_user_card }>
        <div className={ styles.wrapper_user_info }>
          <img alt="user_logo" src={ iconTester } className={
styles.wrapper_user_logo } />
          <div className={ styles.fio_info }>
            <strong>Логин:</strong>
            <p>{currentUser.login}</p>
          </div>
          <div className={ styles.number_info }>
            <Link to="/">
              <Button className={ styles.btn_exit } size="sm" col-
orScheme="red">Выход</Button>
            </Link>
          </div>
        </div>
      </div>
    </section>
  </div>
)

```

Графический интерфейс модуля *UserPage* является веб-страницей и содержит следующие элементы управления.

Интерактивный текст:

- интерактивный текст «Создание нового теста» для перехода на страницу создания нового теста;
- интерактивный текст «Результаты» для перехода на страницу с просмотром результатов пройденных тестов.

					ПГУ 1.090501.04.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Интерактивный тест на странице прохождения теста отображается только при входе в систему от имени администратора.

Иконка, при нажатии на которую выполняется переход в личный кабинет.

Иконка «*Yes Test*», при нажатии на которую выполняется переход на главную страницу.

Кнопка «Выход» для выхода из системы.

Графический интерфейс модуля *UserPage* представлен на рисунке 22.

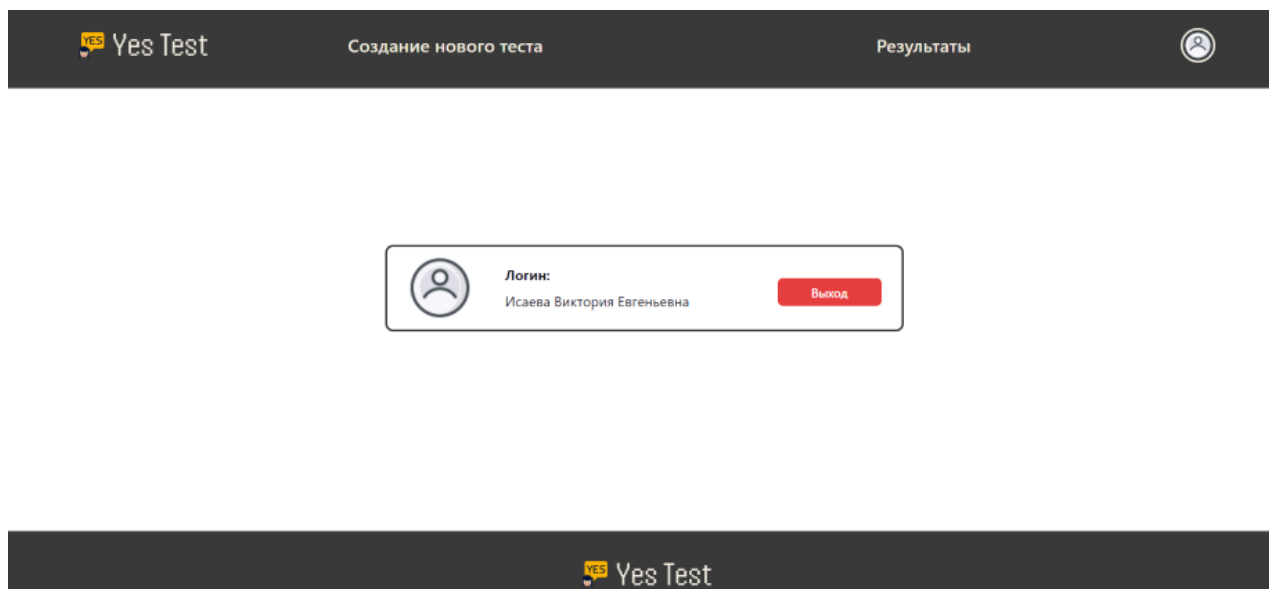


Рисунок 22 – Модуль *UserPage*. Личный кабинет пользователя

При нажатии на кнопку «Выход» выполняется выход из системы и возвращение на домашнюю страницу модуля *Home* (рисунок 11).

4.3 Описание работы с базой данных

Запускаем программу и вводим пароль на рисунке 23. На рисунке 24 отображается главное окно *pgAdmin*.

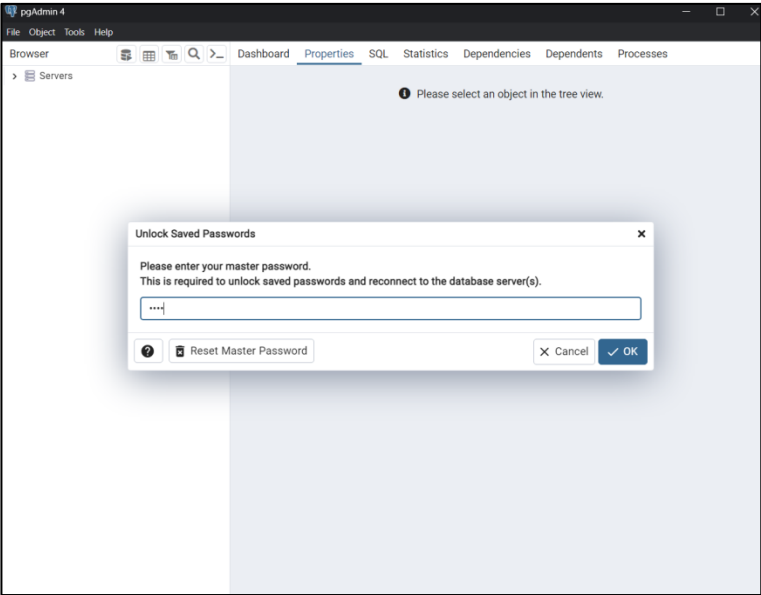


Рисунок 23 – Запуск PostgreSQL

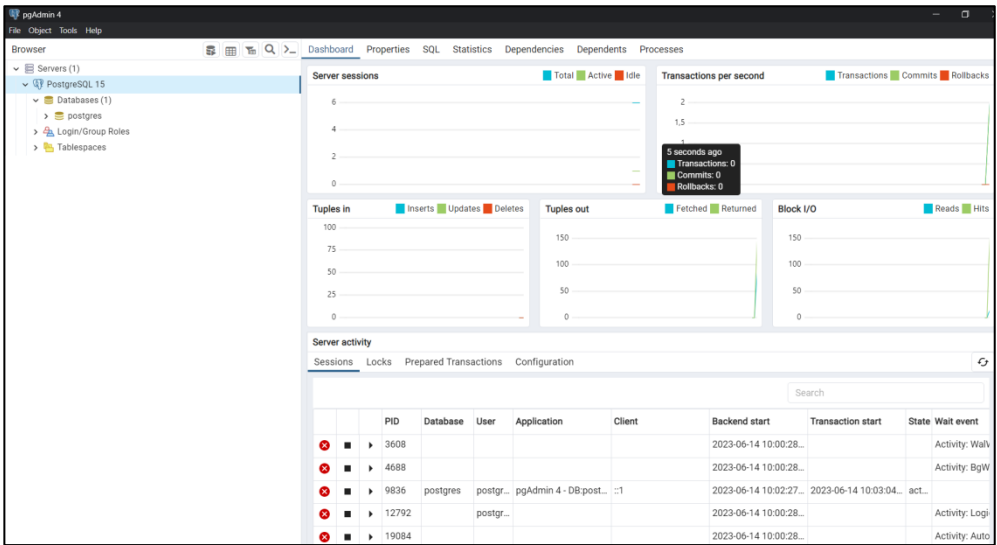


Рисунок 24 – Главное окно pgAdmin

На рисунках 25-27 создание новой базы данных «YesTest».

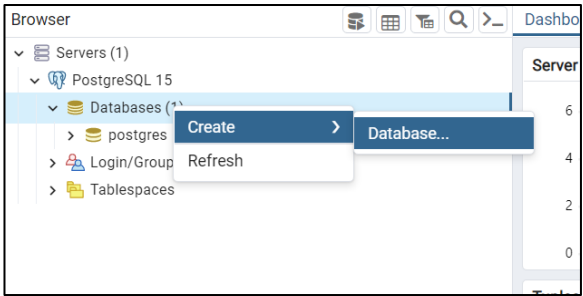


Рисунок 25 – Новая база данных

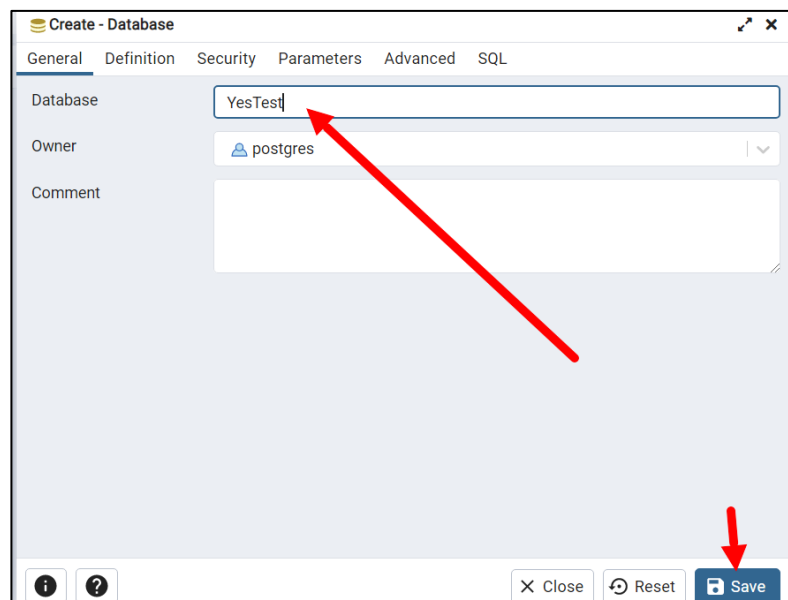


Рисунок 26 – Новая база данных



Рисунок 27 – Запрос для создания базы данных

В созданной базе данных переходим в окно *Query Editor* и создаем таблицы «*Admins*», «*Users*» (рисунки 28-29).

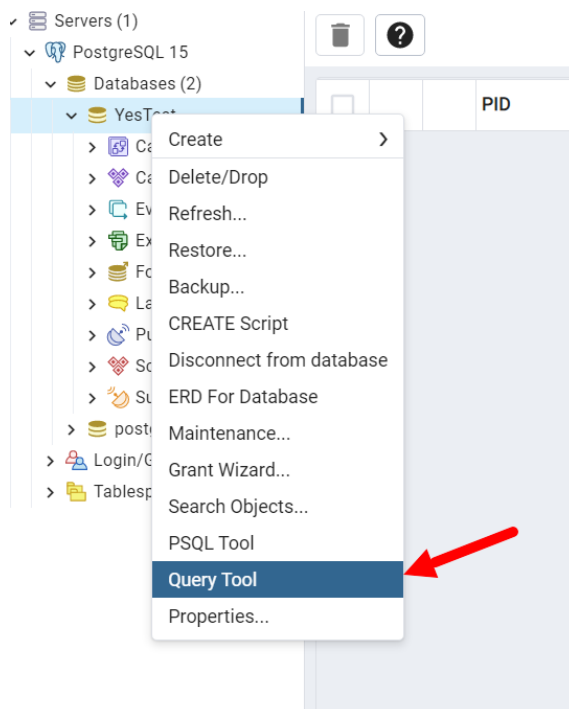


Рисунок 28 – Инструмент «*Query Tool*»

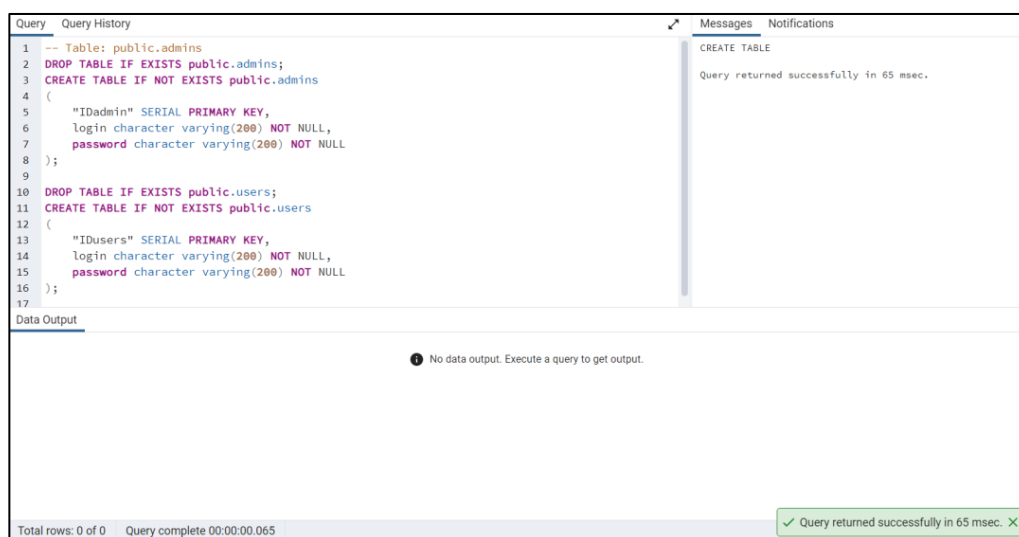


Рисунок 29 – Создание таблиц «*Admins*», «*Users*»

На рисунке 30 выполнен запрос:

```
select * from public.admins, public.answerlist, public.passedquestions, public.questionlist, public.taskquestion, public.users;
```

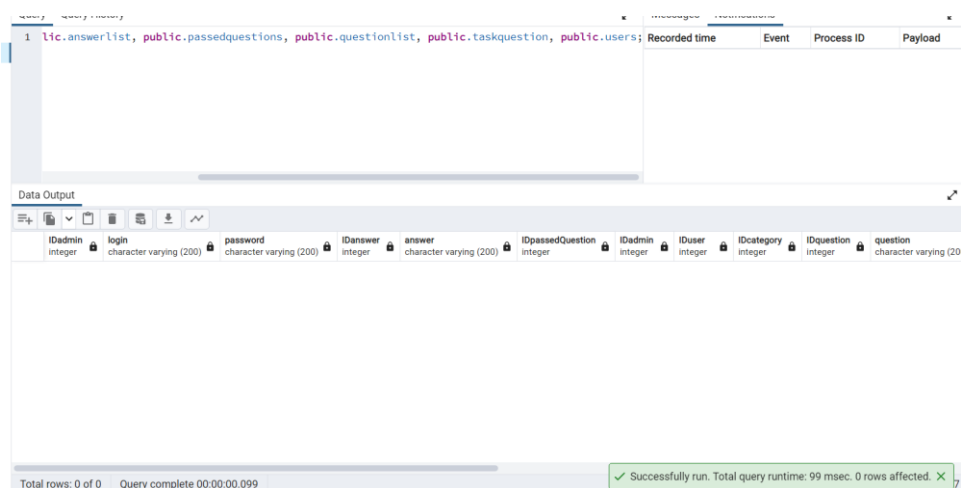


Рисунок 30 – Запрос всех полей для всех полей

5 Оценка затрат на разработку и внедрение программы для интерактивного тестирования ИТ специалистов

5.1 Организация производства разработки программы для интерактивного тестирования ИТ специалистов

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 18 и 19.

Таблица 18 – Потребность в оборудовании

Оборудование	Источник	Количество	Ориентировочная цена за единицу, рублей	Потребляемая мощность, Ватт
Компьютер <i>Lime 410</i>	ООО «ДНС Ритейл»	5	51000	150
Сервер <i>Lime S510</i>	ООО «ДНС Ритейл»	1	172000	500
Клавиатура и мышь	ООО «ДНС Ритейл»	5	900	2,5
Монитор <i>HP H2718</i>	ООО «ДНС Ритейл»	5	15500	18

Таблица 19 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
<i>Microsoft Windows 10</i>	Интернет-магазин ООО «1Бит»	29000	Бессрочная

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Антивирус <i>Kaspersky Plus 2023</i>	Интернет-магазин <i>Kaspersky URL: https://www.kaspersky.ru/</i>	1999	365
<i>VS Code</i>	Интернет-магазин <i>VS Code URL: https://code.visualstudio.com/download</i>	-	Бессрочная
СУБД <i>PostgreSQL</i>	Интернет-магазин <i>PostgreSQL URL: https://www.postgresql.org/download/</i>	-	Бессрочная
<i>OpenModelSphere 3.2</i>	Интернет-магазин <i>Open Model Sphere URL: http://www.modelsphere.com/org/download.html</i>	-	Бессрочная

Перечень дополнительных материалов, источники их приобретения, объем, и ориентировочные цены приведены в таблице 20.

Таблица 20 – Потребность в дополнительных материалах

Материал	Источник	Количество, штук	Ориентировочная цена за единицу, рублей
Стол рабочий с надстройкой	ООО «Мебельер»	1	2500
Настольная лампа	ООО «Мебельер»	5	1200
Стол компьютерный	ООО «Мебельер»	4	5000
Кресло	ООО «Мебельер»	4	5000
Бумага	Гипермаркет «Канцлер»	5	300
Канцтовары	Гипермаркет «Канцлер»	5	800

Разработка программы для интерактивного тестирования IT специалистов осуществлялась командой разработчиков, представленных в таблице 21.

Данные по основной заработной плате разработчиков программного приложения и их количественному составу приведены в таблице 22.

Таблица 21 – 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
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	- тестирование программы

Таблица 22 – Сведения о заработной плате

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	80000
Программист	1	60000
Тестировщик	1	40000

5.2 Расчет себестоимости программного обеспечения для интерактивного тестирования ИТ специалистов

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 509000 рублей и представлены в таблице 18.

Затраты на программное обеспечение в период разработки составят 30999 рублей и представлены в таблице 19.

Затраты на дополнительные материалы в период разработки составят 54000 рублей и представлены в таблице 20.

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 180000 рублей в месяц, 509000 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.

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%.

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.

					ПГУ 1.090501.04.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Данные по суммам отчислений на страховые взносы приведены в таблице 23.

Таблица 23 – 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	80000	5872,6	44758	4325,6	9198,2
Программист	60000	5872,6	39758	3945,6	8818,2
Тестировщик	40000	5872,6	6758	1285,6	6158,2

Затраты на заработную плату и отчисления на страховые взносы представлены в таблице 24.

Таблица 24 – Затраты на заработную плату

Сотрудник	Заработная плата, рублей	Общая сумма отчислений, рублей	Всего за 1 месяц, рублей	Всего за 3 месяца, рублей
Руководитель проекта	80000	9198,2	89198,2	267 594,6
Программист	60000	8818,2	68818,2	206 454,6
Тестировщик	40000	6158,2	46158,2	138 474,6
Итого:	180000	24 174,6	204174,4	612 523,2

Затраты на заработную плату команды проекта по созданию программного приложения составят 612 523,2 рублей из которых общая сумма отчислений на страховые взносы разработчиков составит 24 174,6 рублей, а заработная плата составит 636 697,8 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 5 персональных компьютеров по 51000 рублей. Таким образом затраты на амортизацию оборудования в месяц составят:

$$\frac{(5 * 51000) * 0,2}{12} = 4\,250 \text{ руб.}$$

Обслуживание оборудования составляет 15% от амортизационных отчислений.

Таким образом затраты на обслуживание оборудования составят:

$$4\,250 * 0,15 = 637,5 \text{ руб.}$$

Предположительно в день по 8 часов будут работать 3 ПК:

$$150 * 8 * 63 * 5 = 378 \text{ кВт.}$$

Таким образом, за 63 дня разработки программного приложения будет использовано 378 кВт электроэнергии на питание ноутбуков, а также 2 кВт на прочие приборы:

$$378 + 2 = 380 \text{ кВт.}$$

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

$$612\,523,2 * 0,75 = 459\,392,4 \text{ руб.}$$

Результат расчетов по статьям затрат на обслуживание приведены в таблице 25.

					ПГУ 1.090501.04.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

Себестоимость разработки определяется как сумма затрат, включающая все статьи затрат на реализацию проекта, представленное в таблице 26.

Таблица 25 – Затраты на обслуживание

Статья затрат	Показатель отчислений	Сумма отчислений в месяц, рублей	Итоговая сумма отчислений за период разработки, рублей
Амортизация оборудования	20%	4250	12750
Обслуживание оборудования	15%	637,5	1912,5
Затраты на электроэнергию	2,97 на 1 кВт/ч	1128,6	3385,8
Накладные расходы	75%	153130,8	459392,4
Аренда офиса	-	15000	45000
Итого, рублей		174146,9	522440,7

Таблица 26 – Общая сумма затрат

Статья затрат	Сумма отчислений, рублей
Приобретение оборудования	509000
Приобретение лицензионного программного обеспечения	30999
Затраты на заработную плату	612 523,2
Затраты на дополнительные материалы	54000
Затраты на обслуживание	522440,7
Итого:	1728962,9

Таким образом, сумма затрат на разработку проекта составит 1728962,9 рублей и будет соответствовать себестоимости программы для интерактивного тестирования *IT*-специалистов.

5.3 Оценка эффективности программы для интерактивного тестирования ИТ специалистов

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 программы для интерактивного тестирования ИТ специалистов также ожидаются следующие неисчисляемые эффекты:

- увеличение скорости проведения собеседований;
- минимизация ошибок, вызванных человеческим фактором;
- повышение эффективности оценки кандидатов;
- упрощение проведения собеседований;
- снижение нагрузки на сотрудников.

5.4 Анализ рисков при разработке и внедрении программы для интерактивного тестирования ИТ специалистов

При разработке и внедрении программы для интерактивного тестирования ИТ специалистов необходимо учитывать и анализировать некоторые риски.

Технические риски связаны с проблемами совместимости программы с различными операционными системами и версиями браузеров, а также с необходимыми зависимостями и инструментами разработки. Также возможны ошибки и несовершенства в программе, которые могут привести к неправильным результатам или сбоям.

К рискам в вопросах безопасности можно отнести тот факт, что программа может стать объектом хакерской атаки или взлома, особенно если она содержит конфиденциальную или чувствительную информацию. Необходимо предусмотреть меры защиты данных и проверки на уязвимости.

					ПГУ 1.090501.04.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

Если программа не будет правильно протестирована на различных сценариях использования, уровень надежности и точности результатов может быть недостаточным. Необходимо провести полное и детальное тестирование программы.

Недостаточное понимание потребностей пользователей также может стать проблемой. Если программа не будет удовлетворять потребностям ИТ специалистов или отраслевым стандартам, она может быть непригодной к использованию. Необходимо провести анализ требований и консультации с пользователями.

Разработка и внедрение программы требуют времени, денег и ресурсов. Если не будет достаточного финансирования или не будет выделены достаточные ресурсы, разработка может быть затянута или качество программы может быть недостаточным.

Также ИТ отрасль быстро развивается, и новые технологии и требования могут возникнуть по ходу разработки программы. Необходимо быть готовым к изменениям и адаптироваться к новым требованиям и технологиям.

Существует также проблема масштабируемости: если программу будут использовать большое количество пользователей или она будет использоваться на разных инфраструктурах, возможны проблемы с производительностью и масштабируемостью. Необходимо учесть это при разработке и тестировании программы.

Недостаточное количество документации о программе или недостаточное обучение пользователей могут стать причиной низкой эффективности использования программы. Необходимо предоставить достаточную документацию и обучение пользователям.

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

6 Общие принципы обеспечения безопасности на рабочем месте оператора программного приложения для интерактивного тестирования ИТ специалистов

6.1 Актуальность вопросов безопасности человека на рабочем месте

Вопросы безопасности на рабочем месте являются крайне актуальными и важными, поскольку небезопасные условия труда могут привести к серьезным последствиям для работников и организации в целом. Несчастные случаи на производстве или офисе могут вызвать травмы, инвалидность или даже смерть сотрудников.

От уровня безопасности на рабочем месте также может зависеть производительность и эффективность работы. Если сотрудники заботятся о своей безопасности и не сталкиваются с рисками, они могут работать более продуктивно и качественно. Вместе с тем, соблюдение правил и мер безопасности на рабочем месте может снизить затраты на несчастные случаи, медицинское обслуживание и правовую ответственность.

Безопасность на рабочем месте включают в себя такие аспекты, как предотвращение пожаров и взрывов, защита от химических, биологических и радиационных опасностей, предотвращение падений, употребление наркотиков и алкоголя на рабочем месте, эргономика и безопасность работы с инструментами и оборудованием, а также обеспечение безопасности информации и цифровых технологий.

В современном мире с постоянными изменениями в технологиях и рабочих процессах, актуальность вопросов безопасности на рабочем месте становится еще более значимой. Организации должны применять передовые методы и технологии для обеспечения безопасной и здоровой рабочей среды, а сотрудники должны быть обучены и осведомлены о правилах и процедурах безопасности для минимизации рисков и улучшения своего благополучия и производительности.

					ПГУ 1.090501.04.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

6.2 Требования СанПиН при работе с компьютером

СанПиН (Санитарные правила и нормы) – это документ, устанавливающий санитарные требования к условиям труда работников в различных сферах деятельности, включая работу с компьютером.

Гигиенические требования к ЭВМ и работе с ними устанавливаются СанПиН 2.2.2/2.4.1340-03 от 30.05.2003 [6]. Статьи 107 и 109 Трудового кодекса Российской Федерации [7] предписывают выделять время на отдых сотрудникам в течение рабочего дня.

Некоторые из основных требований СанПиН при работе с компьютером включают:

- оптимальное освещение рабочего места: рекомендуется использовать освещение соответствующей интенсивности и равномерности, чтобы избежать несоответствия яркости экрана и окружающей среды;

- рациональное размещение рабочего места: высота стола, стула и монитора должна быть настроена таким образом, чтобы соблюдать правильное положение тела и предотвращать напряжение спины, шеи и глаз;

- правильная организация экрана: рекомендуется настраивать яркость, контрастность и цветовую гамму монитора в соответствии с требованиями пользователя для более комфортного зрения;

- разумные паузы и физические упражнения: рекомендуется делать перерывы через определенные промежутки времени для отдыха глаз и мышц, а также для выполнения физических упражнений, чтобы снизить мускульное напряжение и предотвратить сухость глаз;

- организация работы: необходимо установить режим работы, включающий перерывы на отдых и физические упражнения, а также предусмотреть возможность выполнения разнообразных задач для снижения монотонности работы с компьютером;

- обучение и консультации: работодатели должны предоставлять своим сотрудникам обучение и консультации по соблюдению санитарных требований при работе с компьютером.

					ПГУ 1.090501.04.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

6.3 Требования безопасности при работе с электроприборами

При работе с электроприборами необходимо соблюдать следующие требования безопасности:

- электроприбор должен иметь надлежащую изоляцию и не поврежденные провода или штепсельную вилку, нельзя использовать электроприборы, у которых обнаружены повреждения;
- устройства, которые требуют заземления, должны быть правильно подключены к специальным заземляющим проводам;
- влага может представлять угрозу для безопасности и привести к поражению электрическим током, поэтому необходимо избегать использования электроприборов во влажной среде или рядом с водой;
- нельзя перегружать электроприборы, подсоединяя их к слишком мощным источникам электроэнергии или допуская перегрузку сети, так как это может привести к перегреву и повреждению прибора, а также стать причиной пожара;
- перед подключением или отключением электроприбора нужно убедиться, что он выключен и отсоединен от источника электроэнергии, а также запрещается трогать провода или штепсельные гнезда мокрыми или грязными руками;
- электроприборы должны храниться в безопасном и сухом месте, в недоступности для детей;
- электроприборы должны регулярно проходить техническое обслуживание для обнаружения и устранения возможных неисправностей или повреждений;
- перед использованием электроприбора, ознакомьтесь с инструкциями по его эксплуатации, следуйте инструкциям и не используйте приборы для работы, для которых они не предназначены.

Соблюдение этих требований безопасности поможет избежать аварийных ситуаций, повреждений электроприборов и предотвратить возможность получения электрического удара или пожара.

					ПГУ 1.090501.04.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

6.4 Влияние ПЭВМ на физиологию человека

ПЭВМ (персональные электронные вычислительные машины) имеют определенное влияние на физиологию человека. Ниже приведены некоторые основные аспекты.

– визуальная нагрузка. Работа за компьютером может привести к усталости глаз и неправильной позиции тела, вызывая напряжение мышц шеи, спины и рук. Длительное использование монитора также может вызвать сухость глаз и другие проблемы со зрением;

– сидячий образ жизни. Частое работа за компьютером может привести к сидячему образу жизни, который связан со множеством здоровотворных проблем, включая ожирение, слабые мышцы и проблемы с сердечно-сосудистой системой;

– повышенный стресс. Работа под давлением, сроки и требования могут вызывать у людей чрезмерный стресс. Это может привести к повышенному уровню тревожности, расстройству сна и другим психологическим проблемам;

– экранные времена. Значительное время, проведенное за компьютером, может привести к увеличению времени, проводимому перед экранами. Это может негативно сказаться на снах, социальном взаимодействии и общем физическом здоровье;

– электромагнитные излучения. Некоторые люди могут быть чувствительны к электромагнитным излучениям, которые испускают компьютеры и другая электроника. Это может вызывать головные боли, раздражительность и другие симптомы;

Для уменьшения негативного воздействия ПЭВМ на физиологию человека, важно принимать следующие меры:

– правильно настроить рабочее место с удобным стулом, подлокотниками и подставкой для ног;

– правильно распределить время и делать перерывы, чтобы избежать чрезмерной усталости и напряжения глаз;

					ПГУ 1.090501.04.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

- постоянно следить за позицией тела и правильно пользоваться клавиатурой и мышью для минимизации нагрузки на шею, спину и руки;
- устанавливать фильтры или использовать специальные очки для снижения напряжения глаз и сухости;
- регулярно заниматься физическими упражнениями, чтобы компенсировать сидячий образ жизни и укрепить мышцы;
- обратиться к специалистам по эргономике или здоровью на работе для получения советов по улучшению рабочих условий.

Важно соблюдать меры предосторожности и заботиться о своем физическом и психологическом здоровье при работе с ПЭВМ, чтобы минимизировать негативное воздействие на организм.

6.5 Техника безопасности при работе с персональным компьютером

При работе с персональным компьютером важно соблюдать определенные техники безопасности, чтобы минимизировать риск возможных травм и проблем со здоровьем. Существует ряд основных правил по БЖД при работе с ПК.

- позиция тела. Необходимо соблюдать правильную осанку. Стул должен быть удобным и обеспечивать поддержку спины. Ноги должны быть полностью на полу или на подставке для ног. Руки должны быть удобно размещены на клавиатуре и мыши, не вызывая напряжения в руках и плечах;
- расположение монитора. Монитор должен быть расположен на уровне глаз или немного ниже, чтобы уменьшить нагрузку на глаза и шею. Яркость и контрастность должны быть отрегулированы так, чтоб чтение было комфортным;
- отведение глаз. Перерывы и упражнения для глаз необходимы, чтоб снять напряжение и предотвратить усталость зрения;
- перерывы. Согласно типовой инструкции по охране труда при работе на ПК (персональном компьютере) ТОИ Р-45-084-01 [8], которая была утверждена Приказом Минсвязи Российской Федерации №162 от 02.07.2001, деятельность

					ПГУ 1.090501.04.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

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;

– клавиатура и мышь. Клавиатуру и мышь нужно использовать правильно, не нажимая слишком сильно и не напрягая пальцы. Расположение клавиатуры и мыши должно быть комфортным, не вызывая перенапряжение запястий или плеч;

– организация рабочего места. Рабочая область свободна от беспорядка и ненужных предметов, чтобы иметь достаточно места для работы;

– регулировка освещения. Обеспечьте хорошее освещение рабочего места. Избегайте слишком яркого или слишком тусклого света, чтобы избежать усталости глаз.

Соблюдение этих правил безопасности поможет предотвратить некоторые проблемы, связанные с работой за компьютером, и поддерживать здоровье во время использования ПК.

					ПГУ 1.090501.04.001 ПЗ	Лист
						78
Изм.	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 программы для интерактивного тестирования *IT* специалистов. Затем были выбраны средства разработки, благодаря которым было разработано программное приложение.

Программа для интерактивного тестирования *IT* специалистов реализована в соответствии с техническим заданием.

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

В разделе безопасность жизнедеятельности были проанализированы и описаны общие принципы обеспечения безопасности на рабочем месте оператора программы для интерактивного тестирования *IT* специалистов.

Задание на дипломное проектирование выполнено в полном объеме.

					ПГУ 1.090501.04.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Официальный сайт *Visual Studio Code* [Электронный ресурс] <https://code.visualstudio.com/> (дата 01.03.2023).
2. Фримен Элизабет, Фриман Эрик – «Изучаем программирование на *HTML5*». – 2013. – 592с.
3. Макфарланд Д. – «Новая большая книга *CSS*». – 2016. -720с.
4. Еременко А.В. – «Проектирование баз данных с помощью *OpenModelSphere*». – 2011. – 73с.
5. Карпова И.П. – «Базы данных. Курс лекций и материалы для практических заданий». – Учебное пособие. – 2013. – 240с.
6. 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.
7. Трудовой кодекс Российской Федерации об утверждении перечней вредных и (или) опасных производственных факторов и работ № 302. – 2011.
8. Типовая инструкция по охране труда при работе на персональном компьютере. – 2001. – 3 с.

					ПГУ 1.090501.04.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

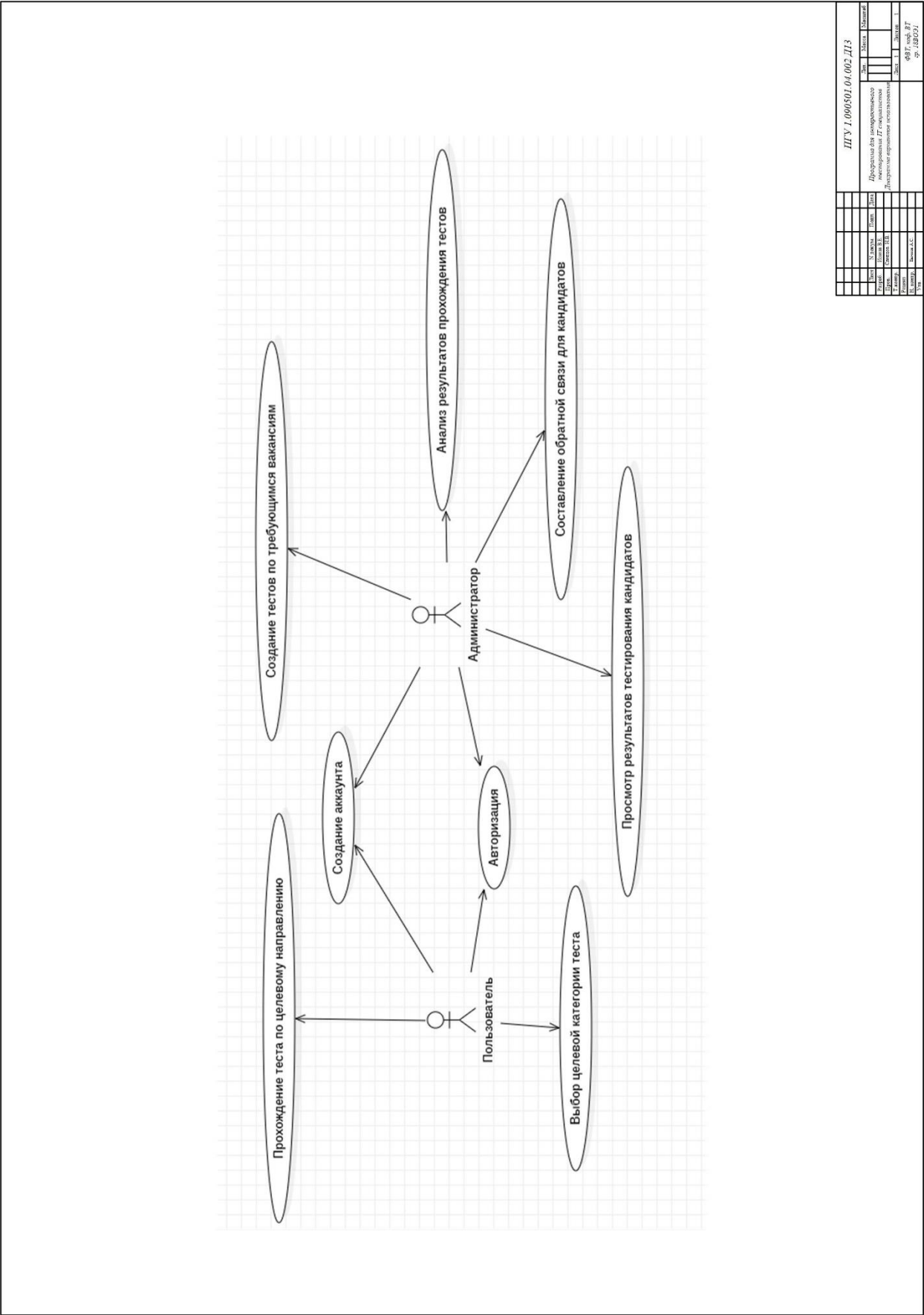
(обязательное)

Программа для интерактивного тестирования *IT* специалистов

UML-диаграммы

					ПГУ 1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		81

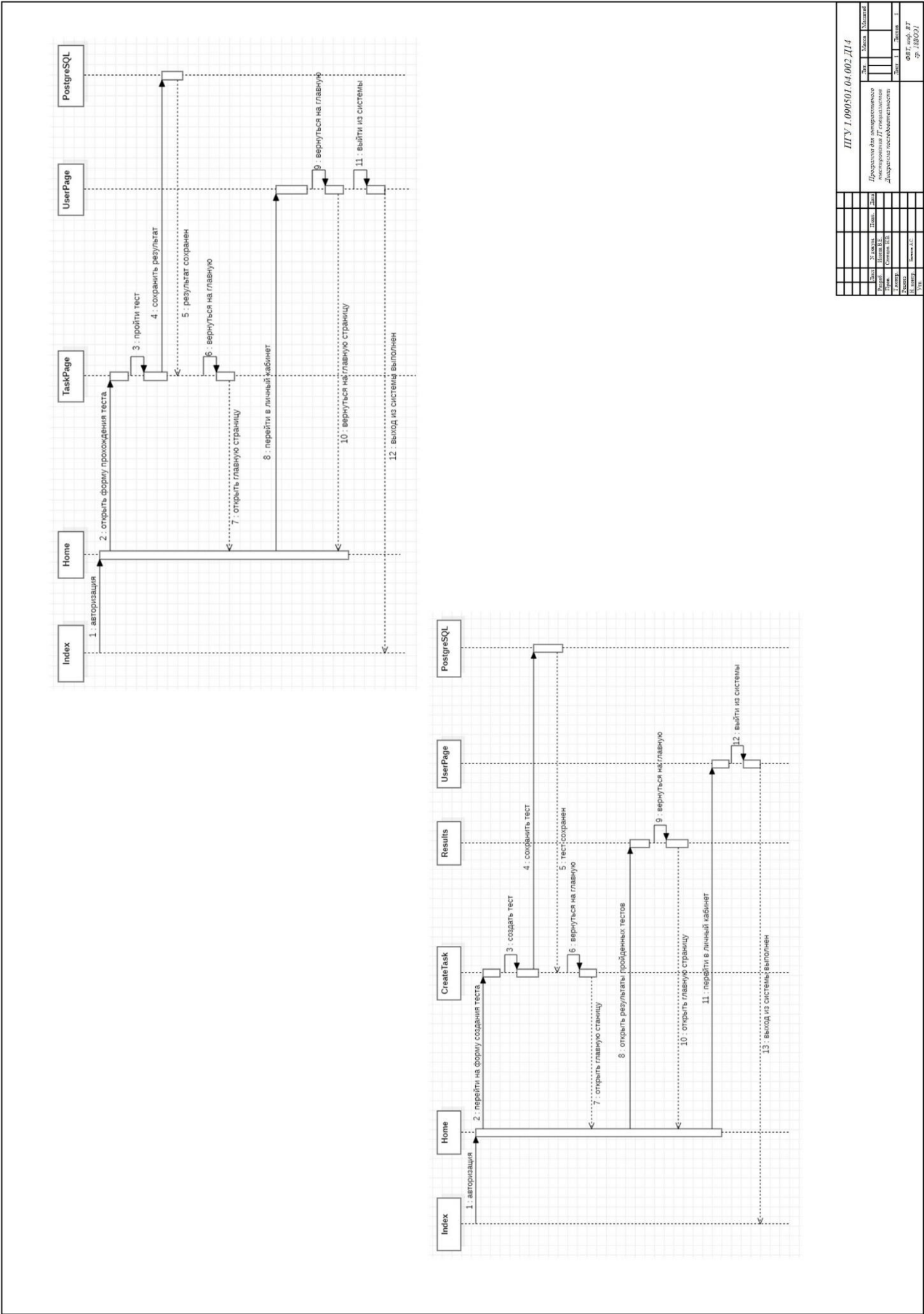
Приложение А.1 – Диаграмма вариантов использования



ПГУ 1.090501.04.001 ПЗ									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия

Изм.	Лист	№ докум.	Подпись	Дата

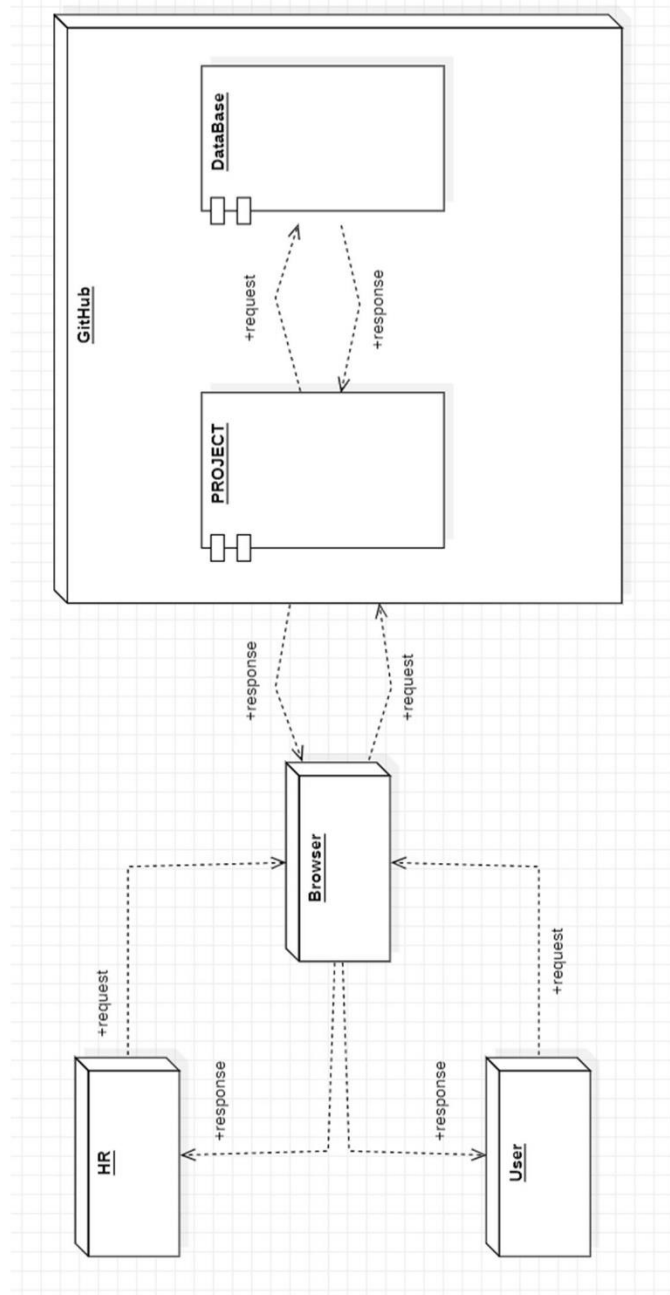
Приложение А.2 – Диаграмма последовательности



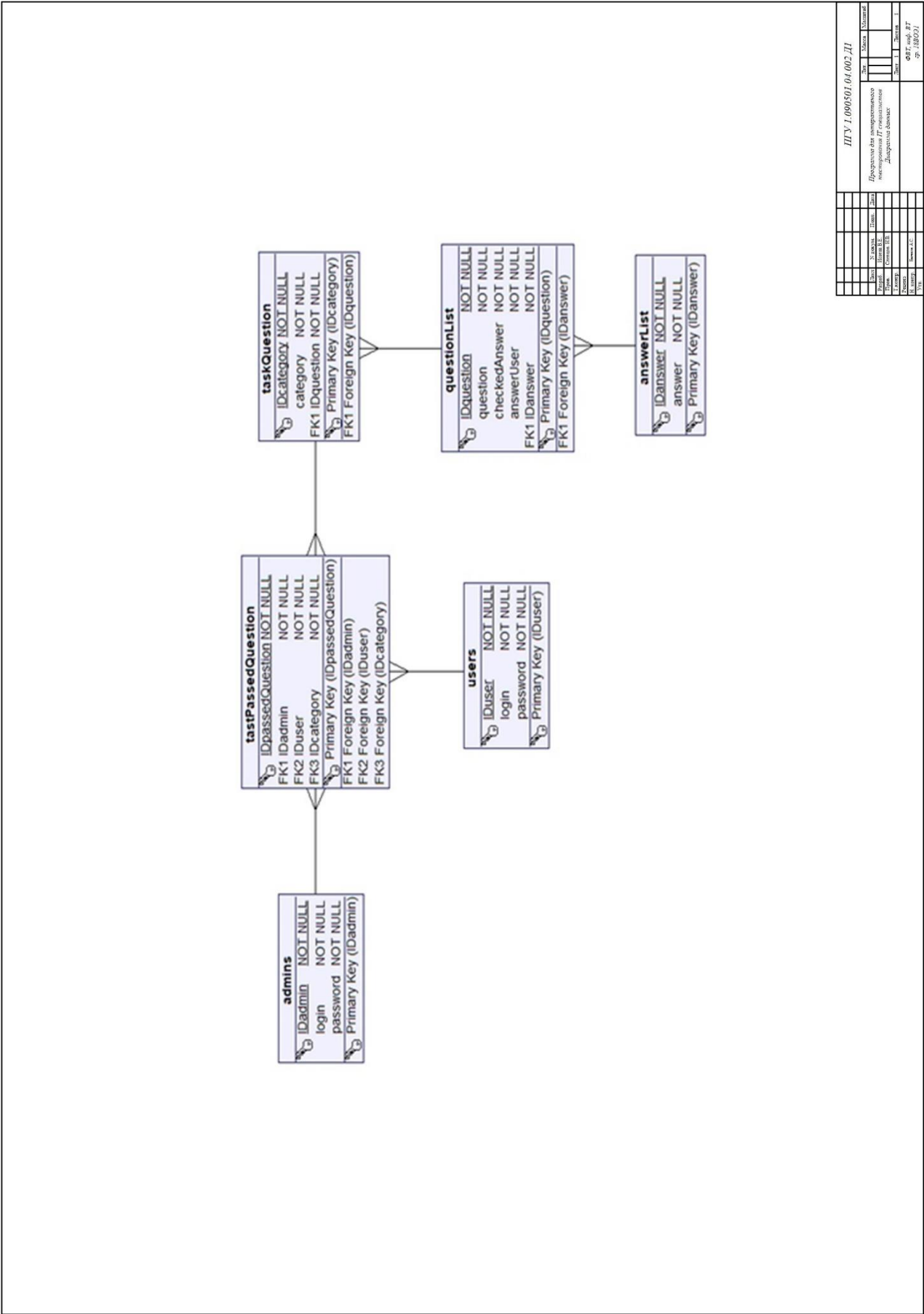
ПГУ 1.090501.04.002 Д14									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 – Диаграмма развертывания

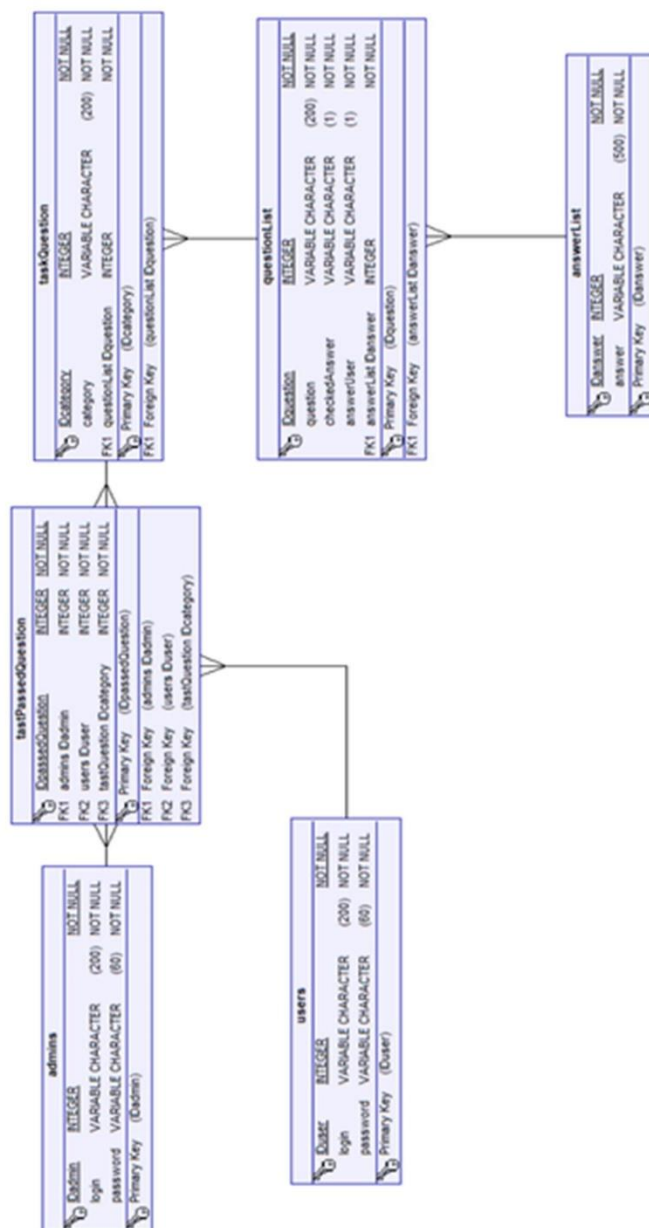
[illegible]

Приложение А.4 – Логическая модель данных



ПУ 1.090501.04.002 ДД									
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Дом

Приложение А.5 – Физическая модель данных

[illegible]

Приложение Б

(обязательное)

Программа для интерактивного тестирования *IT* специалистов

SQL-скрипт

					ПГУ 1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

-- Table: public.admins
DROP TABLE IF EXISTS public.admins;
CREATE TABLE IF NOT EXISTS public.admins
(
    "IDadmin" SERIAL PRIMARY KEY,
    login character varying(200) NOT NULL,
    password character varying(200) NOT NULL
);

DROP TABLE IF EXISTS public.users;
CREATE TABLE IF NOT EXISTS public.users
(
    "IDusers" SERIAL PRIMARY KEY,
    login character varying(200) NOT NULL,
    password character varying(200) NOT NULL
);

-- Table: public.answerList
DROP TABLE IF EXISTS public.answerList;
CREATE TABLE IF NOT EXISTS public.answerList
(
    "IDanswer" SERIAL PRIMARY KEY,
    answer character varying(200)
);

-- Table: public.questionList
DROP TABLE IF EXISTS public.questionList;
CREATE TABLE IF NOT EXISTS public.questionList
(
    "IDquestion" SERIAL PRIMARY KEY,
    question character varying(200),
    "checkedAnswer" character varying(200),
    "answerUser" character varying(200),
    "answerListIDanswer" INTEGER,
    FOREIGN KEY ("answerListIDanswer") REFERENCES
public.answerList ("IDanswer")

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

```

);

-- Table: public.taskQuestion
DROP TABLE IF EXISTS public.taskQuestion;
CREATE TABLE IF NOT EXISTS public.taskQuestion
(
    "IDcategory" SERIAL PRIMARY KEY,
    category character varying(200),
    "questionList" INTEGER,
    FOREIGN KEY ("questionList") REFERENCES public.questionList
("IDquestion")
);

-- Table: public.passedQuestions
DROP TABLE IF EXISTS public.passedQuestions;
CREATE TABLE IF NOT EXISTS public.passedQuestions
(
    "IDpassedQuestion" SERIAL PRIMARY KEY,
    "IDadmin" INTEGER NOT NULL,
    "IDuser" INTEGER NOT NULL,
    "IDcategory" INTEGER NOT NULL,
    FOREIGN KEY ("IDadmin") REFERENCES public.admins ("IDadmin"),
    FOREIGN KEY ("IDuser") REFERENCES public.users ("IDusers"),
    FOREIGN KEY ("IDcategory") REFERENCES public.taskQuestion
("IDcategory")
);

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Программа для интерактивного тестирования *IT* специалистов

Листинг программы

					ПГУ 1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

// @ts-nocheck
import React, { FC, useState } from 'react';
import {
  Alert,
  AlertDescription,
  AlertIcon,
  AlertTitle,
  Button, Input, Radio, RadioGroup, Select, Stack, Textarea,
} from '@chakra-ui/react'

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

const LABELS = {
  CREATE_TASK: 'Создание нового теста',
}

interface Props {
  setTaskQuestion: (params:{
    time: number,
    taskName: string,
    taskDescription: string,
    taskTester: string,
    type: number,
    testCheck: number,
    taskComment: string,
  }[]) => void,
  taskQuestion: {
    idCategory: number,
    category: string,
    questionList : {
      idQuestion: number,
      question: string,
      checkedAnswer: number,
      answerList : {
        idAnswer: number,
        answer: string,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }[],
        }[],
    }[],
}

const CreateTask: FC<Props> = ({ setTaskQuestion, taskQuestion })
=> {
    const [ flagAlarmError, setFlagAlarmError ] =
useState<boolean>(false)

    const [ questionList, setQuestionList ] = useState([])
    const [ category, setCategory ] = useState<string>('')
    const [ question, setQuestion ] = useState<string>('')
    const [ answerOne, setAnswerOne ] = useState<string>('')
    const [ answerTwo, setAnswerTwo ] = useState<string>('')
    const [ answerThree, setAnswerThree ] = useState<string>('')
    const [ answerFour, setAnswerFour ] = useState<string>('')
    const [ answerFive, setAnswerFive ] = useState<string>('')
    const [ checkedAnswer, setCheckedAnswer ] = useState('1')
    const [ btnFlag, setBtnFlag ] = useState(2)
    const ruleCreateQuestion = () => {
        const date = new Date().getTime()

        if (answerFive !== '') {
            if (answerOne === '' || answerTwo === '' || answerThree ===
'' || answerFour === '') {
                return setFlagAlarmError(true)
            }
            setFlagAlarmError(false)
            return (
                setQuestionList([ ...questionList, {
                    idQuestion: date,
                    question,
                    checkedAnswer,
                    answerList: [{
                        idAnswer: 1,
                        answer: answerOne,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

    },
    {
        idAnswer: 2,
        answer: answerTwo,
    },
    {
        idAnswer: 3,
        answer: answerThree,
    },
    {
        idAnswer: 4,
        answer: answerFour,
    },
    {
        idAnswer: 5,
        answer: answerFive,
    },
],
]))
)
}

```

```

if (answerFour !== '') {
    if (answerOne === '' || answerTwo === '' || answerThree ===
    '') {
        return setFlagAlarmError(true)
    }
    setFlagAlarmError(false)
    return (
        setQuestionList([ ...questionList, {
            idQuestion: date,
            question,
            checkedAnswer,
            answerList: [{
                idAnswer: 1,
                answer: answerOne,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

        },
        {
            idAnswer: 2,
            answer: answerTwo,
        },
        {
            idAnswer: 3,
            answer: answerThree,
        },
        {
            idAnswer: 4,
            answer: answerFour,
        },
    ],
    ]))
    })
}

if (answerThree !== '') {
    if (answerOne === '' || answerTwo === '') {
        return setFlagAlarmError(true)
    }
    setFlagAlarmError(false)
    return (
        setQuestionList([ ...questionList, {
            idQuestion: date,
            question,
            checkedAnswer,
            answerList: [{
                idAnswer: 1,
                answer: answerOne,
            },
            {
                idAnswer: 2,
                answer: answerTwo,
            },

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            idAnswer: 3,
            answer: answerThree,
        },
    ],
    })
    )
}
if (answerOne === '' || answerTwo === '') {
    return setFlagAlarmError(true)
}
setFlagAlarmError(false)
return (
    setQuestionList([ ...questionList, {
        idQuestion: date,
        question,
        checkedAnswer,
        answerList: [{
            idAnswer: 1,
            answer: answerOne,
        },
        {
            idAnswer: 2,
            answer: answerTwo,
        },
    ],
    ])
    )
}
const hendaleCreateQuestion = () => {
    ruleCreateQuestion()
    setQuestion('')
    setAnswerOne('')
    setAnswerTwo('')
    setAnswerThree('')
    setAnswerFour('')

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

        setAnswerFive('')
        setBtnFlag(2)
    }
    const hendaleCreateTask = () => {
        const date = new Date().getTime()

        setTaskQuestion([ ...taskQuestion, {
            idCategory: date,
            category,
            questionList,
        }])
        localStorage.setItem('taskQuestion', JSON.stringify([
...taskQuestion, {
            idCategory: date,
            category,
            questionList,
        }]))
        setQuestionList([])
        setCategory('')
        setQuestion('')

        setQuestion('')
        setAnswerOne('')
        setAnswerTwo('')
        setAnswerThree('')
        setAnswerFour('')
        setAnswerFive('')
        setBtnFlag(2)
    }
    const alarmError = () => {
        if (flagAlarmError) {
            return (
                <Alert status="error">
                    <AlertIcon />
                    <AlertTitle>Вы не заполнили данные!</AlertTitle>

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <AlertDescription>Заполните все поля и повторите
попытку.</AlertDescription>
    </Alert>
  )
}
return null
}
const btnAddAnswer = () => {
  if (btnFlag >= 5) {
    return null
  }
  return (
    <Button colorScheme="gray" onClick={ () =>
setBtnFlag(btnFlag + 1) }>
      Добавить вариант
    </Button>
  )
}
const renderItem = answerListItem => (
  answerListItem.map((el, index) => (
    <div className={ styles.radioBlock } key={ el.idAnswer }>
      <Radio isDisabled className={ styles.radio }
colorScheme="green" value={ `${index + 1}` }>{el.answer}</Radio>
    </div>
  )))
const renderQuestion = () => {
  if (questionList.length !== 0) {
    return (
      questionList.map(el => (
        <div key={ el.idQuestion }>
          <p>{ el.question }</p>
          <RadioGroup defaultValue={ el.checkedAnswer }>
            <Stack>
              { renderItem(el.answerList) }
            </Stack>
          </RadioGroup>
        </div>
      ))
    )
  }
}

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

```

        </div>
    ))
)
}
return null
}
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>Категория</p>
                    <Input
                        placeholder="Категория"
                        value={ category }
                        onChange={ e => setCategory(e.target.value) }
                    />
                    { renderQuestion() }
                    <p>Вопрос</p>
                    <Textarea
                        value={ question }
                        onChange={ e => setQuestion(e.target.value) }
                    />
                    <p>Варианты ответа</p>
                    <RadioGroup defaultValue="1">
                        <Stack>
                            <div className={ styles.radioBlock }>
                                <Radio className={ styles.radio }
colorScheme="green" value="1" onChange={ e =>
setCheckedAnswer(e.target.value) } />
                                <Input
                                    placeholder="Вариант ответа"
                                    value={ answerOne }
                                    onChange={ e => setAnswerOne(e.target.value) }

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						98
Изм.	Лист	№ докум.	Подпись	Дата		

```

        />
        <Button colorScheme="gray" onClick={ () =>
setAnswerOne('') }>
            &#10006;
        </Button>
    </div>
    <div className={ styles.radioBlock }>
        <Radio className={ styles.radio }
colorScheme="green" value="2" onChange={ e =>
setCheckedAnswer(e.target.value) } />
        <Input
            placeholder="Вариант ответа"
            value={ answerTwo }
            onChange={ e => setAnswerTwo(e.target.value) }
        />
        <Button colorScheme="gray" onClick={ () =>
setAnswerTwo('') }>
            &#10006;
        </Button>
    </div>
    {btnFlag >= 3 ? (
        <div className={ styles.radioBlock }>
            <Radio className={ styles.radio }
colorScheme="green" value="3" onChange={ e =>
setCheckedAnswer(e.target.value) } />
            <Input
                placeholder="Вариант ответа"
                value={ answerThree }
                onChange={ e =>
setAnswerThree(e.target.value) }
            />
            <Button colorScheme="gray" onClick={ () =>
setAnswerThree('') }>
                &#10006;
            </Button>
        </div>
    )

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

```

        ) : null}
        {btnFlag >= 4 ? (
            <div className={ styles.radioBlock }>
                <Radio className={ styles.radio }
colorScheme="green" value="4" onChange={ e =>
setCheckedAnswer(e.target.value) } />
                <Input
                    placeholder="Вариант ответа"
                    value={ answerFour }
                    onChange={ e =>
setAnswerFour(e.target.value) }
                />
                <Button colorScheme="gray" onClick={ () =>
setAnswerFour('') }>
                    &#10006;
                </Button>
            </div>
        ) : null}
        {btnFlag === 5 ? (
            <div className={ styles.radioBlock }>
                <Radio className={ styles.radio }
colorScheme="green" value="5" onChange={ e =>
setCheckedAnswer(e.target.value) } />
                <Input
                    placeholder="Вариант ответа"
                    value={ answerFive }
                    onChange={ e =>
setAnswerFive(e.target.value) }
                />
                <Button colorScheme="gray" onClick={ () =>
setAnswerFive('') }>
                    &#10006;
                </Button>
            </div>
        ) : null}
    </Stack>

```

```

        </RadioGroup>
        { alarmError() }
        { btnAddAnswer() }
        <Button colorScheme="yellow" onClick={ () =>
hendaleCreateQuestion() }>
            Подтвердить создание вопроса
        </Button>
        <Button colorScheme="yellow" onClick={ () =>
hendaleCreateTask() }>
            Подтвердить создание теста
        </Button>
    </div>
</div>
</section>
</div>
);
}

export default CreateTask;
import React, { FC, useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import {
    Alert,
    AlertDescription,
    AlertIcon,
    AlertTitle,
    Input, InputGroup, InputRightElement, Button, ButtonGroup,
} from '@chakra-ui/react'
import { useNavigate } from 'react-router-dom';
import { observer } from 'mobx-react';
import styles from '../styles/index.module.css'
import Card from '../../../components/Card';
const CODE = '123'
interface Props {
    users: {
        id: number,
        login: string,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

```

        password: string,
    }[],
    admins: {
        id: number,
        login: string,
        password: string,
    }[],
    setUsers: (params:{
        id: number,
        login: string,
        password: string,
    }[]) => void,
    setAdmins: (params:{
        id: number,
        login: string,
        password: string,
    }[]) => void,
    serCurrentUser : (params:{
        id: number,
        login: string,
        password: string,
    }) => void,
}

const Entry: FC<Props> = observer(({
    users,
    admins,
    setUsers,
    setAdmins,
    serCurrentUser,
})) => {
    const [ showPasswod, setShowPasswod ] = useState(false)
    const [ showCode, setShowCode ] = useState(false)
    const [ flagAlarmError, setFlagAlarmError ] = useState(false)
    const [ userLogin, setUserLogin ] = useState('')
    const [ userPassword, setUserPassword ] = useState('')
    // console.log('🔗 ~ userLogin в компоненте:', userLogin)

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

```

// console.log('🔑 ~ userPassword в компоненте:', userPassword)

const [ codeAdmin, setCodeAdmin ] = useState('')
// console.log('🔑 ~ codeAdmin в компоненте', codeAdmin)

const navigate = useNavigate();

const handleClickPasswod = () => setShowPasswod(!showPasswod)
const handleClickCode = () => setShowCode(!showCode)

const handleRegistration = () => {
  const date = new Date().getTime()

  if (userLogin === '' || userPassword === '') {
    setFlagAlarmError(true)
  } else {
    setFlagAlarmError(false)

    if (codeAdmin !== CODE) {
      setUsers([ ...users, { id: date, login: userLogin,
password: userPassword } ])
      serCurrentUser({ id: date, login: userLogin, password:
userPassword })
      localStorage.setItem('users', JSON.stringify([ ...users, {
        id: date, login: userLogin, password: userPassword,
      } ]))
      localStorage.setItem('currentUser', JSON.stringify([ { id:
date, login: userLogin, password: userPassword } ]))
      setCodeAdmin('')
      navigate('/home');
      setFlagAlarmError(false)
    } else {
      serCurrentUser({ id: date, login: userLogin, password:
userPassword })
      setAdmins([ ...admins, { id: date, login: userLogin,
password: userPassword } ])
    }
  }
}

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

```

        localStorage.setItem('admins', JSON.stringify([ ...admins,
{
    id: date, login: userLogin, password: userPassword,
    ]]))
        localStorage.setItem('currentUser', JSON.stringify([ { id:
date, login: userLogin, password: userPassword } ]))
    }
    navigate('/home');
    setFlagAlarmError(false)
}
}

const handleEntry = () => {
    if (codeAdmin !== CODE) {
        const protectLogin = users.find(el => el.login ===
userLogin)
        if (protectLogin?.password === userPassword) {
            navigate('/home');
            serCurrentUser({ ...protectLogin })
            setFlagAlarmError(false)

            setCodeAdmin('')
        } else {
            setFlagAlarmError(true)
        }
    } else {
        const protectLogin = admins.find(el => el.login ===
userLogin)
        // console.log('🔑 ~ adminLogin:', adminLogin)
        // console.log('🔑 ~ admins:', admins)
        // console.log('🔑 ~ protectLogin:', protectLogin)
        if (protectLogin?.password === userPassword) {
            navigate('/home');
            serCurrentUser({ ...protectLogin })
            setFlagAlarmError(false)
        } else {
            setFlagAlarmError(true)

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						104
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }
  }
}

const alarmError = () => {
  if (flagAlarmError) {
    return (
      <div className={ styles.alarmMsg }>
        <Alert status="error">
          <AlertIcon />
          <AlertTitle>Вы ввели неверные данные для
входа!</AlertTitle>
          <AlertDescription>Проверьте правильность пароля и
логина и повторите попытку.</AlertDescription>
        </Alert>
      </div>
    )
  }
  return null
}

return (
  <div className={ styles.wrapper_all }>
    <section className={ styles.wrapper_entry }>
      <div className={ styles.wrapper_entry_user }>
        <p className={ styles.title }>Войдите или
зарегистрируйтесь</p>
        <Input
          focusBorderColor="white.400"
          placeholder="Логин"
          onChange={ e => setUserLogin(e.target.value) }
        />
        <InputGroup size="md">
          <Input
            pr="4.5rem"
            focusBorderColor="white.400"
            type={ showPasswod ? 'text' : 'password' }
            placeholder="Пароль для входа"

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

```

        onChange={ e => setUserPassword(e.target.value) }
      />
      <InputRightElement width="4.5rem">
        <Button h="1.75rem" size="sm" onClick={
handleClickPasswod }>
          {showPasswod ? 'Hide' : 'Show'}
        </Button>
      </InputRightElement>
    </InputGroup>
    <div className={ styles.wrapper_btn }>
      <Button colorScheme="yellow" onClick={ () =>
handleEntry() }>Вход</Button>
      <Button colorScheme="yellow" onClick={ () =>
handleRegistration() }>Регистрация</Button>
    </div>
    <InputGroup size="md">
      <Input
        pr="4.5rem"
        focusBorderColor="white.400"
        type={ showCode ? 'text' : 'password' }
        placeholder="Код(только для администраторов)"
        onChange={ e => setCodeAdmin(e.target.value) }
      />
      <InputRightElement width="4.5rem">
        <Button h="1.75rem" size="sm" onClick={
handleClickCode }>
          {showCode ? 'Hide' : 'Show'}
        </Button>
      </InputRightElement>
    </InputGroup>
    { alarmError() }
  </div>
</section>
</div>
);
})

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						106
Изм.	Лист	№ докум.	Подпись	Дата		

```

export default Entry;

import React, { FC, useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import {
  Card, CardHeader, CardBody, CardFooter, SimpleGrid, Heading,
  Text, Button,
} from '@chakra-ui/react'
import { Link } from 'react-router-dom';
import styles from '../styles/index.module.css'
interface Props {
  taskQuestion: {
    idCategory: number,
    category: string,
    questionList : {
      idQuestion: number,
      question: string,
      checkedAnswer: number,
      answerList : {
        idAnswer: number,
        answer: string,
      }[],
    }[],
  }[],
}

const Home: FC<Props> = ({ taskQuestion }) => {
  const handleCategory = (idCategory:number) => {
    localStorage.setItem('idCategory', JSON.stringify(idCategory))
  }
  const renderCard = () => {
    if (taskQuestion.length !== 0) {
      return (
        taskQuestion.map(el => (
          <Card key={ el.idCategory } className={
styles.card_color }>
            <CardHeader>

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <Heading size="md">{el.category}</Heading>
      </CardHeader>
      <CardBody>
        { /* <Text>View a summary of all your customers over
the last month.</Text> */ }
      </CardBody>
      <CardFooter>
        <Link to="/taskPage">
          <Button colorScheme="yellow" onClick={ () => {
handleCategory(el.idCategory) } }>Пройти тест</Button>
        </Link>
      </CardFooter>
    </Card>
  ))
)
}
return null
}
return (
  <div className={ styles.wrapper_all }>
    <section className={ styles.wrapper_moveList }>
      <div className={ styles.wrapper_posters }>
        <SimpleGrid spacing={ 4 } templateColumns="repeat(auto-
fill, minmax(200px, 1fr))">
          { renderCard() }
        </SimpleGrid>
      </div>
    </section>
  </div>
);
}
export default Home;
/* eslint-disable react/no-children-prop */
// @ts-nocheck
import React, { FC, useEffect, useState } from 'react';
import {

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						108
Изм.	Лист	№ докум.	Подпись	Дата		

```

    Alert,
    AlertDescription,
    AlertIcon,
    AlertTitle,
    Button, Input, InputGroup, InputLeftAddon,
    NumberDecrementStepper, NumberIncrementStepper, NumberInput,
    NumberInputField, NumberInputStepper, Radio, RadioGroup, Stack,
  } from '@chakra-ui/react';
import iconTester from '../image/icon_tester.png'
import styles from '../styles/index.module.css'
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>()

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						109
Изм.	Лист	№ докум.	Подпись	Дата		

```

const [ post, setPost ] = useState<string>>('')
const [ flagAlarmError, setFlagAlarmError ] =
useState<boolean>(false)

const hendaleCreateManagers = () => {
  if (!name || !floor || !age || !numberManager || !post) {
    setFlagAlarmError(true)
  } else {
    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('')
    setFlagAlarmError(false)
  }
}

const deleteCard = id => {
  const newManagers = managers.filter(e => e.id !== id)
  setManagers(newManagers)
  localStorage.setItem('managers', JSON.stringify(newManagers))
}

const alarmError = () => {
  if (flagAlarmError) {
    return (
      <Alert status="error">
        <AlertIcon />
        <AlertTitle>Вы не заполнили данные!</AlertTitle>

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <AlertDescription>Заполните все поля и повторите
попытку.</AlertDescription>

    </Alert>

    )

}

return null

}

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;

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {el.post}
      </p>
    </div>
    <div className={ styles.button_edit }>
      <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>
        <Input
          value={ name }
          onChange={ e => setName(e.target.value) }
        />
        <p>Пол</p>
        <RadioGroup value={ floor } onChange={ setFloor }>
          <Stack direction="row">
            <Radio value="Мужской">Мужской</Radio>
            <Radio value="Женский">Женский</Radio>
          </Stack>
        </RadioGroup>
        <p>Возраст</p>
        <NumberInput onChange={ setAge } min={ 0 } max={ 150 }>
          <NumberInputField />
          <NumberInputStepper>
            <NumberIncrementStepper />
            <NumberDecrementStepper />

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

```

        </NumberInputStepper>
      </NumberInput>
      <p>Номер телефона</p>
      <NumberInput onChange={ setNumberManager }>
        <NumberInputField />
        <NumberInputStepper>
          <NumberIncrementStepper />
          <NumberDecrementStepper />
        </NumberInputStepper>
      </NumberInput>
      <p>Должность</p>
      <Input
        value={ post }
        onChange={ e => setPost(e.target.value) }
      />
      { alarmError() }
      <Button colorScheme="blue" onClick={ () =>
hendaleCreateManagers() }>
        Добавить руководителя
      </Button>
    </div>
    <div className={ styles.wrapper_managers }>
      <strong>Руководители</strong>
      {renderManagers() }
    </div>
  </section>
</div>

);
}

export default Managers;

import React, { FC } from 'react';
import PropTypes from 'prop-types';
import styles from '../styles/index.module.css'
import frontDropped from '../image/front_dropped.png'
const NotFoundPage: FC = () => (

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						113
Изм.	Лист	№ докум.	Подпись	Дата		

```

    <div>
      <section className={ styles.wrapper_list_error }>
        <div className={ styles.wrapper_error }>
          <strong className={ styles.title_front_dropped
}>404</strong>
          <strong className={ styles.title_front_dropped }>Страница
не найдена!</strong>
          <img alt="frontDropped" src={ frontDropped } className={
styles.front_dropped } />
        </div>
      </section>
    </div>
)
export default NotFoundPage;

// @ts-nocheck
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,
  AlertIcon,
  AlertDescription,
  AlertTitle,
  Alert,
  RadioGroup,
  Stack,
  Radio,
} from '@chakra-ui/react';
import styles from '../styles/index.module.css'

const TaskPage: FC<Props> = observer(({
  taskQuestion = [],
  currentUser = {},
  taskPassedQuestion = [],

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						114
Изм.	Лист	№ докум.	Подпись	Дата		

```

    setTaskPassedQuestion,
  }) => {
    const [ flagAlarmError, setFlagAlarmError ] = useState(false)
    const [ category, setCategory ] = useState('')
    const [ questionList, setQuestionList ] =
useState<questionList>([])
    const [ succsesTestFlag, setSuccsesTestFlag ] = useState(false)
    const [ passedTask, setPassedTask ] = useState([])

    useEffect(() => {
      const localIdCategory =
Number(localStorage.getItem('idCategory'))

      if (taskQuestion) {
        const taskQuestionFilter = taskQuestion.find(el =>
el.idCategory === localIdCategory)
        setCategory(taskQuestionFilter?.category)
        setQuestionList(taskQuestionFilter?.questionList)
      }
    }, [])

    // questionList - массив без ответов(начальный), passedTask -
растущий массив уже с ответами

    const handlePassed = (valuePassed, idQuestion) => {
      const passedQuestion = questionList.find(el => el.idQuestion
=== idQuestion)// ищем объект в начальном массиве
      const idxPassedQuestion = questionList.findIndex(el =>
el.idQuestion === idQuestion)// ищем индекс оюъекта из начального
массива

      if (passedTask[idxPassedQuestion]?.answerUser) {
        const newPassedQuestion = passedTask.slice()
        newPassedQuestion[idxPassedQuestion] = {
          idQuestion: passedQuestion.idQuestion,
          question: passedQuestion.question,
          checkedAnswer: passedQuestion.checkedAnswer,
          answerUser: valuePassed,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						115
Изм.	Лист	№ докум.	Подпись	Дата		

```

        answerList: passedQuestion.answerList,
    }

    setPassedTask(newPassedQuestion)
} else {
    setPassedTask([ ...passedTask, {
        idQuestion: passedQuestion.idQuestion,
        question: passedQuestion.question,
        checkedAnswer: passedQuestion.checkedAnswer,
        answerUser: valuePassed,
        answerList: passedQuestion.answerList,
    }])
}
}

const hendaleEndTest = () => {
    setSuccsesTestFlag(true)
    const localIdCategory =
Number(localStorage.getItem('idCategory'))
    const date = new Date().getTime()

    if (questionList.length === passedTask.length) {
        setFlagAlarmError(false)
        setTaskPassedQuestion([ ...taskPassedQuestion, {
            idPassed: date,
            idCategory: localIdCategory,
            loginUser: currentUser.login,
            category,
            questionList: passedTask,
        },
    ])

    localStorage.setItem('taskPassedQuestion', JSON.stringify([
...taskPassedQuestion, {
            idPassed: date,
            idCategory: localIdCategory,

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

```

        loginUser: currentUser.login,
        category,
        questionList: passedTask,
    ]]))
} else {
    setFlagAlarmError(true)
}
}

const alarmError = () => {
    if (flagAlarmError) {
        return (
            <Alert status="error">
                <AlertIcon />
                <AlertTitle>Вы ответили не на все вопросы!</AlertTitle>
                <AlertDescription>Заполните все поля и повторите
попытку.</AlertDescription>
            </Alert>
        )
    }
    return null
}

const succsesTest = () => (
    <Alert
        status="success"
        variant="subtle"
        flexDirection="column"
        alignItems="center"
        justifyContent="center"
        textAlign="center"
        height="200px"
    >
        <AlertIcon boxSize="40px" mr={ 0 } />
        <AlertTitle mt={ 4 } mb={ 1 } fontSize="lg">
            Тест пройден!
        </AlertTitle>
    </Alert>
)

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						117
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <AlertDescription maxWidth="sm">
            Спасибо, через некоторое время с вами свяжутся для
            обратной связи.
        </AlertDescription>
    </Alert>
)

const renderItem = (answerListItem, idQuestion) => (
    answerListItem.map((el, index) => (
        <div className={ styles.radioBlock } key={ el.idAnswer }>
            <Radio className={ styles.radio } colorScheme="green"
value={ `${index + 1}` } onChange={ e =>
handlePassed(e.target.value, idQuestion) }>{el.answer}</Radio>
        </div>
    )))

const renderQuestion = () => (
    questionList?.map(el => (
        <div key={ el.idQuestion }>
            <strong>{el.question}</strong>
            <RadioGroup className={ styles.question_block }>
                <Stack>
                    { renderItem(el.answerList, el.idQuestion) }
                </Stack>
            </RadioGroup>
        </div>
    ))
)

const renderContent = () => {
    if (succsesTestFlag && !flagAlarmError) {
        return succsesTest()
    }
    return (
        <section className={ styles.wrapper_moveList }>
            { alarmError() }
            <div className={ styles.wrapper_posters }>

```

```

        <p className={ styles.wrapper_title }>{category}</p>
        <div className={ styles.wrapper_content }>
            { renderQuestion() }
        </div>
    </div>

    <Button colorScheme="yellow" onClick={ () =>
hendaleEndTest() }>
        Завершить тест
    </Button>
</section>

)
}

return (
    <div>
        {renderContent()}
    </div>
)
})

export default TaskPage;

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

import {
    Alert,
    AlertDescription,
    AlertIcon,
    AlertTitle,
    Button, Input, InputGroup, InputLeftAddon,
    NumberDecrementStepper, NumberIncrementStepper, NumberInput,
    NumberInputField, NumberInputStepper, Radio, RadioGroup, Stack,
} from '@chakra-ui/react';
import iconTester from '../..image/icon_tester.png'

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						119
Изм.	Лист	№ докум.	Подпись	Дата		

```

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

const Results: FC<Props> = ({ taskPassedQuestion }) => {
  const [ currentTest, setCurrentTest ] = useState([])

  const hendaleCurrentTest = idPassed => {
    const filterCurrentTest = taskPassedQuestion.find(el =>
el.idPassed === idPassed)
    setCurrentTest(filterCurrentTest)
  }

  const calcAnswer = questionList => {
    const questionRight = questionList.filter(el => el.answerUser
=== el.checkedAnswer)
    return `${questionRight.length} из ${questionList.length}`
  }

  const renderItem = (answerListItem, checkedAnswer) => (
    answerListItem.map((el, index) => (
      <div className={ styles.radioBlock } key={ el.idAnswer }>
        <Radio isDisabled className={ styles.radio }
colorScheme="green" value={ `${index + 1}` }>
          <p className={ checkedAnswer === `${index + 1}` ?
styles.color_text_green : styles.color_text_red }>{el.answer}</p>
        </Radio>
      </div>
    )))

  const renderTest = () => (
    <div className={ styles.wrapper_question_list }>
      <p className={ styles.wrapper_title }>{currentTest.category}</p>
      { currentTest.questionList?.map(el => (
        <div key={ el.idQuestion } className={
styles.wrapper_question_list }>
          <strong>{el.question}</strong>
          <RadioGroup defaultValue={ el.answerUser }>
            <Stack>
              { renderItem(el.answerList, el.checkedAnswer) }
            </Stack>
          </RadioGroup>
        </div>
      ))}
    </div>
  )
}

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						120
Изм.	Лист	№ докум.	Подпись	Дата		

```

        </Stack>
      </RadioGroup>
    </div>
  )) }
</div>
)

const renderUser = () => {
  if (taskPassedQuestion.length > 0) {
    return (
      taskPassedQuestion.map(el => (
        <div key={ el.idPassed } className={
styles.wrapper_user_card }>
          <img alt="user_logo" src={ iconTester } className={
styles.wrapper_user_logo } />
          <div>
            <strong>Логин:</strong>
            <p>{el.loginUser}</p>
          </div>
          <div>
            <strong>Категория:</strong>
            <p>{el.category}</p>
          </div>
          <div>
            <strong>Результат:</strong>
            <p>{calcAnswer(el.questionList)}</p>
          </div>
          <Button colorScheme="yellow" onClick={ () =>
hendaleCurrentTest(el.idPassed) }>
            Подробнее
          </Button>
        </div>
      ))
    )
  }
}

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						121
Изм.	Лист	№ докум.	Подпись	Дата		

```

    return <p>Никто пока не прошёл тест</p>
  }

  return (
    <div>
      <section className={ styles.wrapper_moveList }>
        <p className={ styles.wrapper_title }>Результаты</p>
        <div className={ styles.wrapper_content }>
          <div className={ styles.wrapper_content_user }>
            { renderUser() }
          </div>
          <div className={ styles.wrapper_content_test }>
            { renderTest() }
          </div>
        </div>
      </section>
    </div>
  );
}

export default Results;

import React, { FC, useEffect, useState } from 'react';
import { Button } from '@chakra-ui/react';
import { Link } from 'react-router-dom';
import iconTester from '../image/icon_tester.png'

import styles from '../styles/index.module.css'
interface Props {
  currentUser: {
    id: number,
    login: string,
    password: string,
  },
}

const UserPage: FC<Props> = ({ currentUser }) => (
  <div>

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						122
Изм.	Лист	№ докум.	Подпись	Дата		

```

    <section className={ styles.wrapper_usesr_page }>
      <div className={ styles.wrapper_user_card }>
        <div className={ styles.wrapper_user_info }>
          <img alt="user_logo" src={ iconTester } className={
styles.wrapper_user_logo } />
          <div className={ styles.fio_info }>
            <strong>Логин:</strong>
            <p>{currentUser.login}</p>
          </div>
          <div className={ styles.number_info }>
            <Link to="/">
              <Button className={ styles.btn_exit } size="sm"
colorScheme="red">Выход</Button>
            </Link>
          </div>
        </div>
      </div>
    </section>
  </div>
)
export default UserPage;

```

					ПГУ 1.090501.04.001 ПЗ	Лист
						123
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Г

(обязательное)

Программа для интерактивного тестирования *IT* специалистов

Презентация

					ПГУ 1.090501.04.001 ПЗ	Лист
						124
Изм.	Лист	№ докум.	Подпись	Дата		



Программа для интерактивного тестирования IT специалистов

Автор работы:
студент гр. 18ВОЭ1 Исаева Виктория Евгеньевна
Научный руководитель:
к.т.н., доцент Слепцов Николай Владимирович

Пенза, 2023

					ПГУ 1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

Цель работы и объекты

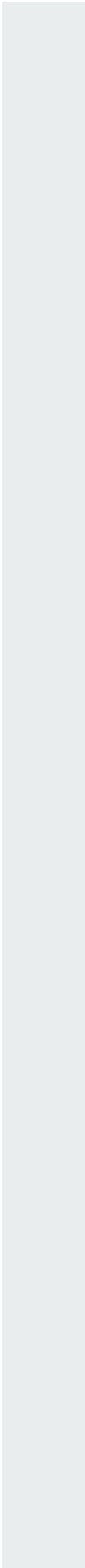
Цель работы - проведение объектного тестирования кандидатов на должность IT специалистов.

Объектом проекта является программа, выполняющая следующие функции:

- Работа в режиме кандидата (тестируемого)
- Работа в режиме сотрудника HR отдела
- Работа в режиме создания теста
- Работа в режиме прохождения тестирования

Задачи программы

- Регистрация и авторизация пользователей в системе
- Возможность формирования и загрузка тестов
- Проведение пользователям тестирования
- Фиксация результатов проведения тестирования
- Возможность просмотра результатов тестирования
- Формирование обратной связи для тестируемого



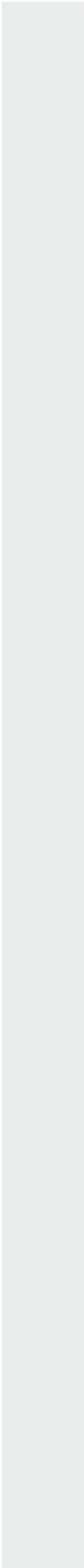
Сравнение аналогов



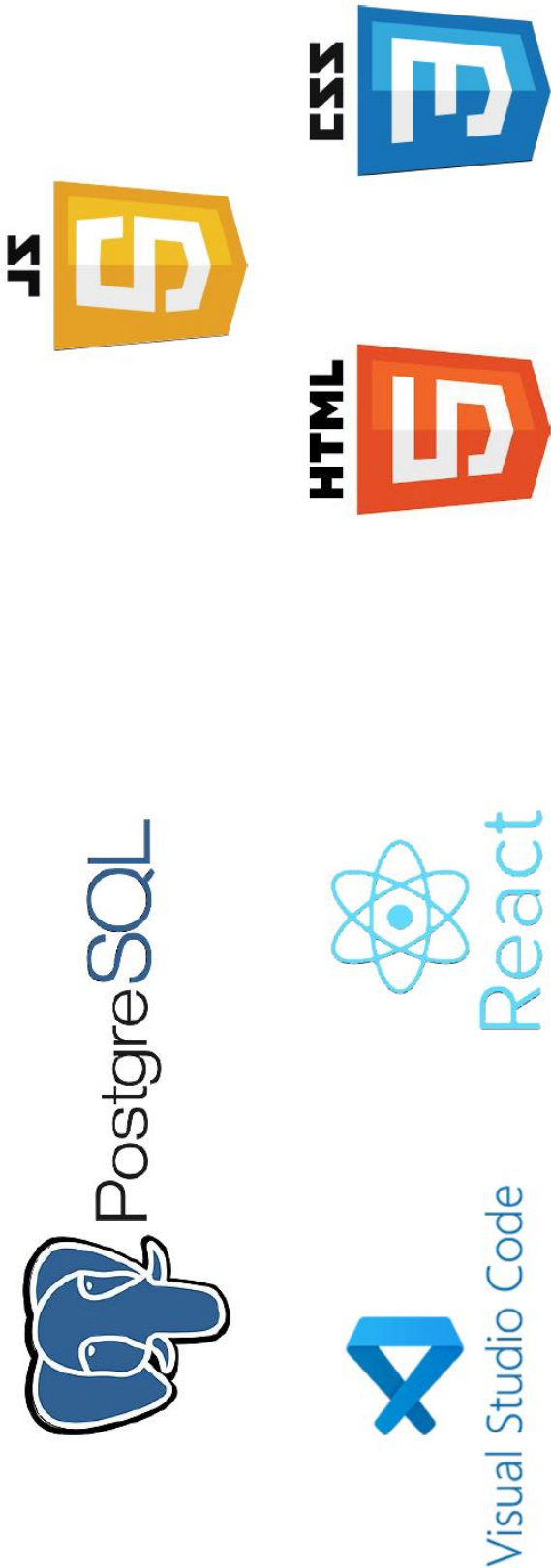
Характеристика	HackerRank	Codility	CoderPad	CodinGame	DevSkiller
Открытый исходный код	-	-	-	-	-
Формирование и загрузка собственных тестов	+	+	+	+	+
Простота технической поддержки	-	+	+	+	+
Реализация на бесплатном и свободно распространяемом инструментарии	+	-	-	+	-

Аналоги обладают недостатками:

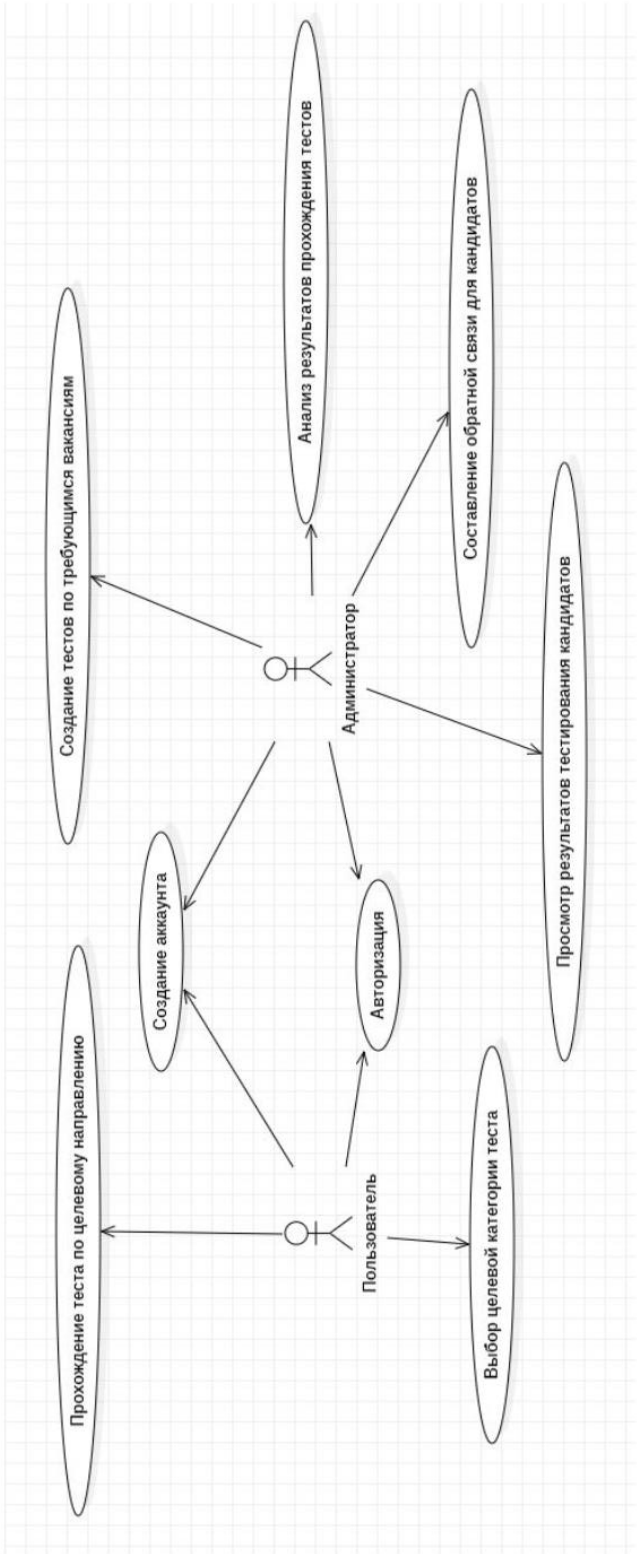
отсутствие открытого исходного кода, а также аналоги реализованы в основном на платном инструментарии.



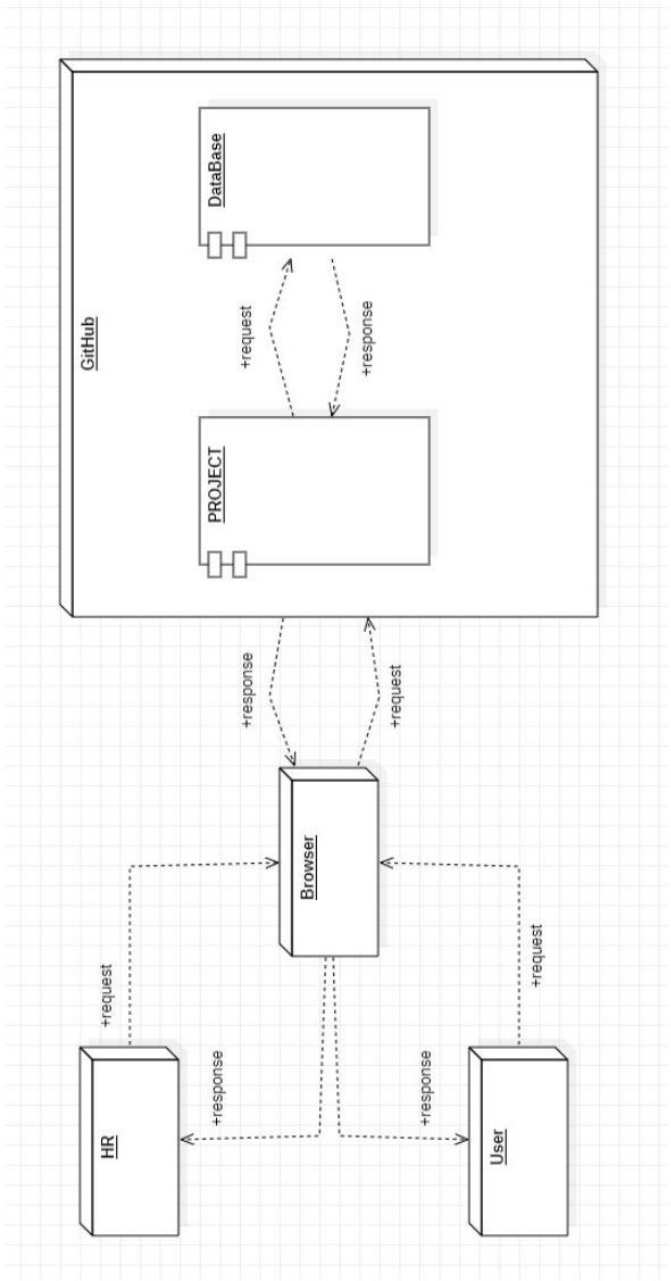
Стек технологий



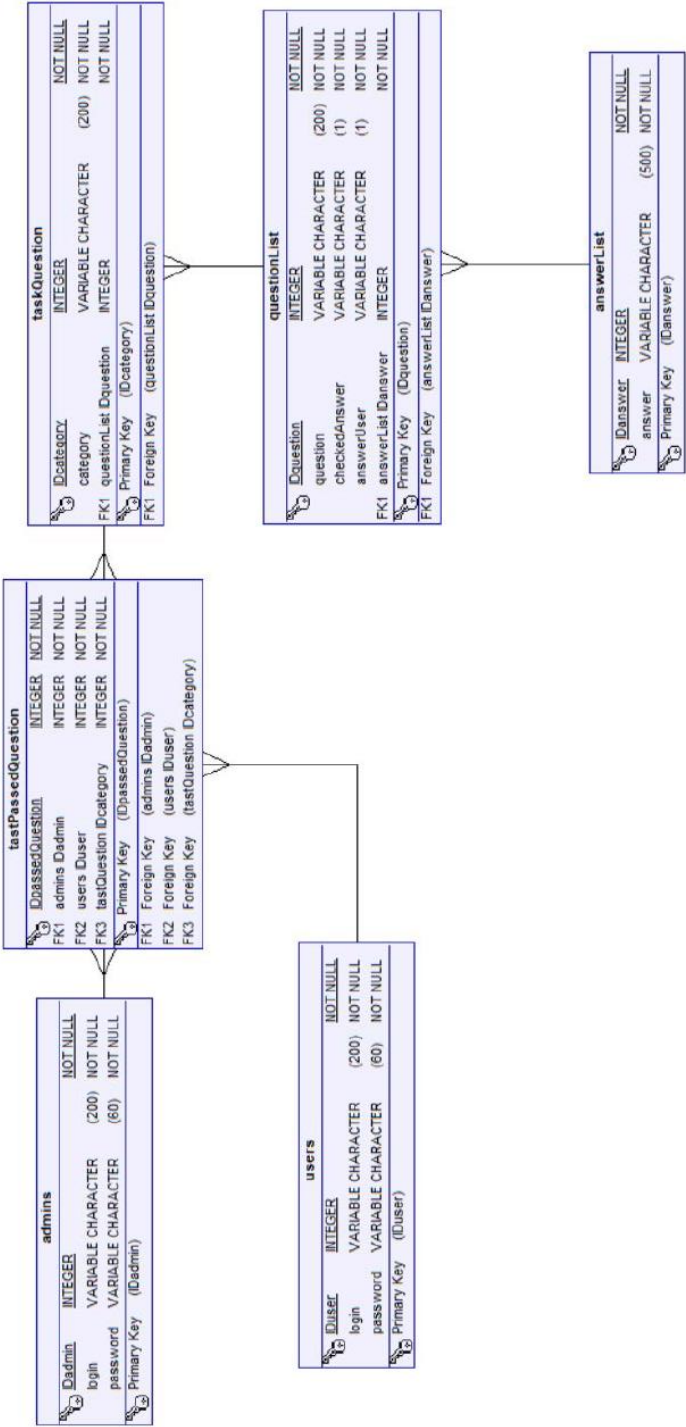
Основные функции программы



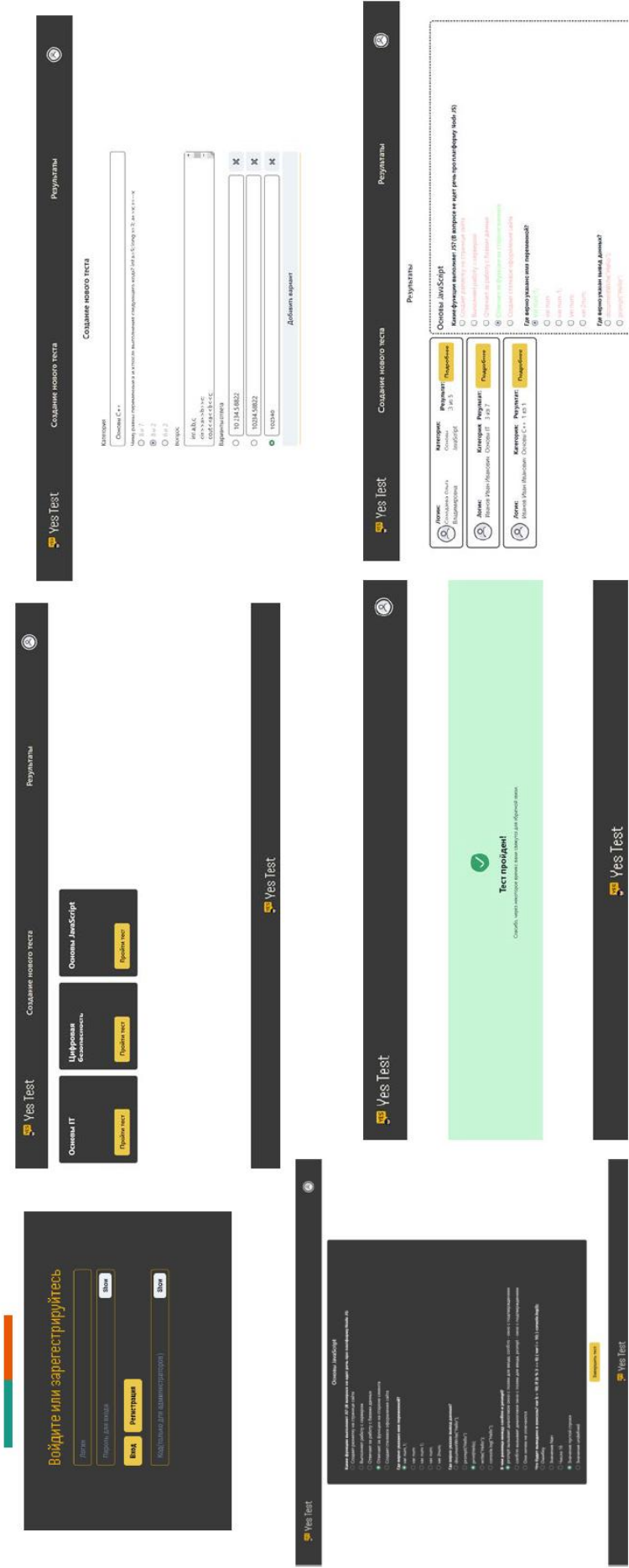
Архитектура программы



База данных программы



Скриншоты запущенной программы



Результаты работы

- Разработана программа, полностью соответствующая техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!

студент гр. 18ВОЭ1 Исаева Виктория Евгеньевна

					ПГУ 1.090501.04.001 ПЗ	Лист
						135
Изм.	Лист	№ докум.	Подпись	Дата		