

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
Факультет среднего профессионального образования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ДИПЛОМНОМУ ПРОЕКТУ

по специальности 09.02.07 «Информационные системы и программирование»

на тему:

«Разработка образовательного веб-ресурса по эпохе Петра I»

Студент

Панаёт

Панаёт Р.Т., У2434

Руководитель



Коцюба И. Ю., к.т.н.,
доцент ф-та ТМИ
Университета ИТМО

Защиту дипломного проекта разрешаю:

Директор факультета СПО Университета ИТМО

Зленко А.Н.

«07» июня 2022г.

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
ВВЕДЕНИЕ.....	3
1. АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ	4
1.1 Формулировка поставленной задачи.....	4
1.2 Описание предметной области.....	4
1.3 Обзор и сравнение аналогичных решений.....	6
1.4 Функциональные требования к разработке.....	17
2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ.....	18
2.1 Описание входных и выходных данных	18
2.2 Моделирование приложения	21
2.3 Выбор архитектуры системы.....	23
2.3.1 Описание клиент-серверных архитектур.....	23
2.3.2 Описание микросервисной архитектуры	28
2.3.3 Выбор архитектуры приложения.....	31
2.4 Обоснование выбора программных средств	32
2.4.1 Выбор средств проектирования.....	32
2.4.2 Обоснование модели данных	33
2.4.3 Выбор СУБД.....	34
2.4.4 Выбор средств реализации серверного приложения	35
2.4.5 Выбор средств реализации клиентского приложения	37
2.4.6 Выбранные технологии и программы	37
2.5 Выбор методов тестирования.....	38
2.6 Программная реализация	40
3. ЭКОНОМИЧЕСКАЯ ОЦЕНКА РАЗРАБОТКИ.....	42
3.1 Расчёт стоимости разработки образовательного веб-ресурса по эпохе Петра I	42
3.2 Выводы по результатам экономической оценки	45
ЗАКЛЮЧЕНИЕ	46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	47
ПРИЛОЖЕНИЕ А.....	51
ПРИЛОЖЕНИЕ Б.....	64

ВВЕДЕНИЕ

Целью дипломного проекта является разработка образовательного веб-ресурса по эпохе Петра I.

Для разработки системы требуется выполнить следующие задачи:

- проанализировать предметную область;
- сформировать требования к системе;
- определить входные и выходные данные;
- выбрать и обосновать средства реализации;
- определить архитектуру приложения;
- произвести детальное проектирование;
- выбрать методы тестирования;
- сформировать техническое задание;
- разработать серверную часть веб-ресурса;
- разработать клиентскую часть веб-ресурса;
- протестировать веб-ресурс.

Пояснительная записка содержит следующие разделы:

1. Введение.
2. Раздел 1 Анализ поставленной задачи.
3. Раздел 2 Проектирование и программная реализация системы.
4. Раздел 3 Экономическая оценка разработка.
5. Заключение.
6. Список источников.
7. Приложение А Техническое задание.
8. Приложение Б Модели разработки.

1. АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

1.1 Формулировка поставленной задачи

Целью дипломного проекта является проектирование и разработка обучающего веб-ресурса по эпохе Петра I, который будет обеспечивать пользователей возможностью проведения тестирования.

Основной целевой аудиторией являются преподаватели истории и обучающиеся, а также люди, желающие ознакомиться с личностью Петра I.

1.2 Описание предметной области

Изучение истории и дисциплин социально-гуманитарного профиля — это одно из самых важных и базовых направлений учебно-образовательных учреждений. История является дисциплиной, способной эффективно развивать критическое мышление и причинно-следственные связи у учащихся.

При условиях цифровизации и информатизации общества преподавание дисциплин стало меняться. В преподавании стали использоваться современные способы получения информации, которые облегчают процесс обучения. История является одной из таких дисциплин. При её изучении могут использоваться следующие типы ресурсов: информационные, методические, графические и интерактивные, которые можно использовать для проверки знаний учащихся.

Стоит отметить, что при большом количестве таких обучающих ресурсов, не всегда учтена специфика конкретной исторической эпохи. Изучение конкретной личности в эпохе, учитывая социально-экономические и политические устои того времени, требует более детального внимания. Примером такой личности может послужить Пётр I - первый Всероссийский император и основатель города Санкт-Петербург. Его персона является неоднозначной в исторической науке и по сей день изучается в Санкт-Петербургских учебных заведениях. Частным примером проекта, связанного с

личностью Петра I, может послужить Междисциплинарный научно-просветительский проект «PETRO pr(i[t]mo) научно-техническое наследие петровской эпохи», который существует на базе Университета ИТМО ФТМИ.

Целью проекта является формирование и развитие ключевых компетенций (далее - КК) студентов, заложенных в рабочую программу дисциплины общеобразовательного модуля(далее – РПД ОМ) «История» в формате проектной работы, направленной на исследование влияния достижений науки и техники петровской эпохи на последующий ход научно-технического развития и культурного процесса.

У проекта есть следующие задачи:

- формирование и развитие, знаний, умений и навыков, заложенных в КК РПД ОМ «История»;
- создание контента и размещение его в формате интеллектуальных карт на выбранном ресурсе;
- овладение более глубокими знаниями по ключевым проблемам истории науки и техники новейшего времени;
- формирование у студентов навыков оценивания работы команды и индивидуальной работы;
- формирование Soft Skills, необходимых в проектной деятельности;
- создание итогового продукта по результатам работы в проекте каждой команды.

Актуальность проекта выражена в следующих пунктах:

- востребованность проектной работы, позволяющей более глубоко изучить материал, работать в командах и реализовать нетривиальные подходы к образовательной деятельности;
- стремление к самостоятельному исследованию и выполнению творческих заданий, позволяющих реализовать их профессиональные компетенции в гуманитарной области;

- востребованность в представлении результатов своей работы широкой аудитории;
- создание проекта в смешанном формате (с использованием цифровых технологий) в соответствии с запросом времени.

Исходя из описания проекта, можно утверждать, что для его успешного выполнения необходимы информационные, методические, графические и интерактивные электронные ресурсы. Для комплексной проверки знаний в интерактивных ресурсах следует использовать различные педагогические методики тестирования.

Таким образом, можно заметить, что существует проблема погружения в эпоху Петра I в соответствии с требованиями заказчика, а с другой стороны, существует потребность в использовании методического материала для проведения педагогических тестирований. На текущий момент решение такой проблемы отсутствует, тем самым доказана актуальность данного веб-ресурса. Его разработка позволит улучшить процесс обучения учащихся эпохе Петра I и создать единое информационное пространство, в котором будут взаимодействовать педагог и студент.

1.3 Обзор и сравнение аналогичных решений

Для выявления актуальности проектирования и необходимых функций проведено сравнение аналогичных решений.

В результате поиска аналогичных решений были выбраны следующие аналоги.

1. LearningApps – прямой аналог . Веб-ресурс для создания и прохождения небольших упражнений. На рисунке 1 представлена главная страница. На рисунке 2 можно увидеть страницу выбора упражнения. А на рисунке 3 показан процесс создания упражнения.

Достоинства ресурса:

- множество шаблонов для создания упражнений;

- классификация упражнений по уровню сложности;
- отсутствие рекламы.

Недостатки ресурса:

- плохая локализация на русский язык;
- устаревший дизайн.

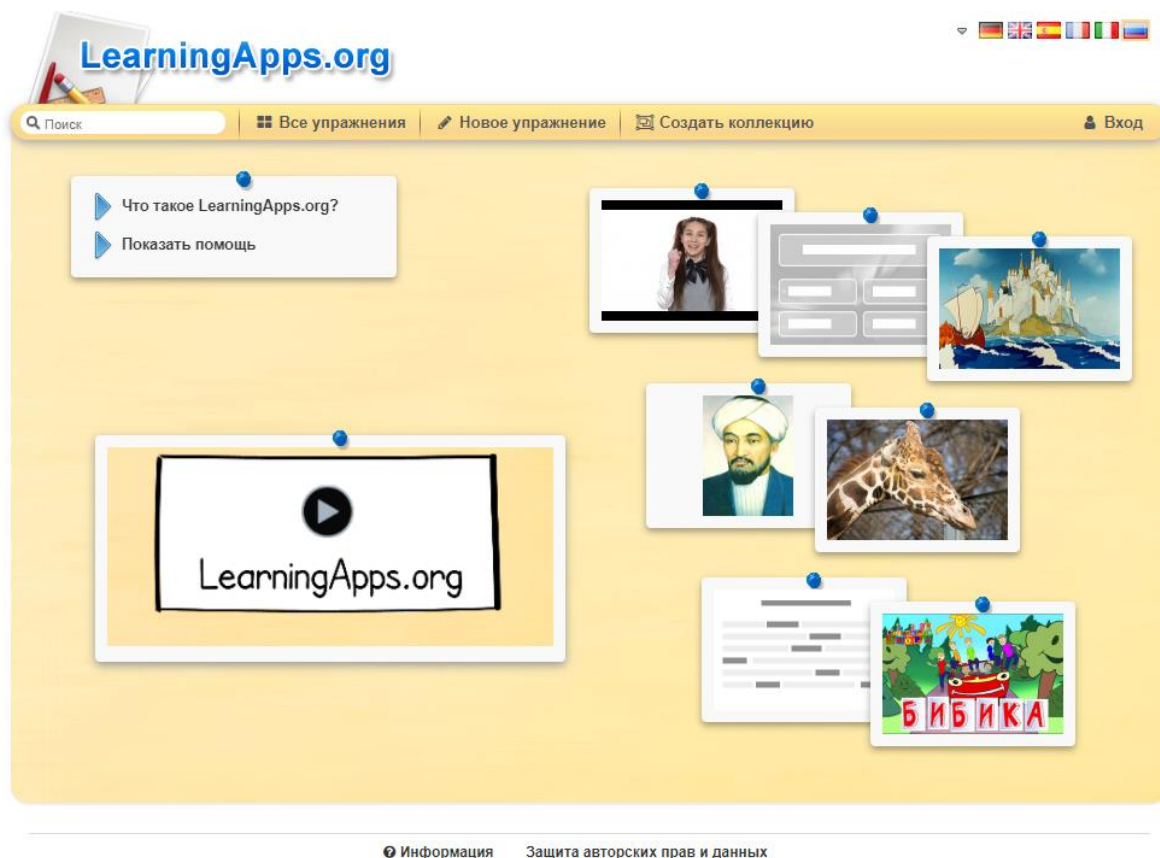


Рисунок 1 – Интерфейс «LearningApps»: главная страница

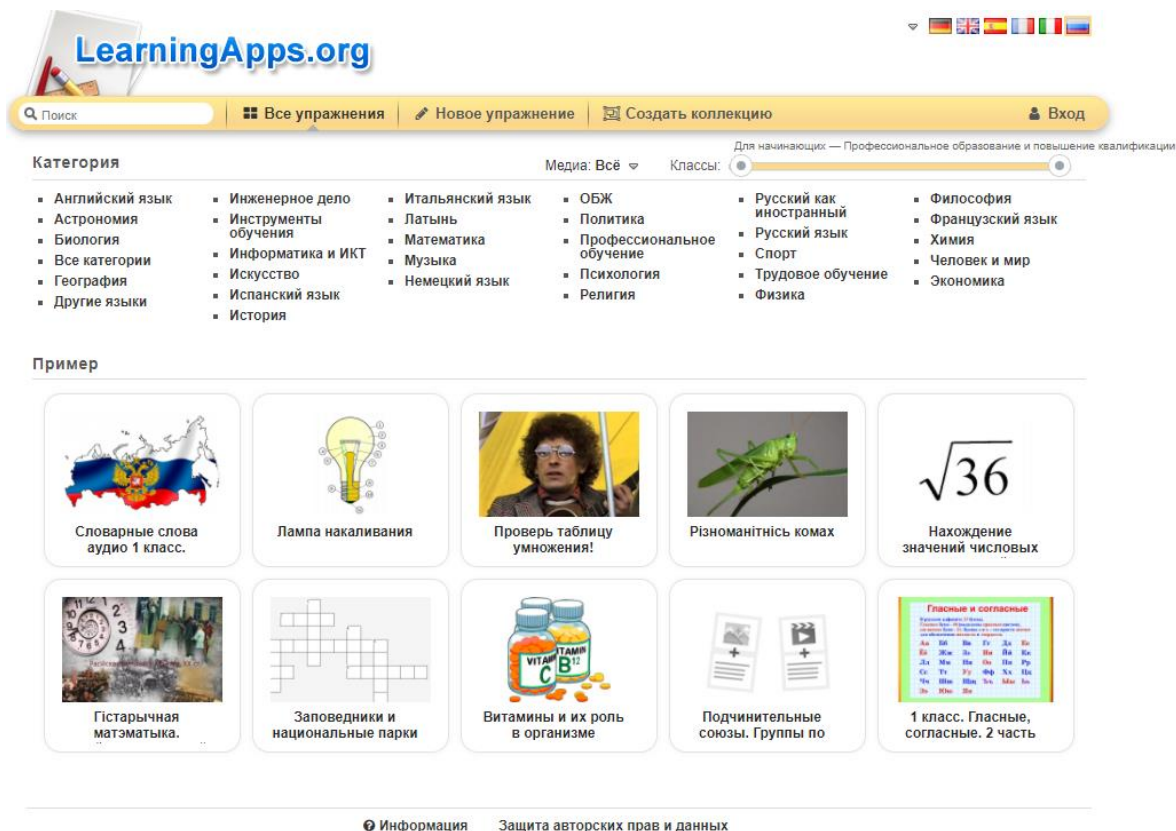


Рисунок 2 – Интерфейс «LearningApps»: все упражнения



Рисунок 3 – Интерфейс «LearningApps»: новое упражнение

2. PenCup – прямой аналог. Веб-сервис, предоставляющий пользователям возможность для создания и прохождения тестов. На рисунке 4 показана главная страница. Рисунок 5 изображает процесс прохождения теста. А рисунок 6 иллюстрирует результаты прохождения теста. У рисунка 7 задача такова – показать процесс создания теста.

Достоинства ресурса:

- возможность оценивания тестов;
- классификация тестов по уровню сложности.

Недостатки ресурса:

- много рекламных баннеров, мешающих просмотру контента.

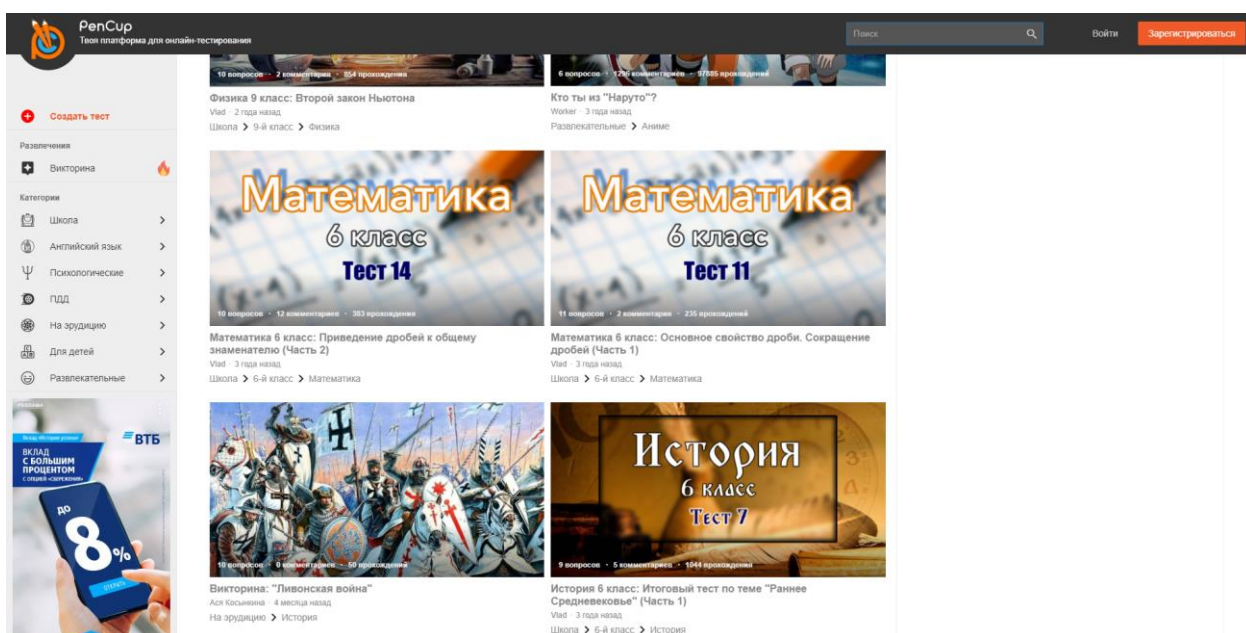


Рисунок 4 – Интерфейс «PenCup»: Главная страница

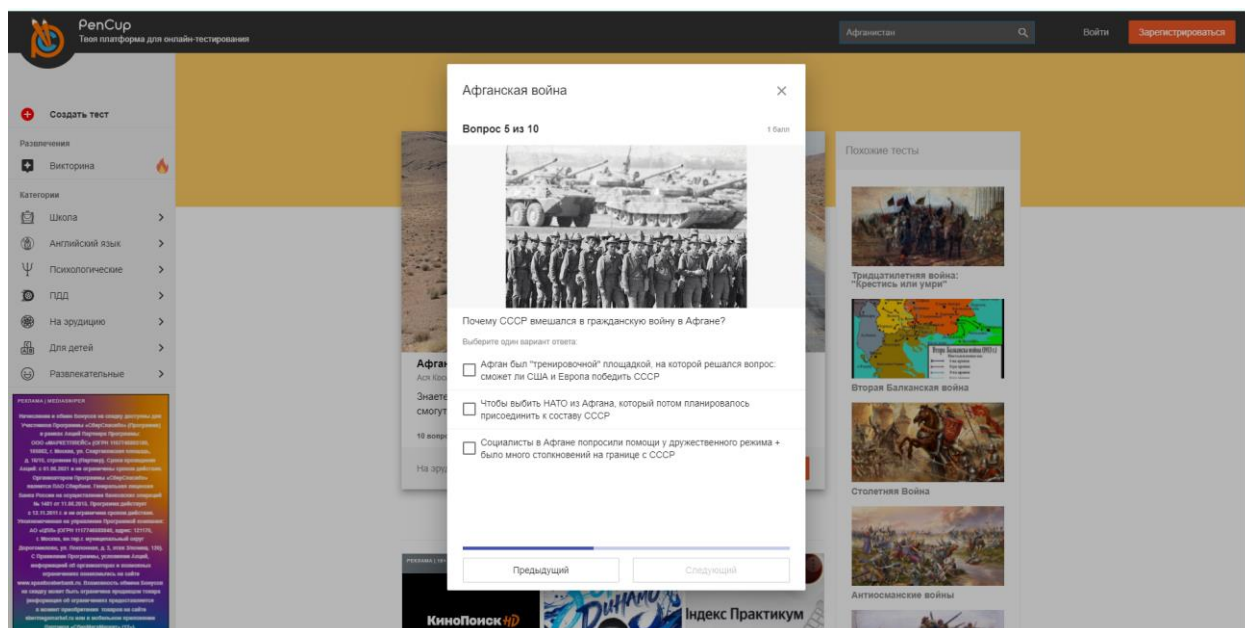


Рисунок 5 – Интерфейс «PenCup»: Процесс прохождения теста

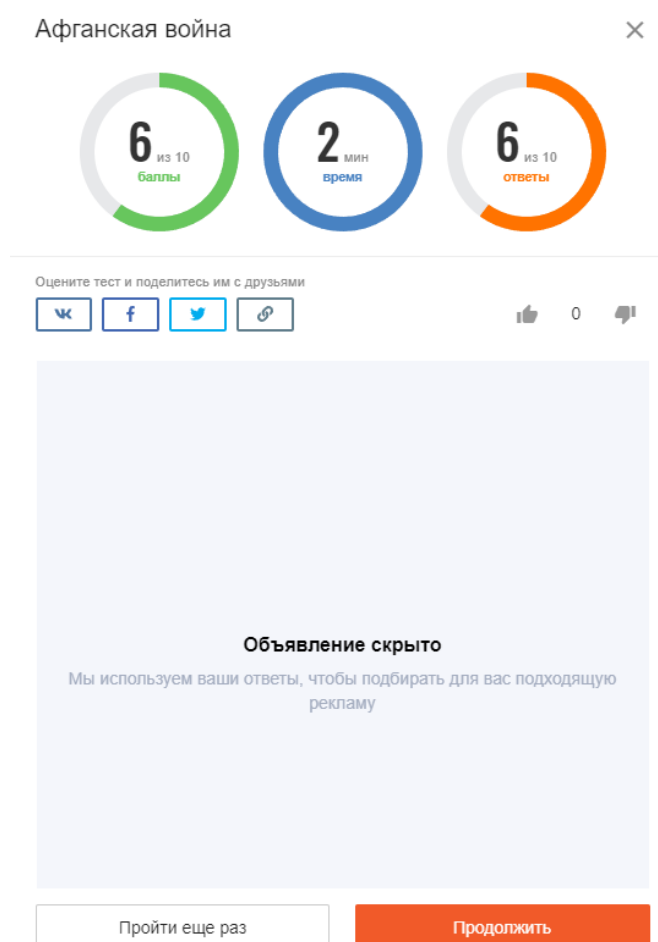



Рисунок 6 – Интерфейс «PenCup»: Результат прохождения теста

Создание нового теста

✕



Загрузить титульное изображение для теста

Название теста *

Краткое описание теста *

//

Категория *

▼

Источник

Название книги, учебника или ссылка на оригинал

Создать

Рисунок 7 – Интерфейс «РепСир»: Создание теста

3. Udoba – прямой аналог. Веб-сервис, представляющий собой конструктор и хостинг открытых интерактивных электронных образовательных ресурсов. На рисунке 8 представлена главная страница. А на рисунке 9 показан процесс создания интерактивного контента.

Достоинства ресурса:

- множество шаблонов для создания упражнений;
- отсутствие рекламы.

Недостатки ресурса:

- отсутствует возможность оценивания;
- неудачный дизайн.

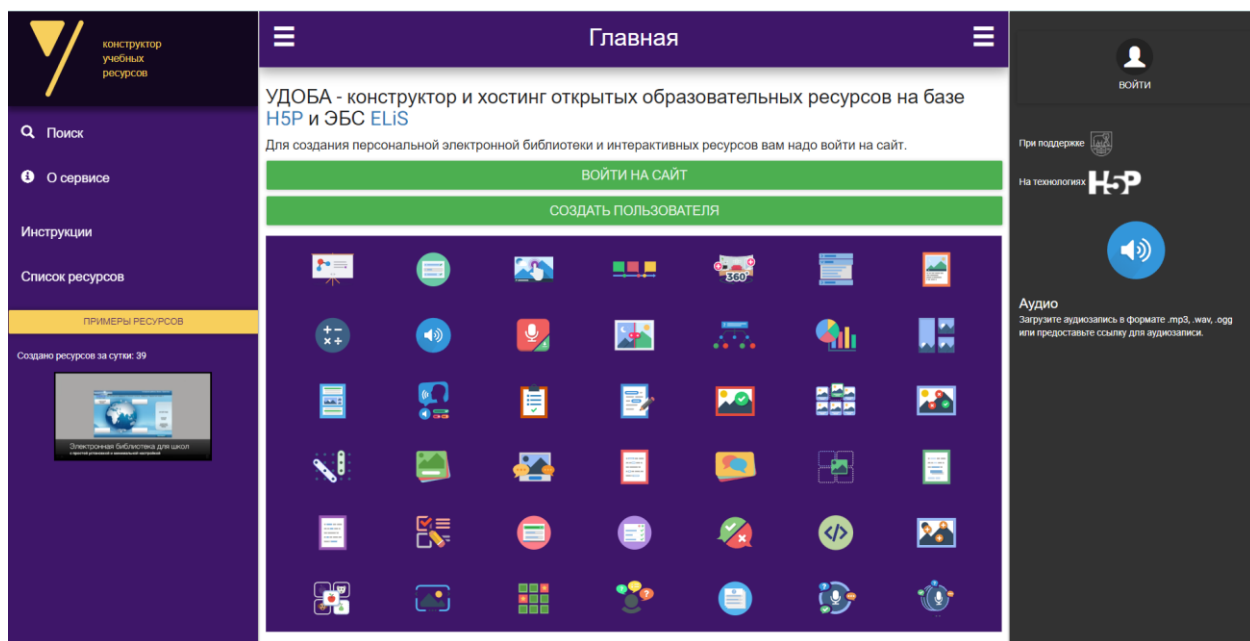


Рисунок 8 – Интерфейс «Udoba»: Главная страница

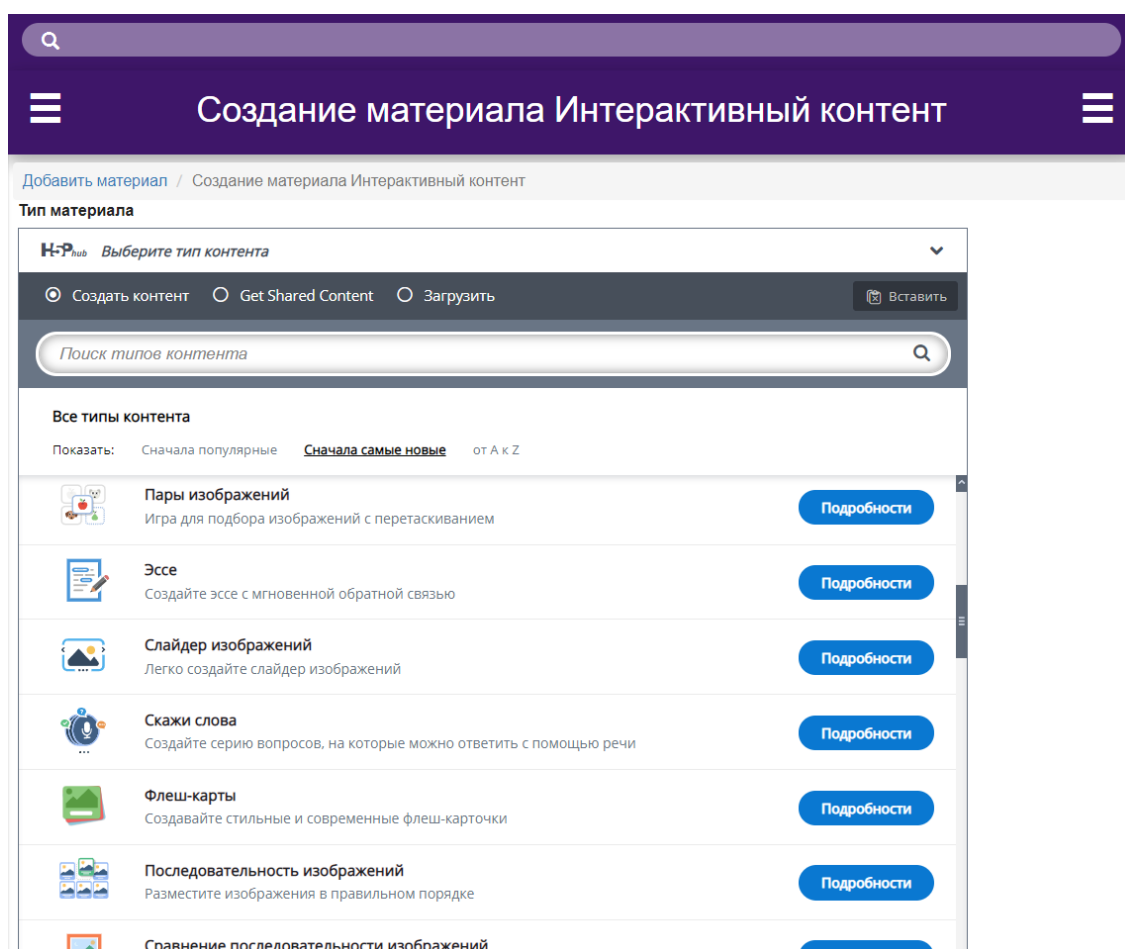


Рисунок 9 – Интерфейс «Udoba»: Создание материала

4. TestServer – прямой аналог. Веб-сервис, предоставляющий пользователям возможность для создания и прохождения тестов. Главная страница представлена на рисунке 10. Каталог тестов – на рисунке 11. Процесс прохождения теста показан на рисунке 12. И на 13 рисунке изображен процесс создания теста.

Достоинства ресурса:

- множество шаблонов для создания тестов;
- есть обучающий материал по созданию тестов.

Недостатки ресурса:

- реклама;
- неудобный интерфейс для создания тестов;
- вопросы без изображений.

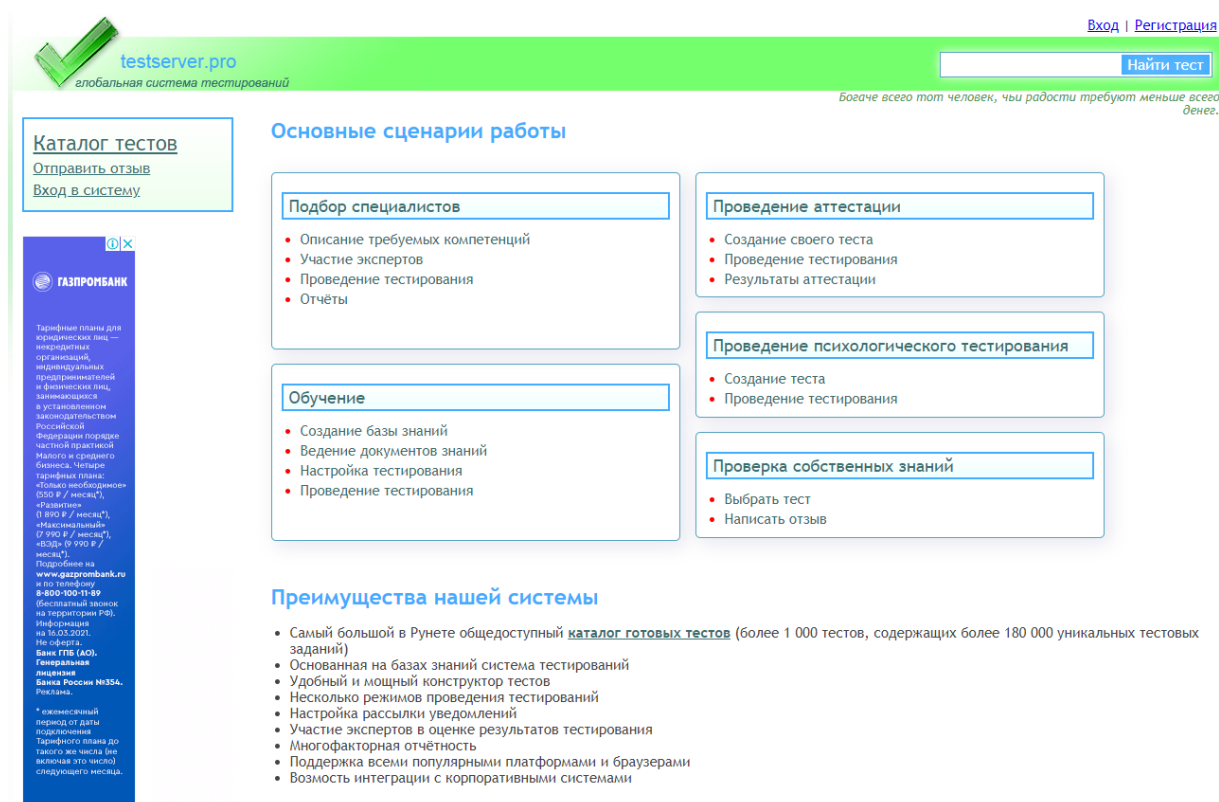


Рисунок 10 – Интерфейс «TestServer»: главная страница

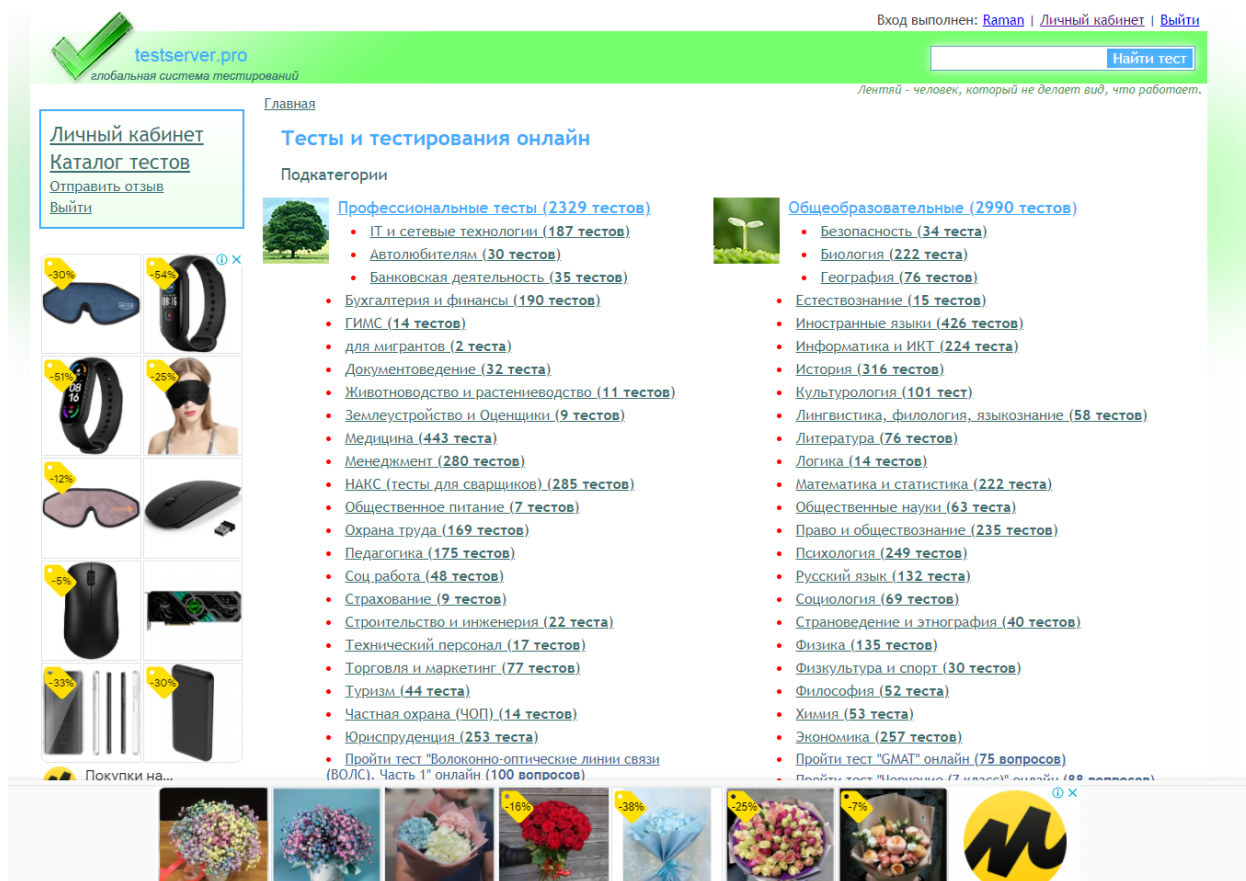



Рисунок 11 – Интерфейс «TestServer»: каталог тестов



Рисунок 12 – Интерфейс «TestServer»: прохождение теста




testserver.pro
глобальная система тестирования

Вход выполнен: [Raman](#) | [Личный кабинет](#) | [Выйти](#)

[Найти тест](#)

[Главная](#) / [Личный кабинет](#) / [Мои тестирования](#)

[Личный кабинет](#)
[Каталог тестов](#)
[Отправить отзыв](#)
[Выйти](#)



Тестирование

Крутой тест

Тестирование

Список тестов в тестировании:

Наименование теста	Порядок	Один вариант
нет данных		

0 из 0

Наименование *:

Крутой тест

от: 2021-10-11 20:19:41.612

Настройки тестирования:

☐ не показывать результаты тестируемому
 ☒ отправлять результаты тестируемому

дополнительно отправлять результаты на:

Вид тестирования:

☐ закрытое тестирование
 ☒ открытое тестирование

Открытое тестирование

имя группы контактов	создано	ссылка
Новая группа	11.10.2021 20:19	

1 из 1

Действия:

[Создать](#)
[Просмотр результатов](#)

Рисунок 13 – Интерфейс «TestServer»: создание теста

Далее представлена таблица 1. На ней можно увидеть сравнение аналогов по следующим критериям: создание тестов, оценивание теста, личный кабинет, сортировка тестов и фильтрация тестов.

Таблица 1 – Сравнение аналогов

Наименование аналогов → Функция ↓	LearningApps	PenCup	Udoba	TestServer
Создание тестов	Можно без регистрации и авторизации	При регистрации и авторизации открывается такая возможность	При регистрации и авторизации открывается такая возможность	При регистрации и авторизации открывается такая возможность
Оценивание теста	Есть, в виде от 1 до 5 звездочек	Есть, в виде лайка / дизлайка	Отсутствует	Отсутствует
Личный кабинет(профиль)	Отсутствует	Присутствует	Да	Присутствует
Сортировка тестов	Присутствует	Присутствует	Отсутствует	Отсутствует
Фильтрация тестов	Присутствует	Присутствует	Отсутствует	Отсутствует

Во всех аналогах присутствуют: регистрация, прохождение тестов и вывод результата тестирования. Во всех аналогах отсутствует создание теста по методикам.

Таким образом, в результате анализа аналогичных решений выделены следующие необходимые функциональные возможности для веб-сервиса:

- регистрация и авторизация;
- управление (создание по методикам, редактирование, удаление) тестами;
- прохождение тестов;
- вывод результата тестирования;
- сортировка тестов;

- фильтрация тестов;
- личный профиль;
- оценивание теста.

1.4 Функциональные требования к разработке

В Системе должны быть определены две категории пользователей:

1. Студент.
2. Преподаватель.

Функционал, предоставляемый пользователям категории Студент:

- создание/редактирование личного аккаунта (регистрация в системе);
- авторизация;
- просмотр данных личного аккаунта;
- просмотр результатов тестирования;
- добавление и редактирование ответов к вопросам теста;
- оценивание теста;
- прохождение тестов.

Преподаватель обладает функционалом Студента, но имеет дополнительную функцию - управление (создание/редактирование/удаление) тестами.

2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ

2.1 Описание входных и выходных данных

По результатам рассмотрения предметной области разрабатываемой Системы был составлен перечень входных и выходных данных.

Входные данные:

Для пользователя категории Студент:

- данные личного аккаунта (при создании или редактировании):
 - логин;
 - пароль;
 - адрес электронной почты;
 - ФИО;
 - роль;
 - аватар-изображение.
- данные ответа (при создании или редактировании) - ответ.
- данные теста - оценка теста.

Для пользователей категории Преподаватель:

- перечисленные входные данные для категории Студент;
- данные теста (при создании):
 - название;
 - тип;
 - описание;
 - картинка.
- данные вопроса(ов) к тесту (при создании):
 - вопрос;
 - картинка;
 - категория вопроса;
 - сложность вопроса;
 - тип вопроса.

- данные ответа(ов) к тесту (при создании):
 - ответ;
 - правильность ответ.
- данные результата(ов) к тесту (при создании):
 - заголовок;
 - описание;
 - картинка;
 - условие получения;
 - успешно ли результат заканчивает тест;
 - правильность ответ.

Выходные данные:

Для пользователя категории Студент:

- личные данные аккаунта пользователя:
 - логин;
 - адрес электронной почты;
 - ФИО;
 - аватар-изображение;
 - роль;
 - кол-во пройденных тестов;
 - дата регистрации.
- результаты прохождения теста:
 - выполнен ли тест;
 - оценка теста;
 - ответы пользователя;
 - дата прохождения теста;
 - процент правильных ответов.
- данные теста:
 - название;
 - тип;

- кол-во вопросов;
- дата создания;
- создатель теста;
- оценка теста;
- описание;
- картинка.

Для пользователей категории Преподаватель:

- перечисленные входные данные для категории Студент;
- ответы пользователей:
 - тест;
 - вопросы;
 - ответы;
 - правильность ответов.
- данные вопроса(ов) к тесту:
 - вопрос;
 - картинка;
 - тип вопроса.
- данные ответа(ов) к тесту:
 - ответ;
 - правильность ответа.
- данные результата(ов) к тесту:
 - заголовок;
 - описание;
 - картинка;
 - условие получения;
 - успешно ли результат заканчивает тест.

2.2 Моделирование приложения

Для построения моделей бизнес-процессов и описания бизнес-процессов используются методологии SADT, семейства IDEF, DFD, UML, ARIS и другие.

При проектировании и моделировании системы были использованы следующие средства:

1. IDEF0 [1] – нотация, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.
2. IDEF3 [2] используется для сбора информации о состоянии моделируемой системы. Это структурный метод, показывающий причинно-следственные связи и события. Он также показывает, как организована работа и какие пользователи работают с моделируемой системой. IDEF3 состоит из двух методов:
 - a. Process Flow Description (PFD) – описание процессов, с описанием того, как организована работа между различными элементами моделируемой системы.
 - b. Object State Transition Description (OSTD) – описание переходов состояний объектов, с описанием того, какие существуют промежуточные состояния у объектов в моделируемой системе.
3. DFD [3] – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.
4. UML [4] – общецелевой язык визуального моделирования, разработанный для спецификации, визуализации,

проектирования и документирования компонентов программного обеспечения и бизнес-процессов.

5. Метод «сущность-связь» [5] – методология моделирования предметной области, применяемая для создания инфологической модели базы данных.
6. Метод нормальных форм [6] – методология моделирования реляционной базы данных, обеспечивающая процесс последовательного перевода отношений из первой нормальной формы в нормальные формы более высокого порядка.

На основании функциональных требований (раздел 1.4) и описания входных и выходных данных (раздел 2.1) Системы была составлена функциональная модель IDEF0.

На уровне А-0 (рисунок Б.1) представлены главный процесс «проведение тестирования», входные и выходные данные, ограничения процесса и используемые механизмы. На уровне А0 (рисунок Б.2) производится детализация главного процесса с разбиением на подпроцессы: составление теста (для Преподавателя), прохождение теста, анализирование ответа, вывод результатов.

Уровень А3.1 (рисунок Б.3), реализован в нотации IDEF3 и представляет собой детализацию процесса анализирования ответов.

Уровень А4.1 (рисунок Б.4), реализован в нотации DFD и описывает потоки данных и хранилища данных, к которым осуществляется доступ в процессе прохождения теста.

Для чёткого разграничения и структуризации всех возможных действий пользователей в Системе при работе через клиентское приложение была составлена диаграмма вариантов использования, представленная на рисунке Б.5. Основанием для составления диаграммы также являются функциональные требования, представленные в разделе 2.1.

Для определения структуры разрабатываемой базы данных Системы была составлена инфологическая модель с использованием метода нормальных форм. Схема инфологической модели представлена на рисунке Б.6.

Основными сущностями БД являются:

- User – пользователи;
- Test – тесты;
- UsersTests – тесты, выбранные пользователем для прохождения;
- Question – вопросы тестов;
- Answer – ответы на вопросы;
- UsersAnswers – ответы пользователей на вопросы тестов;
- Result – результаты тестов;
- UsersResults – результаты прохождения тестов пользователей.

Далее инфологическая модель была преобразована в логическую модель, представленную в таблицах Б.1 – Б.8.

Поведение клиентского и серверного приложения при действиях пользователя должны соответствовать диаграмме активности на рисунке Б.7.

Разбиение и связь структурных компонентов приложения должны соответствовать диаграмме развертывания на рисунке Б.8.

2.3 Выбор архитектуры системы

2.3.1 Описание клиент-серверных архитектур

Представленная на рисунке 14 архитектура «Клиент-Сервер» предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.

В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервер.

Клиентская машина предоставляет пользователю так называемый «дружественный интерфейс», чтобы облегчить его взаимодействие с сервером.

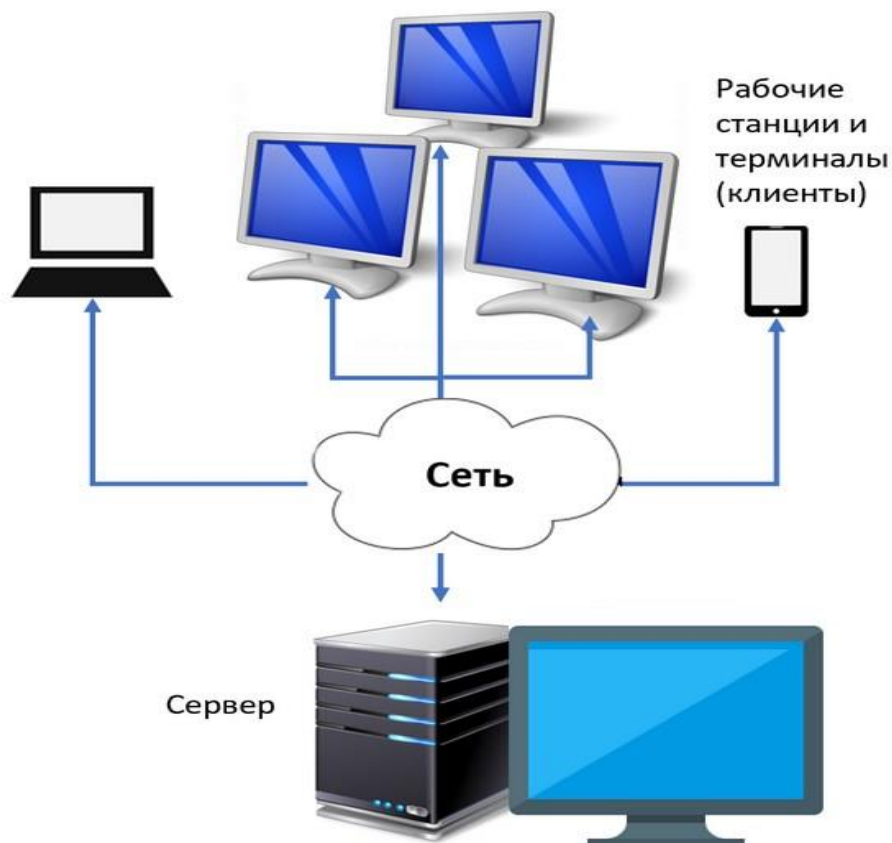


Рисунок 14 – Архитектура клиент-сервер

Архитектуру «клиент-сервер» принято разделять на три класса: одно-, двух- и трёхуровневую. Рассмотрим их подробнее:

1. Одноуровневая архитектура

Одноуровневая архитектура «клиент-сервер» (1-Tier) – такая, где все прикладные программы рассредоточены по рабочим станциям, которые обращаются к общему серверу баз данных или к общему файловому серверу. Её схему можно увидеть на рисунке 15. При

использовании такой архитектуры никаких прикладных программ сервер при этом не исполняет, только предоставляет данные. В целом такая архитектура очень надёжна, однако, ей сложно управлять, поскольку в каждой рабочей станции данные будут присутствовать в разных вариантах. Поэтому возникает проблема их синхронизации на отдельных машинах.

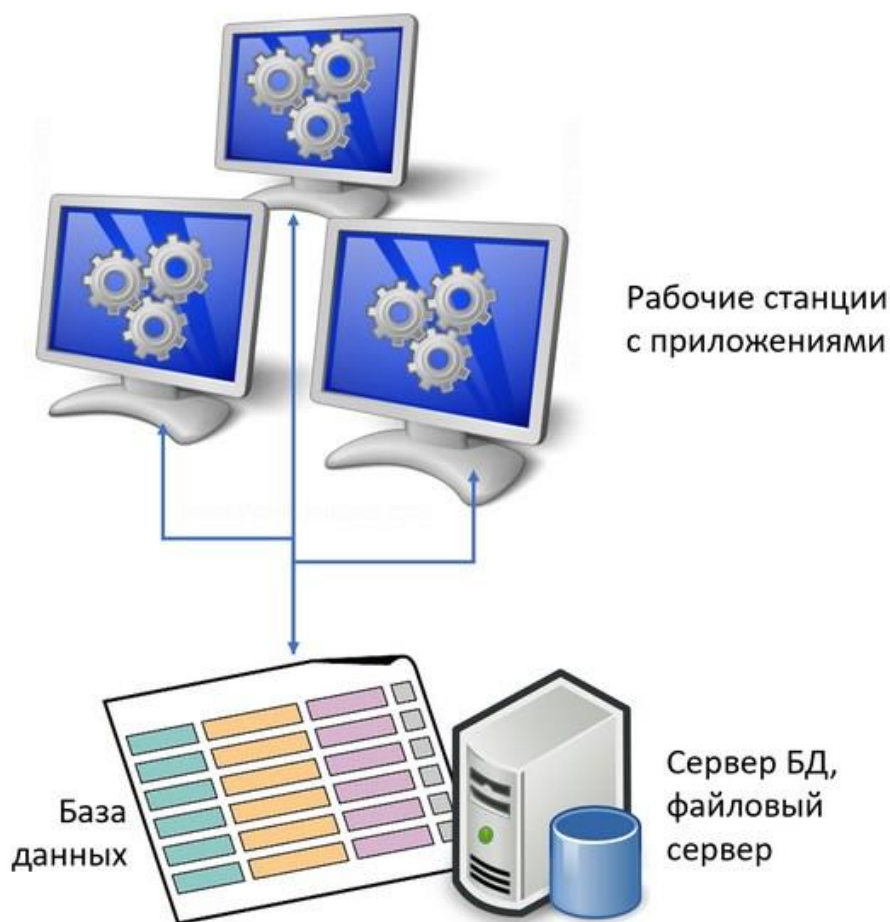


Рисунок 15 – Одноуровневая архитектура клиент-сервер

2. Двухуровневая архитектура

К двухуровневой архитектуре «клиент-сервер», представленную на рисунке 16, следует относить такую, в которой прикладные программы сосредоточены на сервере приложений, а в рабочих станциях находятся программы-клиенты, которые предоставляют

для пользователей интерфейс для работы с приложениями на общем сервере.

Такая архитектура представляется наиболее логичной для архитектуры «клиент-сервер». В ней, однако, можно выделить два варианта. Когда общие данные хранятся на сервере, а логика их обработки и бизнес-данные хранятся на клиентской машине, то такая архитектура носит название “fat client thin server” (толстый клиент, тонкий сервер). Когда не только данные, но и логика их обработки и бизнес-данные хранятся на сервере, то это называется “thin client fat server” (тонкий клиент, толстый сервер).



Рисунок 16 – Двухуровневая архитектура клиент-сервер

Преимущества двухуровневой архитектуры:

- легко конфигурировать и модифицировать приложения;
- пользователю обычно легко работать в такой среде;

- хорошая производительность и масштабируемость.

Однако, у двухуровневой архитектуры есть и недостатки:

- производительность может падать при увеличении числа пользователей;
- потенциальные проблемы с безопасностью, поскольку все данные и программы находятся на центральном сервере;
- все клиенты зависимы от базы данных одного производителя.

3. Трёхуровневая архитектура

В трёхуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение, которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность. Увидеть схему архитектуры можно на рисунке 17.

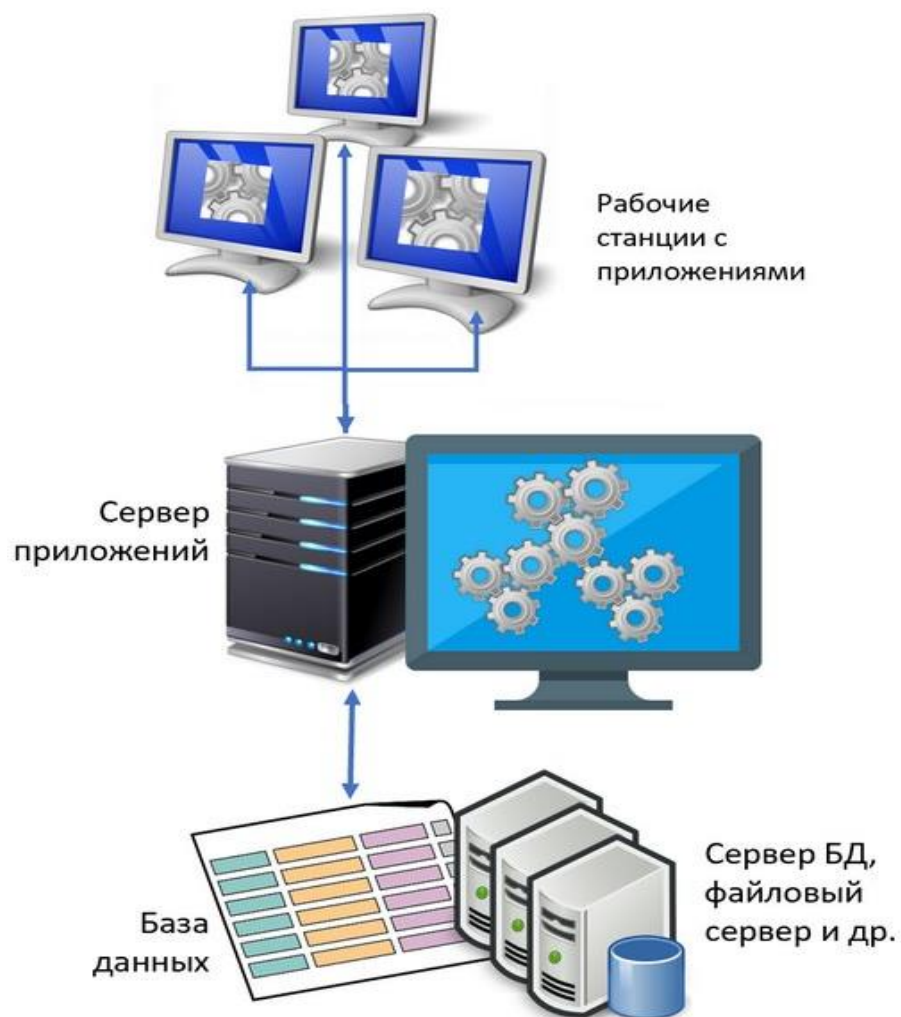


Рисунок 17 – Трёхуровневая архитектура клиент-сервер

Преимущества трёхуровневой архитектуры:

- целостность данных;
- более высокая безопасность, по сравнению с 2-уровневой;
- защищённость базы данных от несанкционированного проникновения.

Недостаток - более сложная структура коммуникаций между клиентов и сервером, поскольку в нём также находится промежуточное программное обеспечение.

2.3.2 Описание микросервисной архитектуры

Микросервисная архитектура — это подход, при котором единое приложение строится из множества слабосвязанных компонентов меньшего

размера, поддерживающих независимое развертывание. На рисунке 18 показан пример как можно выделить микросервисы по ведению каталога товаров, работе с корзиной, оформлению заказов, оплате и так далее.

Как правило, эти компоненты:

- имеют собственный стек технологий, включая базу данных и модель управления данными;
- взаимодействуют между собой посредством REST API, потоков событий и агентов сообщений;
- изолированы по бизнес-функциям с помощью ограниченного контекста.

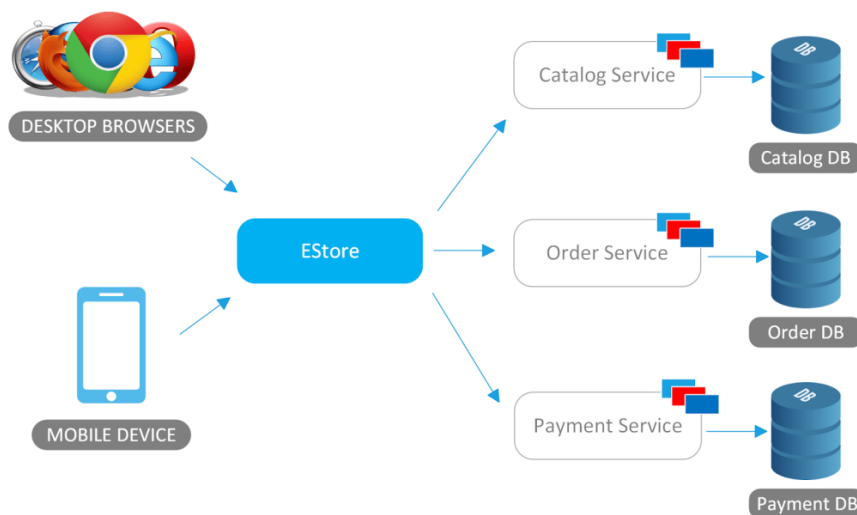


Рисунок 18 – Пример микросервисной архитектуры

Каждый из компонентов отвечает за конкретную задачу, имеет собственное хранилище данных и общается с другими компонентами через простые API-интерфейсы для решения более сложных задач.

Преимущества микросервисной архитектуры:

1. Простота развертывания.

Можно развертывать только изменяющиеся микросервисы, независимо от остальной системы, что позволяет производить обновления чаще и быстрее.

2. Оптимальность масштабирования.

Можно расширять только те сервисы, которые в этом нуждаются, то есть сервисы с наименьшей производительностью, оставляя работать остальные части системы на менее мощном оборудовании.

3. Устойчивость к сбоям.

Отказ одного сервиса не приводит к остановке системы в целом. Когда же ошибка исправлена, необходимое изменение можно развернуть только для соответствующего сервиса — вместо повторного развертывания всего приложения. Но для этого нужно тщательно продумать связи на этапе проектирования.

4. Небольшие команды разработки.

При разработке микросервисов команды принято закреплять за конкретными бизнес-задачами (и сервисами, соответственно). Такие команды, как правило, показывают большую эффективность, а управлять ими легче.

5. Упрощение замены сервисов при необходимости.

Небольшие сервисы проще заменить на более подходящую версию или удалить вовсе — это несет значительно меньше рисков по сравнению с монолитным приложением.

6. Независимость моделей данных.

Каждый микросервис, как правило, использует собственное хранилище данных — поэтому изменение модели данных в одном сервисе не влияет на работу остальных.

Недостатки:

1. Распределенная система.

Микросервисы по своей природе распределены, а это, как известно, имеет свои недостатки: удаленные вызовы медленнее и чаще подвержены сбоям.

2. Необходимость большой и квалифицированной команды для всего проекта.

Рост числа небольших независимых сервисов неизбежно увеличивает сложность проекта.

3. Сложности тестирования.

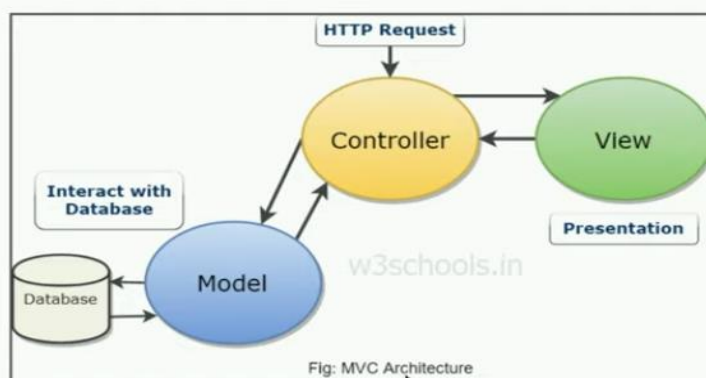
2.3.3 Выбор архитектуры приложения

Ранее были рассмотрены клиент-серверная и микросервисная архитектура. Микросервисная архитектура обладает многими преимуществами в сравнении с клиент-серверной архитектурой, но крайне требовательна к аппаратному и программному обеспечению реализации приложения. Поэтому использование микросервисной архитектуры представляется возможным только при разработке больших систем.

Подводя итог, для разрабатываемого веб-приложения была выбрана трёхуровневая монолитная клиент-серверная архитектура. Она удовлетворяет требованиям производительности, безопасности и надёжности данного веб-приложения.

С целью разграничения внутреннего представления информации от способов ее представления для проектирования программы используется шаблон MVC (Model-View-Controller – Модель-Представление-Контроллер) [26]. Он представлен на рисунке 19. Представленный шаблон изолирует компоненты и позволяет эффективно реализовать повторное использование кода.

MVC (Model - View - Controller)



MVC - паттерн проектирования приложений

- **Model** - логика работы с данными
- **View** - логика представления, интерфейс
- **Controller** - логика навигации, обработка запросов

Рисунок 19 – Шаблон «Модель-Представление-Контроллер»

2.4 Обоснование выбора программных средств

2.4.1 Выбор средств проектирования

Для функционального моделирования Системы прагматичным будет использование специализированного инструмента CA ERwin Process Modeler [10] в полной мере соответствующего заявленным нотациям (IDEF0, IDEF3, DFD), потому что Process Modeler в значительной степени опережает аналогичные программные средства Design/IDEF(поддерживает только IDEF0 и IDEF1X) и Ramus Educational в удобстве использования, оформлении внешнего вида диаграмм, а также предоставляет больше необходимых возможностей и ограничений для корректного создания многоуровневых диаграмм.

Для создания UML диаграмм был выбран ресурс GenMyModel [16]. Это веб-ресурс, поддерживающий все типы UML диаграмм. Возможность онлайн использования отличает в лучшую GenMyModel по сравнению с аналогами

Umbrello [15], StarUML [11], а бесплатность пользования с Visual Paradigm [12].

2.4.2 Обоснование модели данных

На сегодняшний день основными подходами организации данных являются SQL и NoSQL, обладающие своими особенностями и рекомендациями по применению, которые представлены ниже.

SQL:

- жёсткая структура данных, необходимость глубокого анализа предметной области при проектировании БД;
- высокая степень надёжности и стабильности;
- эффективность хранения данных (нормализация препятствует их дублированию и противоречивости);
- гибкие возможности по выполнению запросов;
- относительно низкая скорость выполнения запросов с множеством соединений таблиц;
- широкое сообщество, большое количество документации и вспомогательных материалов;
- стандартизованность языка SQL для различных реляционных СУБД;
- дополнительные трудности при работе с объектно-ориентированной моделью данных приложений.

NoSQL:

- гибкая, легко модифицируемая структура данных;
- возможность хранения данных с произвольной структурой;
- высокая производительность при выполнении небольших запросов, сложность выполнения составных запросов;
- реализации NoSQL сильно отличаются друг от друга, что затрудняет изучение и переход от одной такой СУБД к другой;

- хорошая совместимость с объектно-ориентированной моделью данных приложений.

Опираясь на необходимость выполнения больших и сложных запросов и требования к непротиворечивости и сохранности данных, лучшим решением будет реляционная модель данных.

2.4.3 Выбор СУБД

Для выбора базы данных рассмотрим рейтинг самых популярных СУБД от DB – Engines [17] по состоянию на май 2021 года. Рейтинг представлен на рисунке 20. Согласно источнику, рейтинг основан на следующих параметрах:

- упоминания на сайте;
- частота поиска;
- частота технических обсуждений;
- актуальные предложения о работе;
- профессиональные сетевые профили;
- актуальность в социальных сетях.

DB-Engines Ranking - May 2021

Search...				
Rank	Name	Type	May 2021	Chart May 2021
1.	Oracle	Relational, Multi-model	1269.94	
2.	MySQL	Relational, Multi-model	1236.38	
3.	Microsoft SQL Server	Relational, Multi-model	992.66	
4.	PostgreSQL	Relational, Multi-model	559.25	
5.	MongoDB	Document, Multi-model	481.01	
6.	IBM Db2	Relational, Multi-model	166.66	
7.	Redis	Key-value, Multi-model	162.17	
8.	Elasticsearch	Search engine, Multi-model	155.35	
9.	SQLite	Relational	126.69	
10.	Microsoft Access	Relational	115.40	

Рисунок 20 – Рейтинг самых популярных СУБД на май 2021

Опираясь на представленный рейтинг, самыми популярными реляционными СУБД являются Oracle, MySQL, Microsoft SQL Server и PostgreSQL. Oracle и Microsoft SQL Server распространяются на платной

основе, поэтому их использование будет затруднительным в проекте. Сравним оставшиеся MySQL и PostgreSQL между собой в таблице 2 .

Таблица 2 – Сравнение СУБД

Наименование СУБД → Критерии сравнения ↓	MySQL	PostgreSQL
Ограничение на размер БД	До 100 ГБ	Отсутствует
Наличие документации	Присутствует, на английском языке	Присутствует, на русском и английском языках
Наличие инструмента для визуального проектирования	Присутствует, называется Workbench	Присутствует, называется pgAdmin
Количество поддерживаемых типов данных	Стандартные типы SQL	Поддерживает все типы SQL, а также множество расширенных типов, таких как money, xml и другие.

В результате сравнения была выбрана СУБД PostgreSQL. Она более продвинутая и имеет документацию на русском языке.

2.4.4 Выбор средств реализации серверного приложения

Для выбора средства разработки серверной части веб-приложения обратимся к рейтингу IEEE Spectrum [18] за 2021 год.

Rank	Language	Type	Score
1	Python	Web, Desktop, Server	100.0
2	Java	Web, Mobile, Desktop	95.4
3	JavaScript	Web	88.1
4	C#	Web, Mobile, Desktop, Server	82.4
5	Go	Web, Desktop	77.7
6	HTML	Web	75.4
7	PHP	Web	68.0
8	Dart	Web, Mobile	67.7
9	Ruby	Web, Desktop	63.6
10	Rust	Web, Desktop, Server	63.1

Рисунок 21 – Рейтинг самых популярных языков программирования на 2021 год

Исходя из представленного рейтинга, наиболее популярными языками программирования являются Python, Java, JavaScript, C# и Go. Из всех этих языков Java является самой производительной [19], поэтому она была выбрана для разработки.

Разработка веб-приложения без фреймворка – очень сложное занятие, поэтому нужно выбрать фреймворк. Из множества фреймворков на Java лучше всего выбрать Spring Framework [20], так как:

1. Является самым популярным фреймворком для Java.
2. Spring имеет проверенный опыт быстрого и ответственного решения проблем безопасности. Spring Security упрощает интеграцию со стандартными отраслевыми схемами безопасности и предоставляет надежные решения, которые защищены по умолчанию.

3. У Spring подробная документация, доступная на русском языке.
4. Удобная и вариативная работа с данными. Spring поддерживает технологии JDBC, JBA, Hibernate и другие.

2.4.5 Выбор средств реализации клиентского приложения

Для разработки клиентской части веб-приложения использован HTML (язык разметки), CSS (каскадные таблицы стилей) и JavaScript [21].

Основой создания веб-приложений с использованием JavaScript являются фреймворки. В таблице 3 представлено сравнение популярных JS-фреймворков [22].

Таблица 3 – Сравнение JavaScript-фреймворков

Фреймворки →			
Критерий сравнения ↓	React	Angular	Vue.js
Деление на компоненты	Да	Да	Да
Реактивность	Да	Посредством библиотеки RxJs	Да
Популярность[23]	12 млн скачиваний	537 тысяч скачиваний	3 млн скачиваний
Сложность синтаксиса	Простой	Сложный	Простой

По результатам сравнения лучшим фреймворком оказался React. Он и будет использован в проекте.

2.4.6 Выбранные технологии и программы

Необходимыми для проектирования и разработки модулей Системы программными средствами являются:

1. ERwin Process Modeler – функциональное проектирование Системы.

2. MySQL Workbench – моделирование БД.

3. GenMyModel – UML проектирование Системы.

4. PostgreSQL – разработка БД и серверного функционала, предоставляемого средствами СУБД.

5. Язык программирования Java – разработка серверного приложения.

6. Фреймворк Spring – разработка серверного приложения.

7. Язык программирования JavaScript – разработка клиентского приложения.

8. JavaScript-библиотека React – разработка клиентского приложения.

2.5 Выбор методов тестирования

Тестирования приложений классифицируется по уровням тестирования [24]:

- модульное тестирование – проверка отдельных программных процедур и подпрограмм, включает в себя проверку синтаксиса и кода на соответствие стандартам кодирования;
- интеграционное тестирование – проверка совместной работы отдельных модулей, проводится перед проверкой всей системы; включает в себя проверку функциональности, промежуточных результатов и корректности передачи информации между модулями;
- системное тестирование – проверка системы в целом, ее функционирования на соответствие требованиям заказчика;
- выходное тестирование – проверка готовности приложения для поставки заказчику, проводится независимым тестировщиком;

- приемочное тестирование – проводится организацией.

Для проверки разрабатываемой системы были выбраны модульное, интеграционное и системное тестирование, так как их использование позволит полностью протестировать разрабатываемую систему.

Тестирование может проводиться с помощью следующих технологий [25]:

- стеклянный (белый) ящик – тестирование на этапе разработки, проводится, основываясь на знании исходного кода;
- черный ящик – тестирование без знания внутреннего устройства программы с позиции конечного пользователя;
- серый ящик – тестирование с пониманием внутреннего устройства приложения, но с позиции конечного пользователя.

На протяжении всего процесса разработки должно проводиться тестирование методом стеклянного ящика для исправления возникающих ошибок. После разработки приложения должно быть проведено тестирование методом серого ящика для проверки работы приложения с точки зрения конечного пользователя.

Также существуют следующие виды тестирования:

- тестирование переходов между состояниями;
- тестирование временных зависимостей – проверка работы приложения при попытке вмешаться в программу во время перехода между состояниями;
- нагрузочные испытания – тестирование ограничений возможностей приложения (например, открытие максимально возможного количества файлов);
- тестирование функциональной эквивалентности – сравнение результатов вычислений одной и той же математической функции разными программами.

Актуальным для представленного проекта является тестирование переходов между состояниями и тестирование временных зависимостей для проверки работоспособности программы при вмешательстве в выполняемый процесс.

Таким образом, для тестирования были выбраны модульное, интеграционное и системное тестирование приложения с помощью технологий белого и серого ящиков, а также тестирование переходов между состояниями и временных зависимостей.

2.6 Программная реализация

Система разработана в соответствии с техническим заданием, представленным в приложении А.

Техническое описание представлено в приложении Б.

Реализация серверной части системы осуществлялась на языке Java с использованием фреймворка Spring, клиентская часть была выполнена на языке JavaScript с использованием фреймворка React.

Функциональная модель поведения приложения соответствует модели на рисунках Б.1 – Б.4.

Поведение пользователей системы соответствует диаграмме вариантов использования, представленной на рисунке Б.5.

База данных приложения реализованы в соответствии с моделью, представленной на рисунках Б.6 и в таблицах Б.1 – Б.8.

Архитектура приложения соответствует диаграмме развертывания, представленной на рисунке Б.8.

Система реализована в виде веб-приложения. Для информационного обеспечения системы использована технология реляционных баз данных и ORM [27]. Для авторизации и аутентификации пользователей использована технология JWT [28]. С целью разграничения внутреннего представления

информации от способов ее представления для проектирования программы используется шаблон MVC [26].

3. ЭКОНОМИЧЕСКАЯ ОЦЕНКА РАЗРАБОТКИ

3.1 Расчёт стоимости разработки образовательного веб-ресурса по эпохе Петра I

В ходе выполнения дипломной работы был произведён расчет фонда оплаты труда специалистов, задействованных в реализации проекта:

- Fullstack разработчик (Java + JS).

При расчете фонда оплаты труда рассматривался случай привлечения специалистов по договорам гражданско-правового характера, соответственно рассчитывался налог на доход физических лиц, который составляет 13%, и страховые взносы, которые составляют 27,1%.

Результаты расчета фонда оплаты труда представлены в таблице 4.

Таблица 4 – Фонд оплаты труда

Перечень выполняемых работ	Исполнители		Кол-во человеко-дней	Средняя оплата труда за 1 день(руб.)	Оплата труда (с учетом НДФЛ) (руб.)	Страховые взносы по договорам ГПХ(руб.)	ИТОГО (руб.)
	Количество	Должность					
Исследование предметной области	1	Fullstack разработчик	4	8 181,82	32 727,28	8 869,09	41 596,37
Разработка ТЗ	1	Fullstack разработчик	4	8 181,82	32 727,28	8 869,09	41 596,37
Согласование ТЗ	1	Fullstack разработчик	2	8 181,82	16 363,64	4 434,55	20 798,19
Моделирование веб-ресурса	1	Fullstack разработчик	7	8 181,82	57 272,74	15 520,91	72 793,65
Разработка пользовательского интерфейса	1	Fullstack разработчик	7	8 181,82	57 272,74	15 520,91	72 793,65
Разработка веб-ресурса	1	Fullstack разработчик	22	8 181,82	180 000,04	48 780,01	228 780,05
Тестирование	1	Fullstack разработчик	5	8 181,82	40 909,10	11 086,37	51 995,47
Выкладывание на хостинг готового веб-ресурса	1	Fullstack разработчик	2	8 181,82	16 363,64	4 434,55	20 798,19

Средняя оплата труда каждого специалиста за 1 день рассчитывалась исходя из анализа рынка вакансий по Санкт-Петербургу. Результаты анализа представлены в таблице 5.

Таблица 5 – Анализ вакансий

Должность	Размер оплаты в месяц на основе анализа (руб.)	Размер оплаты в день (руб.)	Ссылка с подтверждением размера оплаты труда
Fullstack разработчик (Java + JS)	180 000,00	8 181,82	https://vk.cc/cel1aZJ Хорошему разработчику – достойная зарплата

Был произведен расчет дополнительных расходов, необходимых для разработки проекта. Результаты представлены в таблице 6.

Таблица 6 – Дополнительные расходы

Тип	Кол-во исполнителей	Кол-во дней	Стоимость в день (руб.)	Итого (руб.)	Ссылка с подтверждением цены
Интернет	1	53	11,6	614,80	https://vk.cc/cel1cPg Разработчику необходим интернет
Электроэнергия	1	53	4,98	263,94	https://vk.cc/cel1cRC Разработчику необходимо электричество
Домен	1	365	1,64	600,00	https://vk.cc/cdZKS5 На сайт нужно как-то зайти
VIP-хостинг	1	365	22,52	8 220,00	https://vk.cc/cdZKS5 Сайт должен где-то храниться

Общая стоимость расходов на разработку образовательного веб-ресурса по эпохе Петра I составила 560 850,68 рублей, в т.ч. 551 151,94 рублей на фонд оплаты труда и 9 698,74 рублей на дополнительные расходы.

3.2 Выводы по результатам экономической оценки

Был проведен ценовой анализ аналогичных решений, представленный в таблице 7.

В качестве аналогичных решений были проанализированы следующие разработки:

1. Веб-сайт для создания и прохождения упражнений “LearningApps”.
2. Веб-сайт для создания и прохождения тестов “PenCup”.
3. Глобальная система тестирований “TestServer”.

Стоимость аналогичных решений по результатам анализа представлена в таблице 7.

Таблица 7 – Результаты ценового анализа аналогичных решений

Наименование аналогичного решения	Стоимость (руб.)
LearningApps	843 000,00
PenCup	978 000,00
TestServer	714 000,00
Средняя стоимость	845 000,00

Проведя ценовой анализ аналогичных решений и сравнив их со стоимостью разработки, представленной выше в дипломном проекте, было выявлено, что разрабатываемый веб-ресурс по стоимости меньше на 44% средней рыночной цены аналогичных решений.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта был разработан образовательный веб-ресурс по эпохе Петра I.

В полном объёме выполнены задачи проекта:

- проанализирована предметная область;
- сформированы требования к системе;
- определены входные и выходные данные;
- выбраны и обоснованы средства реализации;
- определена архитектура приложения;
- произведено детальное проектирование;
- выбраны методы тестирования;
- сформировано техническое задание;
- разработана серверная часть веб-ресурса;
- разработана клиентская часть веб-ресурса;
- протестирован веб-ресурс.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методология функционального моделирования IDEF0. Москва: Госстандарт России, 2000. 75 с.
2. Бистерфельд О. А. Моделирование бизнес-процессов с использованием методологии IDEF3: Учебно-методическое пособие Рязань: Ряз. гос. ун-т им. С. А. Есенина 2008. 44 с.
3. Киселев Д. Ю., Киселев Ю. В., Макарьев В. Д. Структурный анализ потоков данных (Data Flow Diagrams – DFD) Самара: Издательство СГАУ, 2014. 12 с.
4. Леоненков А. В. Самоучитель UML 2-е издание. СПб: БХВ-Петербург, 2004. 432с. (дата обращения: 20.10.2021)
5. Проектирование базы данных [Электронный ресурс] // БГЭУ – учебно-методический комплекс : [сайт], 2005-2019. URL: http://www.bseu.by/it/tohod/lekcii4_3.htm (дата обращения: 20.10.2021)
6. Нормальные формы [Электронный ресурс] // Научно-исследовательская лаборатория «Бизнес-школа информационных технологий» Регионального финансово-экономического институт : [сайт], 2020г. URL: <https://it.rfei.ru/course/~gk8r/~tjrlefRQ/~Mv0JyjU5> (дата обращения: 20.10.2021)
7. Itelon [Электронный ресурс] // Статья об архитектуре Клиент-Сервер : [сайт], URL: <https://itelon.ru/blog/arkhitektura-klient-server/> (дата обращения: 24.10.2021)
8. VK Cloud Solutions [Электронный ресурс] // Статья о микросервисной архитектуре : [сайт], URL: <https://mcs.mail.ru/blog/prostym-jazykom-o-mikroservisnoj-arhitecture> (дата обращения: 24.10.2021)

9. Академия Microsoft: Базы данных // НОУ «ИНТУИТ» [сайт], 2003–2021 URL: <https://intuit.ru/studies/courses/508/364/lecture/8643?page=2> (дата обращения: 20.10.2021)
10. Business Process Modeling & Analysis [Электронный ресурс] // Erwin by Quest [сайт], 2021г. URL: <https://erwin.com/solutions/business-process-modeling/> (дата обращения: 24.10.2021))
11. StarUML [Электронный ресурс] // URL: <https://staruml.io/> (дата обращения: 24.10.2021)
12. Visual Paradigm [Электронный ресурс] // URL: <https://www.visual-paradigm.com/> (дата обращения: 24.10.2021)
13. Edraw Soft [Электронный ресурс] // URL: <https://www.edrawsoft.com/> (дата обращения: 24.10.2021)
14. Online Visual Paradigm [Электронный ресурс] // URL: <https://online.visual-paradigm.com/> (дата обращения: 24.10.2021)
15. Umbrello [Электронный ресурс] // URL: <https://umbrello.kde.org/> (дата обращения: 24.10.2021)
16. GenMyModel [Электронный ресурс] // URL: <https://www.genmymodel.com/> (дата обращения: 24.10.2021)
17. DB-Engines Ranking [Электронный ресурс] // DB-Engines [сайт], 2021 URL: <https://db-engines.com/en/ranking> (дата обращения: 24.10.2021)
18. Top Programming Languages 2021 [Электронный ресурс] // IEEE Spectrum [сайт], 2021 URL: <https://spectrum.ieee.org/top-programming-languages-2021> (дата обращения: 24.10.2021)
19. Energy Efficiency across Programming Languages [Электронный ресурс] // IEEE Spectrum [сайт], 2021 URL:

- https://greenlab.di.uminho.pt/wpcontent/uploads/2017/09/paperSL_E.pdf (дата обращения: 24.10.2021)
- 20.Spring [Электронный ресурс] // URL: <https://spring.io/> (дата обращения: 24.10.2021)
- 21.GeekBrains [Электронный ресурс] // Выбираем язык для веб-разработки : [сайт] URL: https://geekbrains.ru/posts/road_to_web_development (дата обращения 24.10.2021)
- 22.VC.ru [Электронный ресурс] // Популярные фреймворки JavaScript : [сайт], URL: <https://vc.ru/dev/147263-populyarnye-freymvorki-javascript> (дата обращения 24.10.2021)
- 23.angular vs react vs vue [Электронный ресурс] // npm trends [сайт], 2021 URL: <https://www.npmtrends.com/angular-vs-react-vs-vue> (дата обращения: 24.10.2021)
- 24.Рудаков А. В., Федорова Г. Н. Технология разработки программных продуктов. Практический курс: учеб. Пособие для студ. учреждений сред. проф. образования. М. : Издательский центр «Академия», 2010. 192 с.
- 25.QAlabs [Электронный ресурс] // Уровни и методы тестирования : [сайт], URL: <https://qalabs.com.ua/urovni-testirovaniya.html> (дата обращения 23.02.2021).
- 26.Habr [Электронный ресурс] // MVC для веб: проще некуда : [сайт], URL: <https://habr.com/ru/post/181772/> (дата обращения 23.02.2021).
- 27.Habr [Электронный ресурс] // ORM или как забыть о проектировании БД : [сайт], URL: <https://habr.com/ru/post/237889/> (дата обращения 23.02.2021).

28.Habr [Электронный ресурс] // Пять простых шагов для понимания JSON Web Tokens (JWT) : [сайт], URL: <https://habr.com/ru/post/340146/> (дата обращения 23.02.2021).

ПРИЛОЖЕНИЕ А

Техническое задание

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Назначением разработки является автоматизация создания тестов и проведения тестирования по эпохе Петра I.

Основной целевой аудиторией являются преподаватели истории и обучающиеся, а также люди, желающие ознакомиться с личностью Петра I.

Веб-приложение позволит улучшить процесс изучения эпохи Петра I и создать единое информационное пространство, в котором будут взаимодействовать преподаватель и обучающийся.

2. ЦЕЛЬ И ЗАДАЧИ, РЕШАЕМЫЕ В ПРОЦЕССЕ ДОСТИЖЕНИЯ ЦЕЛИ

Целью является создание веб-приложения для составления и проведения тестирования по эпохе Петра I в рамках проекта «PETRO pr(i[t]mo) научно-техническое наследие петровской эпохи».

Задачи, решаемые в процессе достижения цели:

1. Уточнение требований к системе (при необходимости).
2. Детальное проектирование системы.
3. Программная реализация.
4. Тестирование.

3. ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ

3.1 Требования к функциональным характеристикам. Описание функциональности разрабатываемой системы

В Системе должны быть определены две категории пользователей:

1. Студент.
2. Преподаватель.

Функционал, предоставляемый пользователям категории Студент:

- создание/редактирование личного аккаунта (регистрация в системе);
- авторизация;
- просмотр данных личного аккаунта;
- добавление и редактирование ответов к вопросам теста;
- оценивание тестов;
- просмотр тестов.

Преподаватель обладает функционалом Студента, но имеет дополнительную функцию - управление (создание/редактирование/удаление) тестами.

3.2 Описание входных и выходных данных

Входные данные:

Для пользователя категории Студент:

- данные личного аккаунта (при создании или редактировании):
 - логин;
 - пароль;
 - адрес электронной почты;
 - ФИО;
 - роль;
 - аватар-изображение.
- данные ответа (при создании или редактировании) - ответ.

- данные теста - оценка теста.

Для пользователей категории Преподаватель:

- перечисленные входные данные для категории Студент;
- данные теста (при создании):
 - название;
 - тип;
 - описание;
 - картинка.
- данные вопроса(ов) к тесту (при создании):
 - вопрос;
 - картинка;
 - категория вопроса;
 - сложность вопроса;
 - тип вопроса.
- данные ответа(ов) к тесту (при создании):
 - ответ;
 - правильность ответ.
- данные результата(ов) к тесту (при создании):
 - заголовок;
 - описание;
 - картинка;
 - условие получения;
 - успешно ли результат заканчивает тест.

Выходные данные:

Для пользователя категории Студент:

- личные данные аккаунта пользователя:
 - логин;
 - адрес электронной почты;
 - ФИО;

- аватар-изображение;
- роль;
- кол-во пройденных тестов;
- дата регистрации.
- результаты прохождения теста:
 - выполнен ли тест;
 - оценка теста;
 - ответы пользователя;
 - дата прохождения теста;
 - процент правильных ответов.
- данные теста:
 - название;
 - тип;
 - кол-во вопросов;
 - дата создания;
 - создатель теста;
 - оценка теста;
 - описание;
 - картинка.

Для пользователей категории Преподаватель:

- перечисленные входные данные для категории Студент;
- ответы пользователей:
 - тест;
 - вопросы;
 - ответы;
 - правильность ответов.
- данные вопроса(ов) к тесту:
 - вопрос;
 - картинка;

- тип вопроса.
- данные ответа(ов) к тесту:
 - ответ;
 - правильность ответа.
- данные результата(ов) к тесту:
 - заголовок;
 - описание;
 - картинка;
 - условие получения;
 - успешно ли результат заканчивает тест.

3.3 Модель приложения

Функциональная модель поведения приложения должна соответствовать модели на рисунках Б.1-Б.4.

Поведение пользователей системы должна соответствовать модели, представленной на рисунке Б.5.

Структура баз данных должна соответствовать моделям на рисунках Б.6 и в таблицах Б.1 – Б.8.

Поведение клиентского и серверного приложения при действиях пользователя должны соответствовать диаграмме активности на рисунке Б.7.

Разбиение и связь структурных компонентов приложения должны соответствовать диаграмме развертывания на рисунке Б.8.

3.4 Эргономические и технико-эстетические требования

Взаимодействие пользователя с системой должно осуществляться посредством визуального графического интерфейса (GUI). Интерфейс должен обеспечивать быстрое отображение экранных форм. Ввод-вывод данных системы, прием управляющих команд и отображение результатов их исполнения должны выполняться в интерактивном режиме. Интерфейс

должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

Управление системой должно осуществляться с помощью экранных кнопок, меню, значков и т.п. элементов.

Все надписи экранных форм и сообщения должны быть на русском языке.

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей.

Экранные формы должны проектироваться с учетом требований унификации:

- все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;
- для обозначения сходных операций должны использоваться сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (запись на занятие, просмотр расписания), а также последовательности действий пользователя при их выполнении, должны быть унифицированы.

3.5 Требования к информационному обмену между компонентами приложения

Информационный обмен между подсистемами должен осуществляться через единое информационное пространство посредством использования стандартизированных протоколов и форматов обмена данными.

Все компоненты приложения должны функционировать в пределах единого логического пространства, обеспеченного интегрированными средствами сервера данных и клиентского приложения.

3.6 Структура приложения

Интерфейс Студента должен состоять из следующих страниц:

- страница авторизации;
- страница регистрации;
- страница профиля пользователя;
- страница выбора теста;
- страница прохождения теста;
- страница просмотра результатов теста.

Пользователя при входе на сайт встречает страницу авторизации. При отсутствии аккаунта пользователь может зарегистрироваться на ресурсе. При потере пароля он может быть восстановлен.

С помощью навигационного меню должен осуществляться переход на остальные страницы.

Интерфейс Преподавателя включает в себя все страницы студента, а также следующие страницы:

- страница создания теста;
- страница редактирования теста;
- страница просмотра созданных тестов.

Создание нового теста должно быть осуществлено посредством выбора методики теста и ввода названия, описания, результатов, вопросов и ответов на них.

3.7 Требования по применению систем управления базами данных

Хранение данных в системе должны быть основаны на современной реляционной СУБД, которая должна удовлетворять следующим требованиям:

1. СУБД должна располагать инструментами управления, контроля и резервирования данных.
2. СУБД должна иметь встроенные средства защиты от несанкционированного доступа.

3.8 Требования по лингвистическому обеспечению системы

При реализации системы должны применяться следующие языки высокого уровня: SQL, Java с фреймворком Spring, JavaScript с фреймворком React.

Должны выполняться следующие требования к кодированию и декодированию данных: UTF-8 для подсистемы хранения данных; UTF-8 информации, поступающей из систем-источников.

Для реализации алгоритмов манипулирования данными необходимо использовать стандартный язык запроса к данным SQL.

Для описания предметной области (объекта автоматизации) должен использоваться ERwin Process Modeler.

3.9 Требования к защите информации от несанкционированного доступа и разграничение прав доступа

Разграничение прав доступа пользователей Системы должно строиться по принципу «что не разрешено, то запрещено».

Защищённая часть системы должна использовать «слепые» пароли (при наборе пароля его символы не показываются на экране либо заменяются одним типом символов).

В соответствии с правами доступа пользователи с типом Студент могут:

- выбрать тест;
- пройти тест;
- оценить тест;
- просматривать результаты тестов;
- просматривать и редактировать свой профиль.

Пользователям с типом Преподаватель доступно всё, что доступно Студентам, а также они могут составлять/редактировать/удалять тесты.

3.10 Требования по сохранности информации при авариях

Информация в базе данных системы должна сохраняться при возникновении аварийных ситуаций, связанных со сбоями электропитания/оборудования.

Резервное копирование данных должно осуществляться на регулярной основе, в объемах, достаточных для восстановления информации.

3.11 Требования к технологиям разработки и использования

При разработке приложения должны применяться технология баз данных, веб-технологии и парадигма объектно-ориентированного программирования.

Архитектура приложения должна соответствовать архитектуре клиент-сервер: на сервере должны храниться данные для входа и ограничения клиентов и обрабатываться запросы клиента, также сервер должен обращаться к удаленной базе данных, клиент должен передавать запросы серверу.

3.12 Требования к программным средствам разработки

Для реализации приложения должны использоваться следующие средства разработки:

- для разработки базы данных – pgAdmin 4;
- для разработки приложения – IntelliJ IDEA Ultimate 2021.3.

При эксплуатации приложения должны быть установлены:

- на сервере: PostgreSQL v12 и выше;
- на сервере: JDK 14 и выше;
- на клиенте: браузер (Google Chrome 87+, Edge 87+, Firefox 84+, Safari 13.1+, Opera 72+).

3.13 Требования к составу и параметрам технических средств, применяемых при разработке системы, с одной стороны, и при использовании системы, с другой стороны

При разработке системы технические средства должны удовлетворять следующим требованиям:

- ОС Windows 10 и выше;
- RAM - не менее 8Гб.

При использовании системы технические средства должны удовлетворять следующим требованиям:

- ОС Microsoft Windows (7, 8, 8.1, 10,11) или Apple OS X (macOS) (10.9-10.14);
- Веб-браузер (Google Chrome 87+, Edge 87+, Firefox 84+, Safari 13.1+, Opera 72+);
- RAM - не менее 2Гб.

4. ТРЕБОВАНИЯ К МЕТОДАМ ТЕСТИРОВАНИЯ

Тестирование должно быть проведено на следующих уровнях:

1. На уровне процедур и подпрограмм – модульное тестирование.
2. На уровне взаимодействия модулей – интеграционное тестирование.
3. На уровне всей системы – системное тестирование.

На этапе разработки должно применяться тестирование методом «стеклянного ящика» с проверкой переходов между состояниями. После разработки необходимо провести тестирование методом «серого ящика» с проверкой временных зависимостей.

ПРИЛОЖЕНИЕ Б

Модель разработки

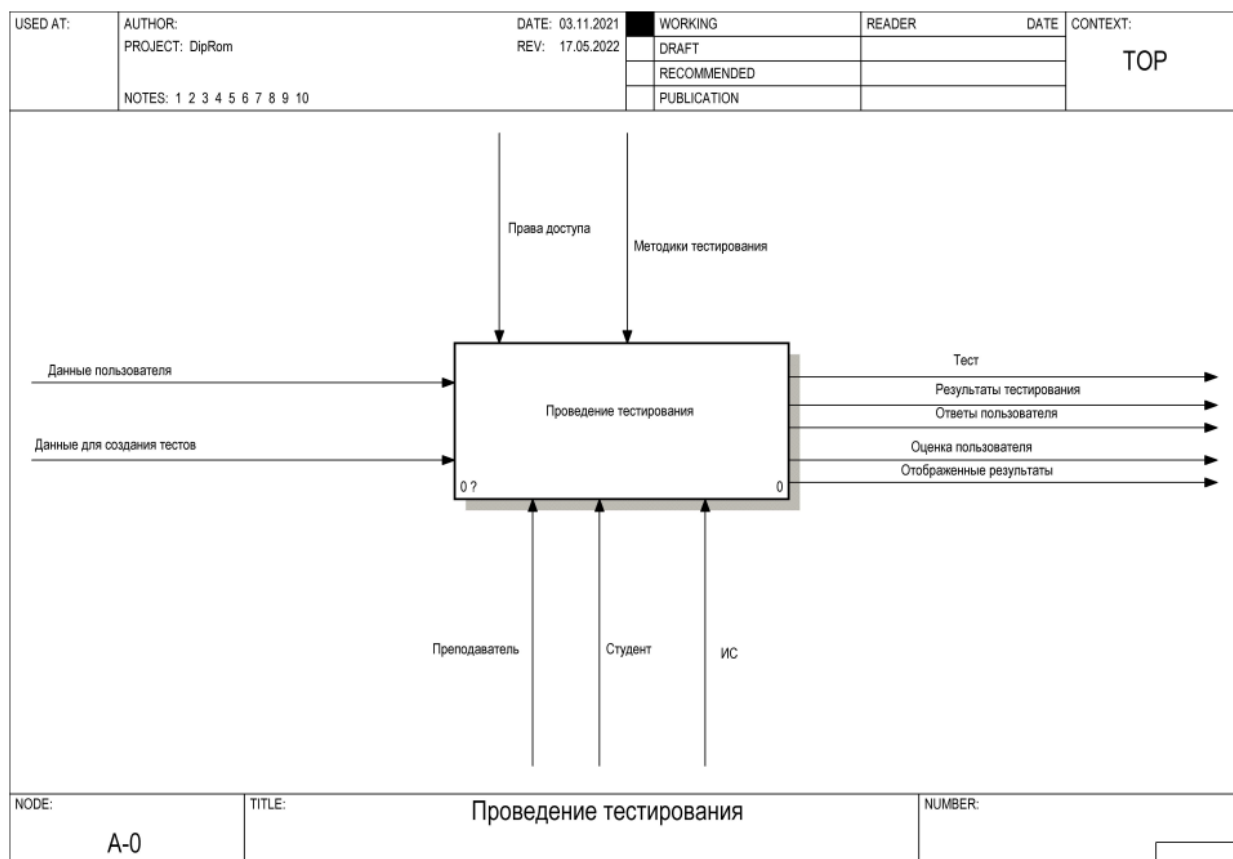


Рисунок Б.1 – Функциональная модель Системы: главный процесс «проведение тестирования»

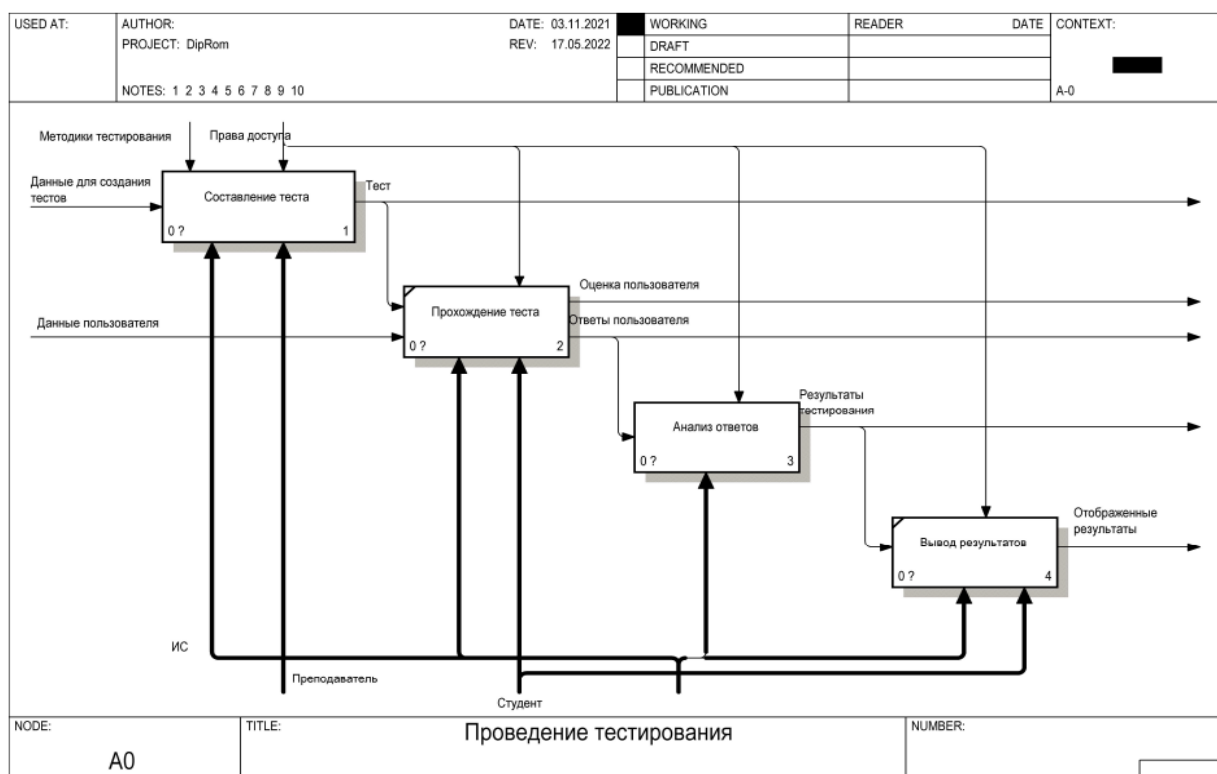


Рисунок Б.2 – Функциональная модель Системы: детализация главного процесса

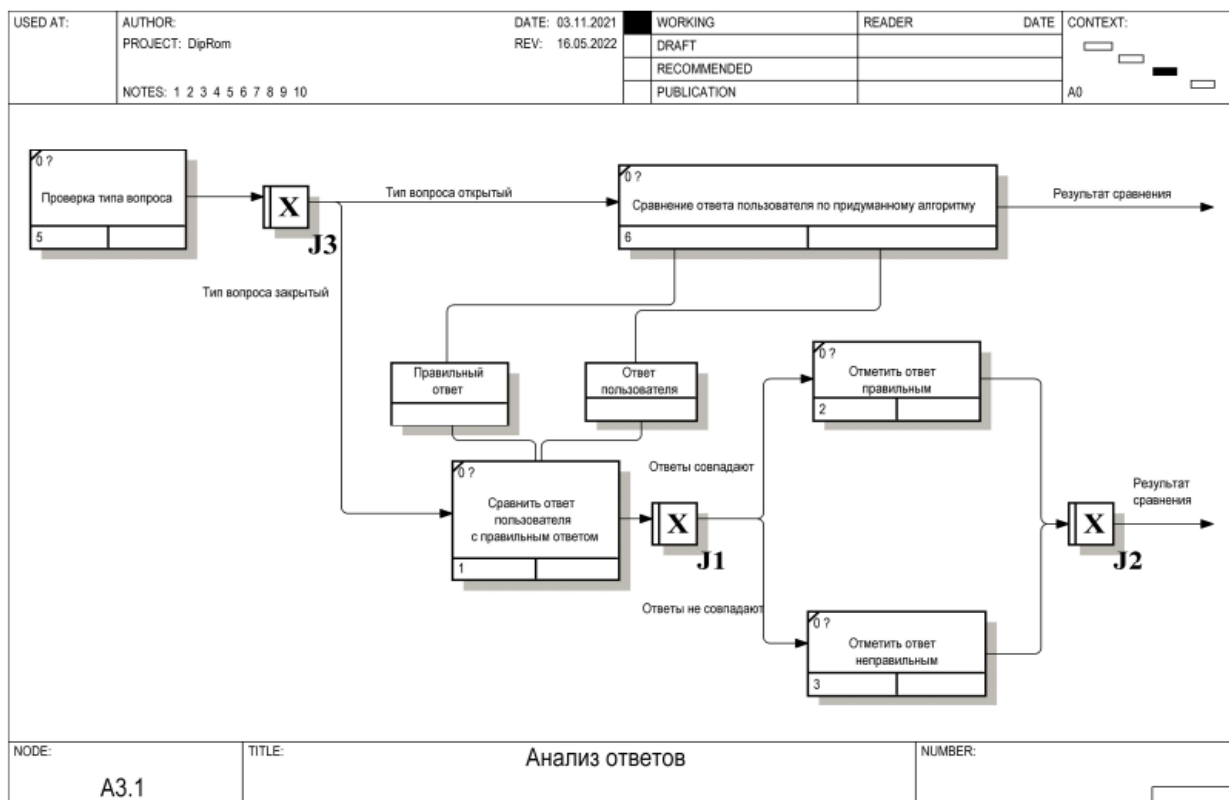


Рисунок Б.3 – Функциональная модель Системы: детализация процесса «анализирование ответов» в нотации IDEF3

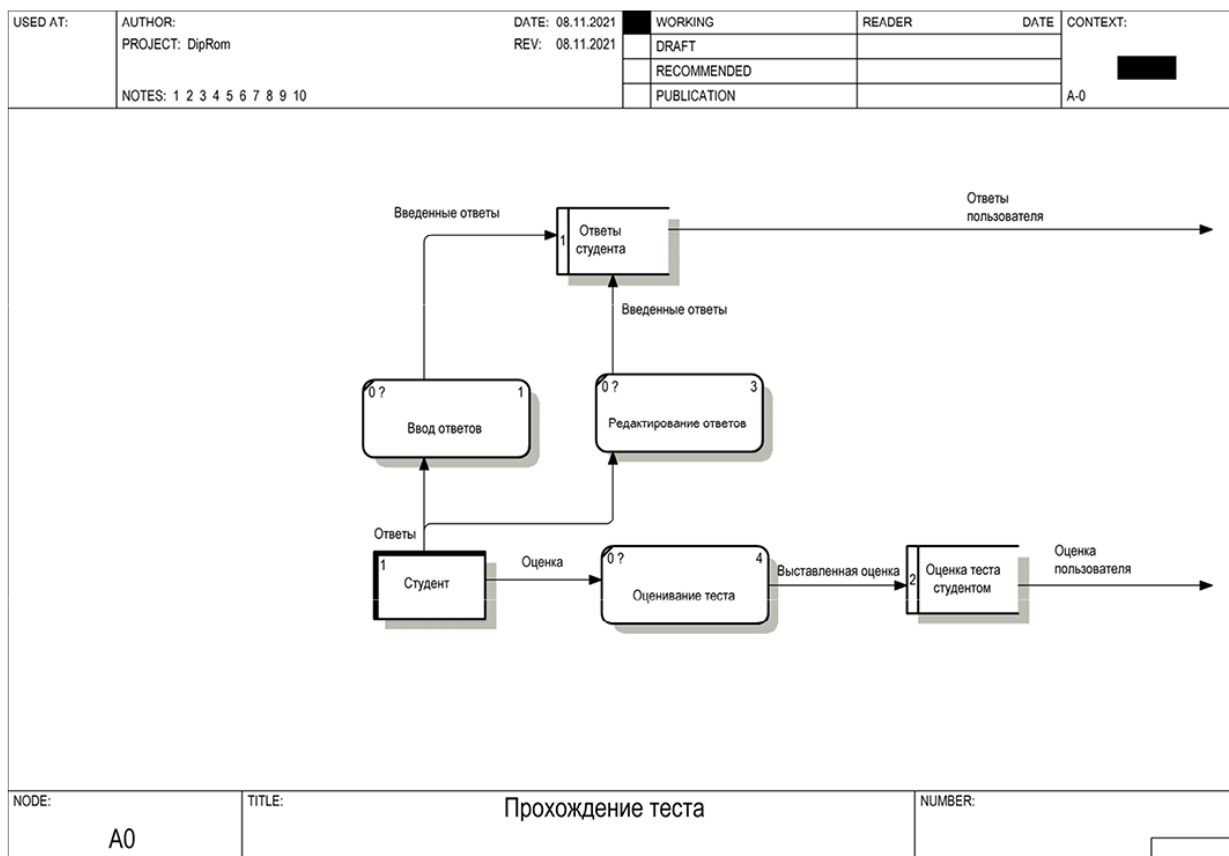


Рисунок Б.4 – Функциональная модель Системы: детализация процесса «прохождение теста» в нотации DFD

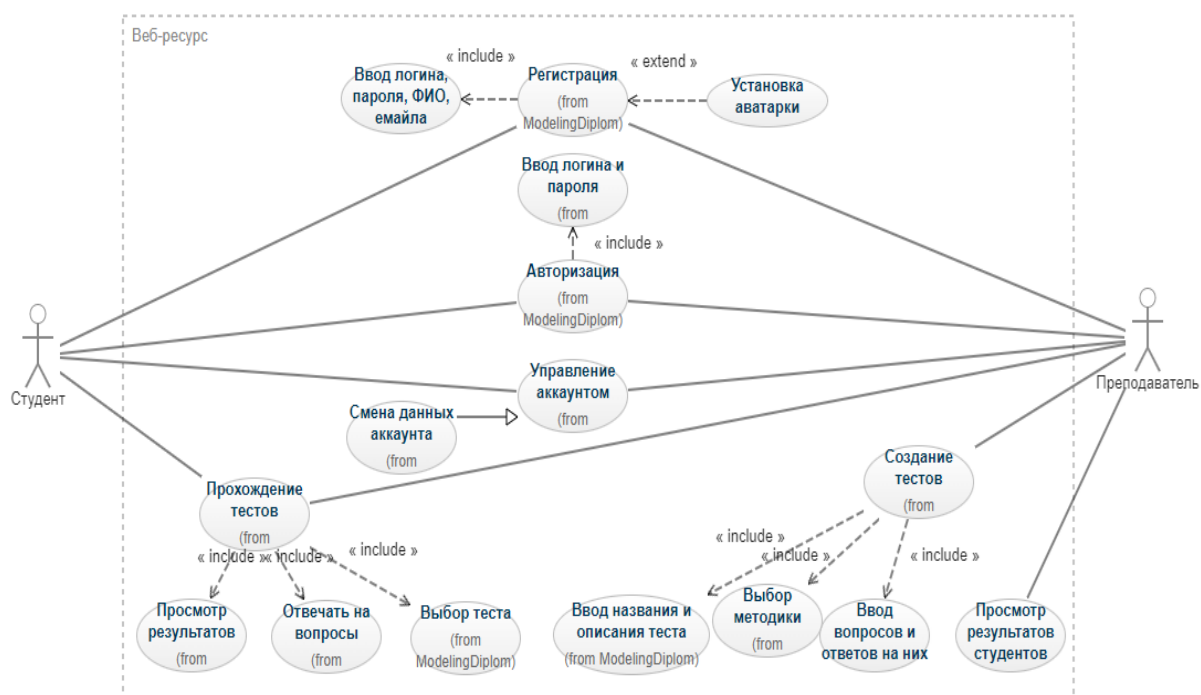


Рисунок Б.5 – Модель поведения пользователей Системы

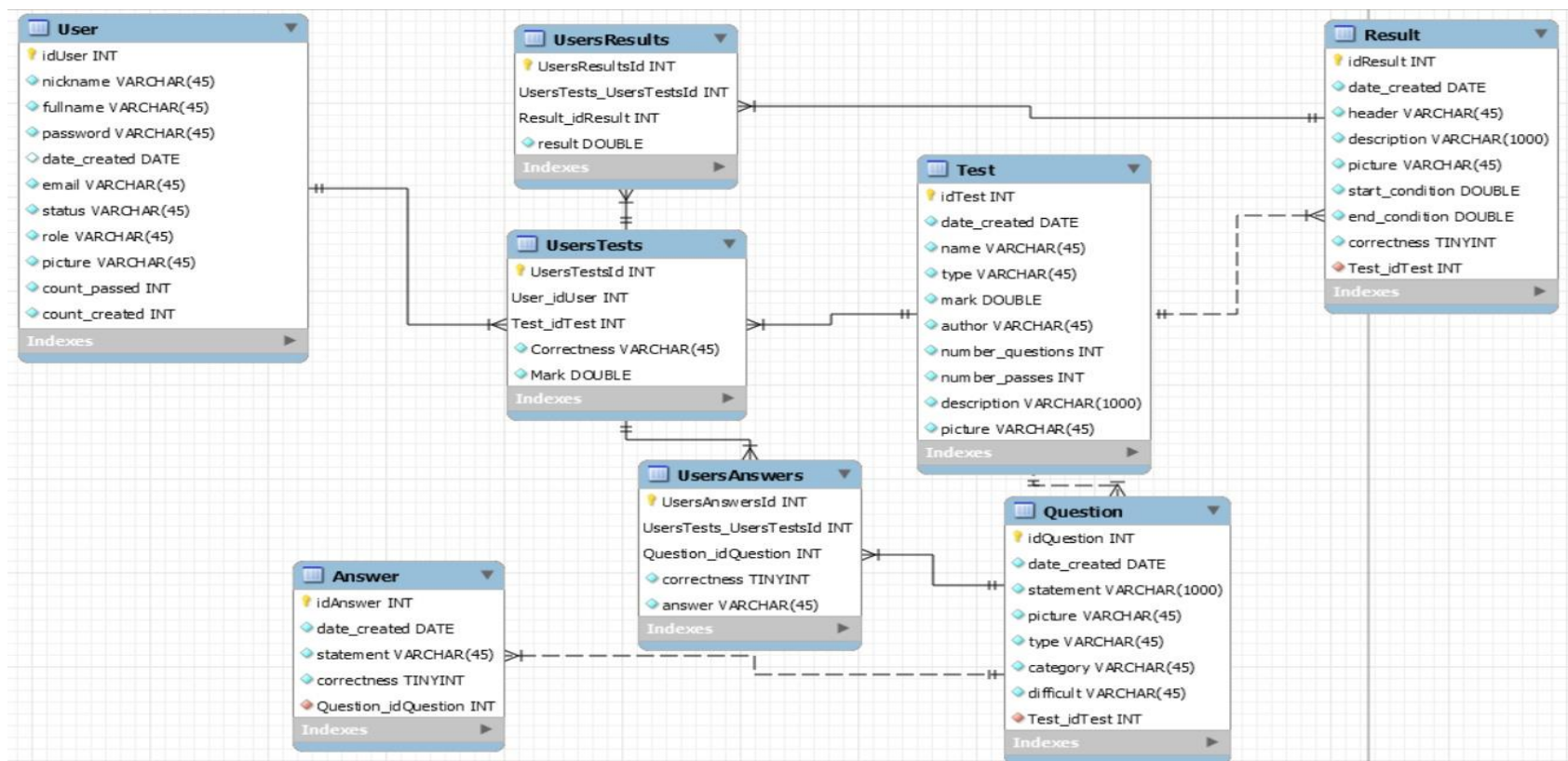


Рисунок Б.6 – Схема инфологической модели БД Системы

Логическая модель Системы представлена в таблицах Б.1 – Б.8, где РК (англ. Primary Key) – первичный ключ, FK (англ. Foreign Key) – внешний ключ, ЗПоУ – Значение по умолчанию, СиП – символы из перечня: *пробел ! “ # \$ % & ' () * + , - . / : ; < = > ? @ № [\] ^ _ ` { / } ~ – —*.

Таблица Б.1 - User

Имя столбца	Тип данного	PK	FK	ЗПоУ	Ограничения	Комментарии
idUser	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор пользователя
nickname	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, нижние подчёркивания, дефисы и цифры; длина – от 3 до 45 символов	Псевдоним пользователя
password	VARCHAR(45)	-	-	-	Результат вычисления алгоритма sha256 над паролем (содержащем от 5 до 20 символов латиницы, кириллицы, СиП или цифр); каждый символ – шестнадцатеричное число (от “0” до “f”)	Хэш-сумма пароля пользователя
email	VARCHAR(45)	-	-	-	Соответствие стандартной маске email-адреса, максимальная длина – 45 символов	Адрес электронной почты пользователя
fullname	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, нижние подчёркивания, дефисы и цифры; длина – от 3 до 45 символов	ФИО пользователя
status	BOOL	-	-	-	Значение 1 или 0(true or false)	Статус пользователя в системе (Преподаватель или Студент)
date_created	DATE	-	-	-	Задаётся автоматически в момент добавление записи	Дата регистрации пользователя
picture	VARCHAR(500)	-	-	-	Любые символы, до 500	Путь к картинке

Таблица Б.2 - Test

Имя столбца	Тип данного	PK	FK	ЗПоУ	Ограничения	Комментарии
idTest	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор курса
name	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Название
author	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Кем создан
date_created	DATE	-	-	-	Задаётся автоматически в момент добавление записи	Дата создания
mark	DOUBLE	-	-	-	Положительное вещественное число	Оценка теста
type	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Тип теста
description	VARCHAR(300)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Краткое описание теста
number_questions	INT	-	-	-	Положительное целое число	Кол-во вопросов в тесте
number_passes	INT	-	-	-	Положительное целое число	Кол-во прохождений
picture	VARCHAR(500)	-	-	-	Любые символы, до 500	Путь к картинке

Таблица Б.3 – UsersTests

Имя столбца	Тип данного	РК	FK	ЗПоУ	Ограничения	Комментарии
UsersTestsId	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
User_idUser	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Пользователь
Test_idTest	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Тест
mark	DOUBLE	-	-	-	Положительное вещественное число до 5.0	Оценка от пользователя
correctness	VARCHAR(45)	-	-	-	Любые символы, до 45	Правильность

Таблица Б.4 – UsersResults

Имя столбца	Тип данного	РК	FK	ЗПоУ	Ограничения	Комментарии
UsersResultsId	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
Result_idUser	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Результат
UserTests_idUsersTests	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Тест, который проходит пользователь
result	DOUBLE	-	-	-	Положительное вещественное число до 1.0	Процент правильных ответов

Таблица Б.5 – Question

Имя столбца	Тип данного	РК	FK	ЗПоУ	Ограничения	Комментарии
idQuestion	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
statement	VARCHAR(300)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 300 символов.	Вопрос к заданию
type	VARCHAR(45)	-	-	-	Любые символы, до 45	Тип вопроса
picture	VARCHAR(500)	-	-	-	Любые символы, до 500	Путь к картинке
Test_idTest	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Тест
category	VARCHAR(45)	-	-	-	Любые символы, до 45	Категория вопроса
difficult	VARCHAR(45)	-	-	-	Любые символы, до 45	Сложность вопроса
date_created	DATE	-	-	-	Задаётся автоматически в момент добавление записи	Дата создания

Таблица Б.6 – Answer

Имя столбца	Тип данного	PK	FK	ЗПоУ	Ограничения	Комментарии
idAnswer	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
Question _id Question	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Задание курса
answer	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Ответ
correctness	BOOLEAN	-	-	False	Булевы значения	Правильный ли ответ
date_created	DATE	-	-	-	Задаётся автоматически в момент добавление записи	Дата создания

Таблица Б.7 – UserAnswers

Имя столбца	Тип данного	PK	FK	ЗПоУ	Ограничения	Комментарии
UsersAnswersId	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
User_idUser	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Пользователь
Users_Tests_idUsers_Tests	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Тесты пользователя
Question_idQuestion	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Задание курса
answer	VARCHAR(45)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 45 символов.	Ответ
correct	BOOLEAN	-	-	False	Булевы значения	Правильный ли ответ

Таблица Б.8 – Result

Имя столбца	Тип данного	PK	FK	ЗПоУ	Ограничения	Комментарии
idResult	INT	+	-	-	Уникален, генерируется автоматически (auto increment)	Идентификатор
description	VARCHAR(1000)	-	-	-	Символы латиницы и кириллицы, цифры, нижние подчёркивания, СиП; длина – от 1 до 1000 символов.	Описание
header	VARCHAR(45)	-	-	-	Любые символы, до 45	Заголовок
picture	VARCHAR(500)	-	-	-	Любые символы, до 500	Путь к картинке
Test_idTest	INT	-	+	-	on_delete=models.PROTECT on_update=models.CASCADE	Задание
start_condition	DOUBLE	-	-	-	Положительное число, до 100 включительно	Начальное условие
end_condition	DOUBLE	-	-	-	Положительное число, до 100 включительно	Конечное условие
correctness	BOOLEAN	-	-	False	Любые символы, до 45	Правильно ли пройден тест
date_created	DATE	-	-	-	Задаётся автоматически в момент добавление записи	Дата создания

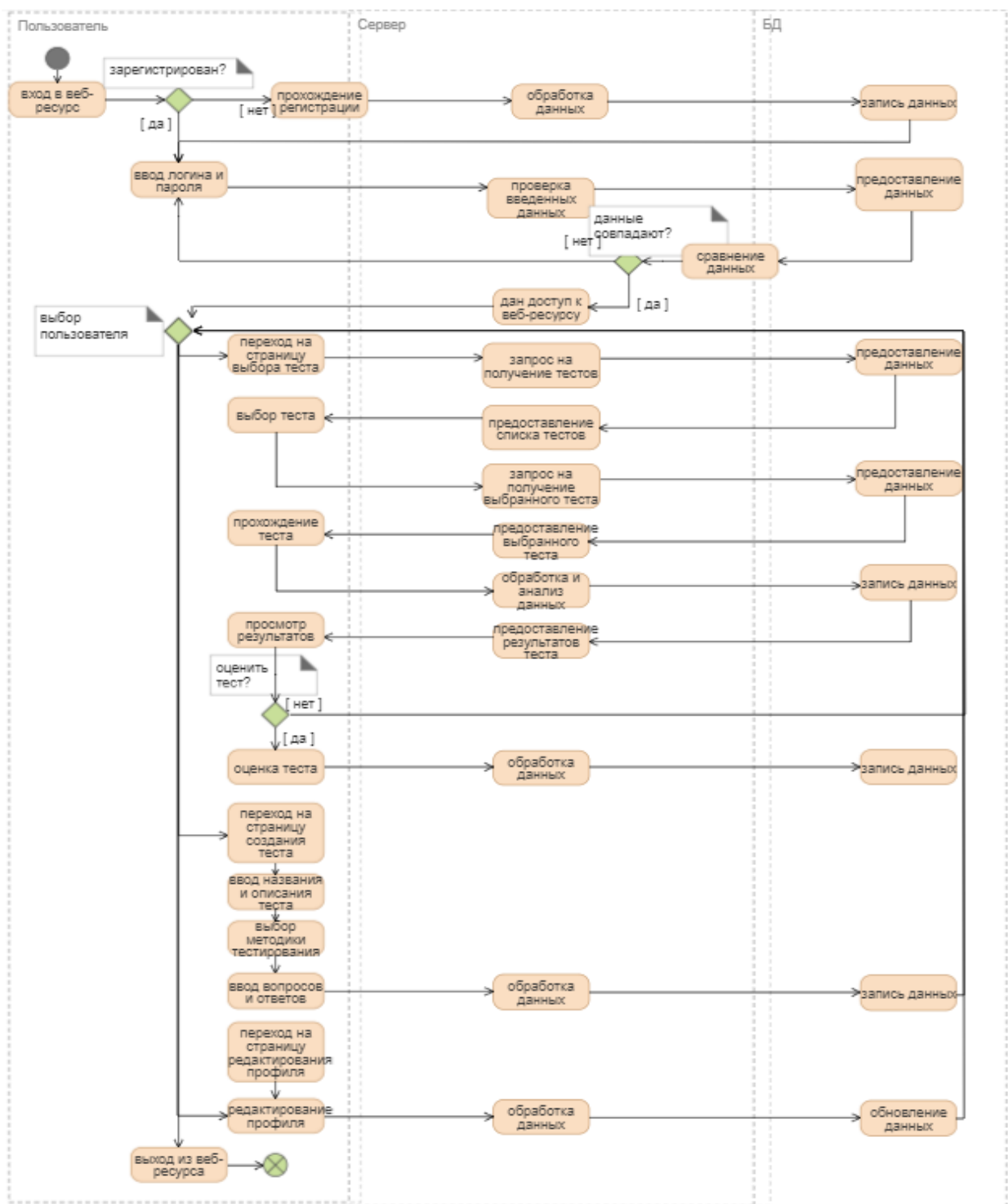


Рисунок Б.7 – Диаграмма активности Системы

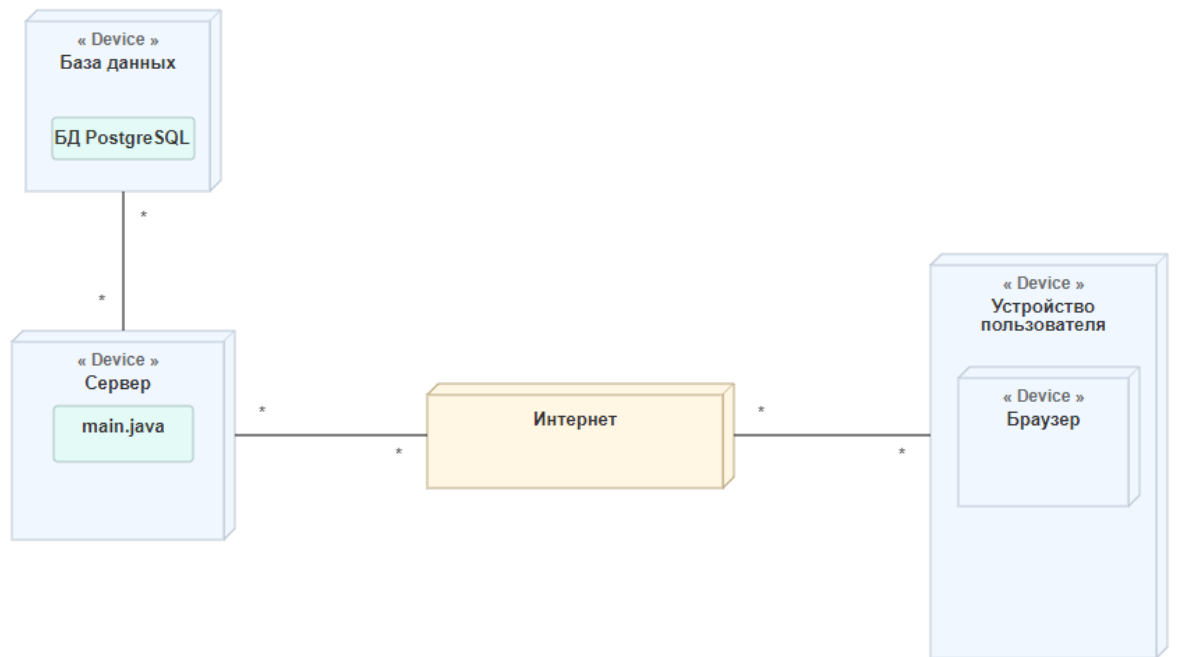


Рисунок Б.8 – Диаграмма развертывания Системы