Муниципальное автономное общеобразовательное учреждение   
«Средняя общеобразовательная школа № 146   
с углубленным изучением математики, физики, информатики» г.Перми

Научно-практическая работа на тему:

Разработка сайта «Системы счисления»

Выполнил ученик 8А класса

Фатыхов П.А.

Научный руководитель

учитель информатики

Нагибин С.С

Пермь, 2025

**Оглавление**

[**Введение** 3](#_Toc197446777)

[**1.** **Архитектура решения** 5](#_Toc197446778)

[**2.** **Разработка расчетных алгоритмов** 8](#_Toc197446779)

[**3.** **Оформление сайта (BackEnd+FrontEnd)** 11](#_Toc197446780)

[**3.1 Структура сайта** 11](#_Toc197446781)

[**3.2 Страница Переводчик** 12](#_Toc197446782)

[**3.3 Страница Калькулятор** 13](#_Toc197446783)

[**3.4 Страница Таблицы Умножения** 15](#_Toc197446784)

[**3.5 Страница Тренажер** 17](#_Toc197446785)

[**4.** **Тестирование сайта** 22](#_Toc197446786)

[**Заключение** 26](#_Toc197446787)

[**Список литературы** 27](#_Toc197446788)

# **Введение**

В настоящее время огромное количество пользователей по всему Интернету регулярно посещает различные сайты и веб‑страницы в поисках нужной им информации. Умение создавать сайты - это важный и востребованный (актуальный) навык. Меня эта тема интересует со стороны программирования, объектом моего изучения в этом проекте является разработка и тестирование сайта. В изученных источниках разработка сайтов описывается в большей степени со стороны задач дизайна и разработки, процесс тестирования для сайтов в открытых источниках рассматривается очень обобщенно и требует проработки и привлечения дополнительных источников для каждого вида тестирования. Выбор актуальных для проекта видов тестирования является отдельной интересной задачей.

Учебные сайты, которые помогают школьникам и студентам изучать материал по выбранной теме, часто используются как в отечественном сегменте сети Интернет, так и в зарубежной его части. Технологии и методы разработки сайтов тоже легко найти в открытом доступе.

В качестве темы сайта выбраны позиционные системы счисления. Существующие обучающие сайты, как правило, содержат или только тренажер, или калькулятор и переводчик, которые ограничены основаниями 2, 8, 10 и 16. Создание сайта с такими функциями, как переводчик, калькулятор, таблицы и тренажер, будет полезно для подготовки к ОГЭ, кроме того, работать с большим количеством различных систем намного интереснее. Использование фотографий с красивыми видами Пермской природы расширяет кругозор обучающихся и украшает сайт.

Цель работы – создать работоспособный сайт. Для достижения этой цели необходимо решить следующие задачи:

1. Разработать архитектуру и выбрать инструменты для реализации
2. Реализовать необходимые расчетные алгоритмы
3. Покрыть расчетные алгоритмы автотестами
4. Разработка собственно сайта
5. Функциональное тестирование сайта

Объектом исследования являются способы и методики создания сайта на Python. Предмет исследования – это технологии HTML, JS и CSS, программные продукты Flask, Jinja, JQuery, методы и инструменты тестирования. При подготовке реферата также дополнительно изучен инструмент для рисования диаграмм Mermaid.

Для изучения веб-технологий был выбран курс на платформе Stepik “Создание сайтов на Python Flask + HTML, CSS, JS” и ряд статей из сети Интернет. По результату изучения создан работающий сайт.

Работа представлена четырьмя главами. В главе Архитектура решения описаны инструменты и способы реализации, приведено обоснование выбранных инструментов для проекта. В главе Разработка расчетных алгоритмов описаны основные реализованные алгоритмы. В главе Оформление сайта указана информация по реализованным страницам сайта. В главе Тестирование рассмотрены основные методы тестирования, определены релевантные для проекта методы и дана информация п их применению в рамках проекта.

Работа представлена на 27 страницах, количество использованных источников литературы - 7.

# **Архитектура решения**

Предмет практической работы – сайт с 4 страницами:

* перевод из одной системы счисления в другую
* простой калькулятор в выбранной системе счисления
* таблицы умножения для различных систем счисления
* тренажер для отработки таблиц умножения с показом туристических видов Пермского края

Сайт представлен клиентской и серверной частью. Схема реализации сайта с перечислением основных использованных инструментов приведена на рисунке 1.

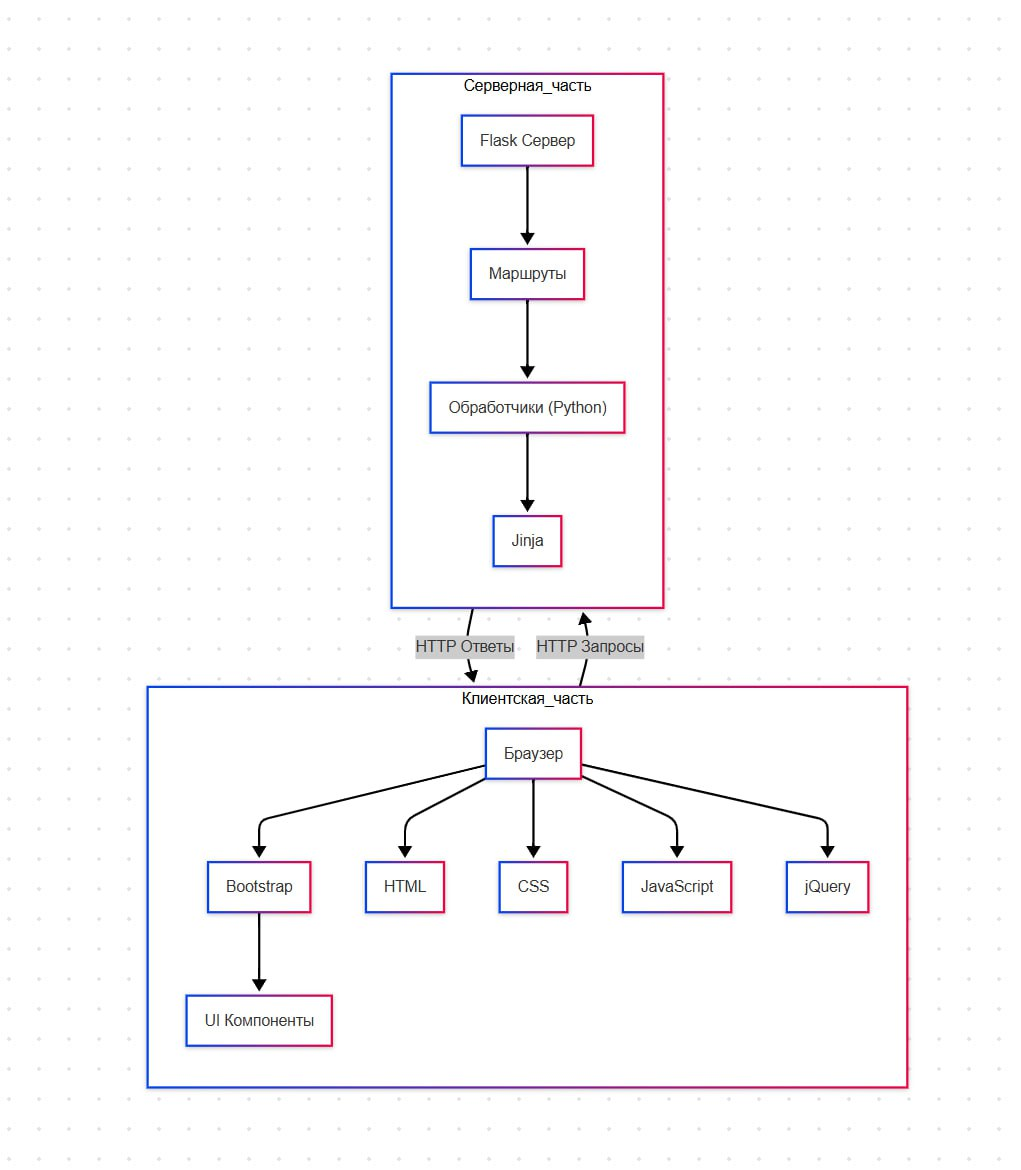


Рисунок . Архитектура проекта

Для создания сайтов могут использоваться различные инструменты. Обоснование их выбора для моего сайта представлено в таблице 1.

Таблица . Обоснование выбранных инструментов

|  |  |
| --- | --- |
| **Инструмент** | **Причина выбора инструмента** |
| Python | * Я в процессе изучения этого языка * Простота и читаемость. Python обладает понятным синтаксисом, что ускоряет разработку и поддержку кода * Поддержка сообщества. Python популярен, поэтому легко найти документацию и решения проблем |
| Flask (микрофреймворк для Python) | * Минималистичность. Flask предоставляет только необходимый функционал, без избыточности * Простота в изучении, отлично подходит для небольших и средних проектов |
| Jinja (шаблонизатор) | * Интеграция с Flask. Jinja является стандартным шаблонизатором для Flask * Динамические шаблоны. Удобный инструмент для встраивания Python-логики в HTML |
| HTML + CSS | * Стандарты веба, без них невозможно создать любой сайт * Стилизация. CSS необходим для визуального оформления |
| JavaScript (JS) | * Интерактивность – позволяет создавать динамические элементы (формы, анимации) * Универсальность – работает во всех браузерах |
| jQuery | * Упрощение JS-кода – jQuery сокращает количество кода для работы с DOM. * Кросс-браузерность – решает проблемы совместимости в старых браузерах |
| Bootstrap | * Готовые компоненты – иконки, кнопки, формы, навигация, сетка (grid). * Экономия времени – не нужно писать CSS с нуля. * Совместимость – работает с jQuery и современными браузерами. |

По результатам исследования выявлено, что выбранный стек (Python + Flask + Jinja + HTML/CSS + JS/jQuery + Bootstrap) хорошо подходит для создания небольших и средних веб-приложений, ориентированных на быструю разработку.

# **Разработка расчетных алгоритмов**

Для правильной работы сайта созданы различные расчетные функции, такие как переводчик и калькулятор. Все функции находятся в файле functions.py.

Одна из основных функций – это функция perevod, она обеспечивает перевод числа из одной системы счисления в другую (поддерживаются системы 2-36). Алгоритм расчета описан на схеме на рисунке 2. Суть алгоритма: Cначала производится валидация (проверка) входных данных. Если все успешно, проводим перевод входящего числа в 10-ю систему счисления, далее переводим из 10-й в нужную.

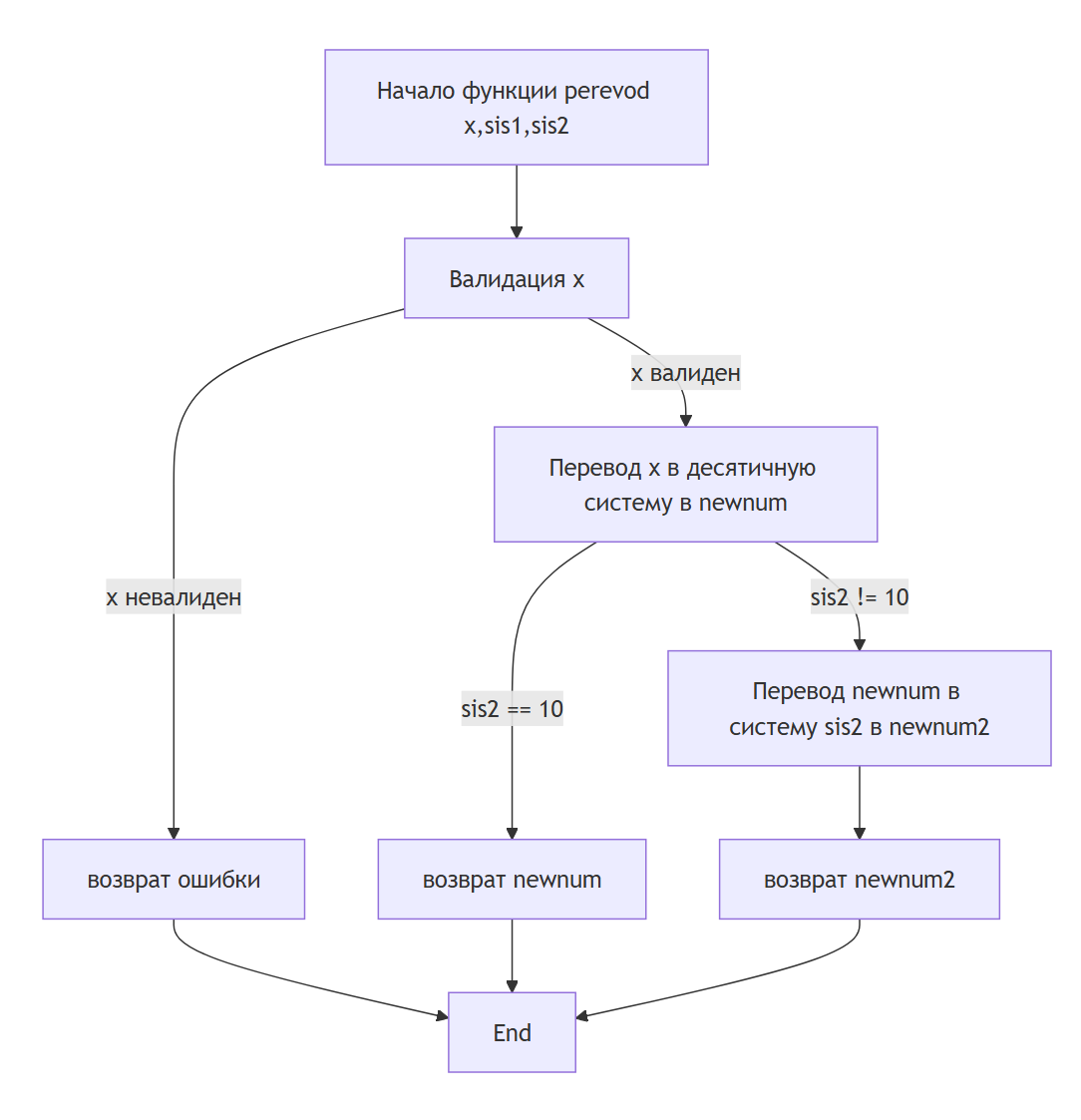


Рисунок . Функция перевода

Логика перевода в 10-ю систему описана на рисунке 3. Суть алгоритма: обрабатываем посимвольно, вычисляем значение текущей цифры (буквы) и умножаем его на систему счисления в степени равной номеру разряда с конца.

Алгоритм перевода из 10-й системы описан на рисунке 4. Суть алгоритма: пока число меньше системы счисления, делим число на систему счисления и получаем очередной разряд (остаток от деления).

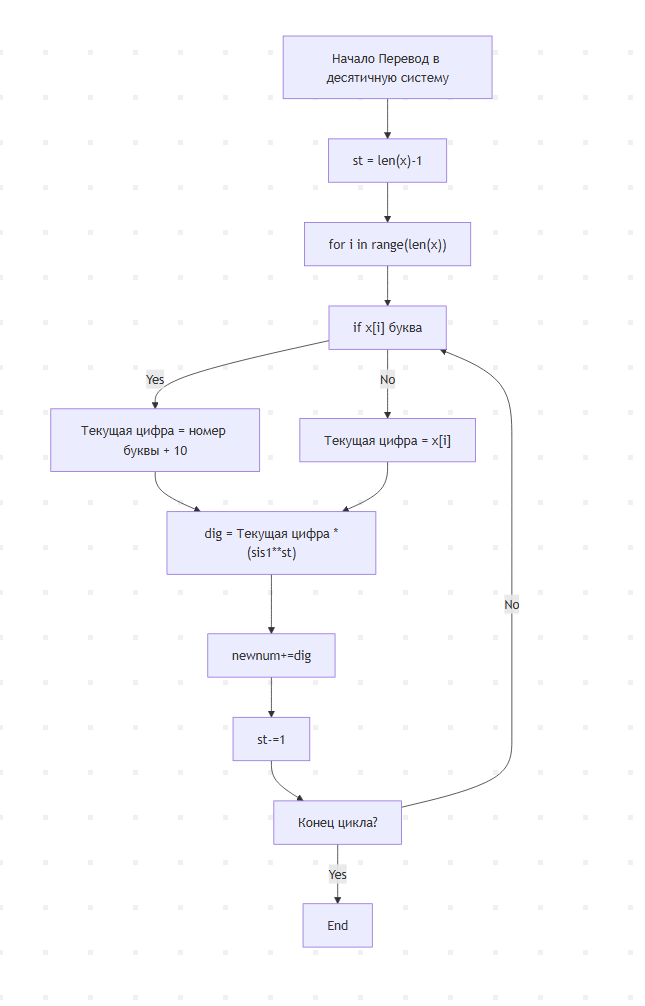


Рисунок . Перевод числа в 10-ю систему счисления

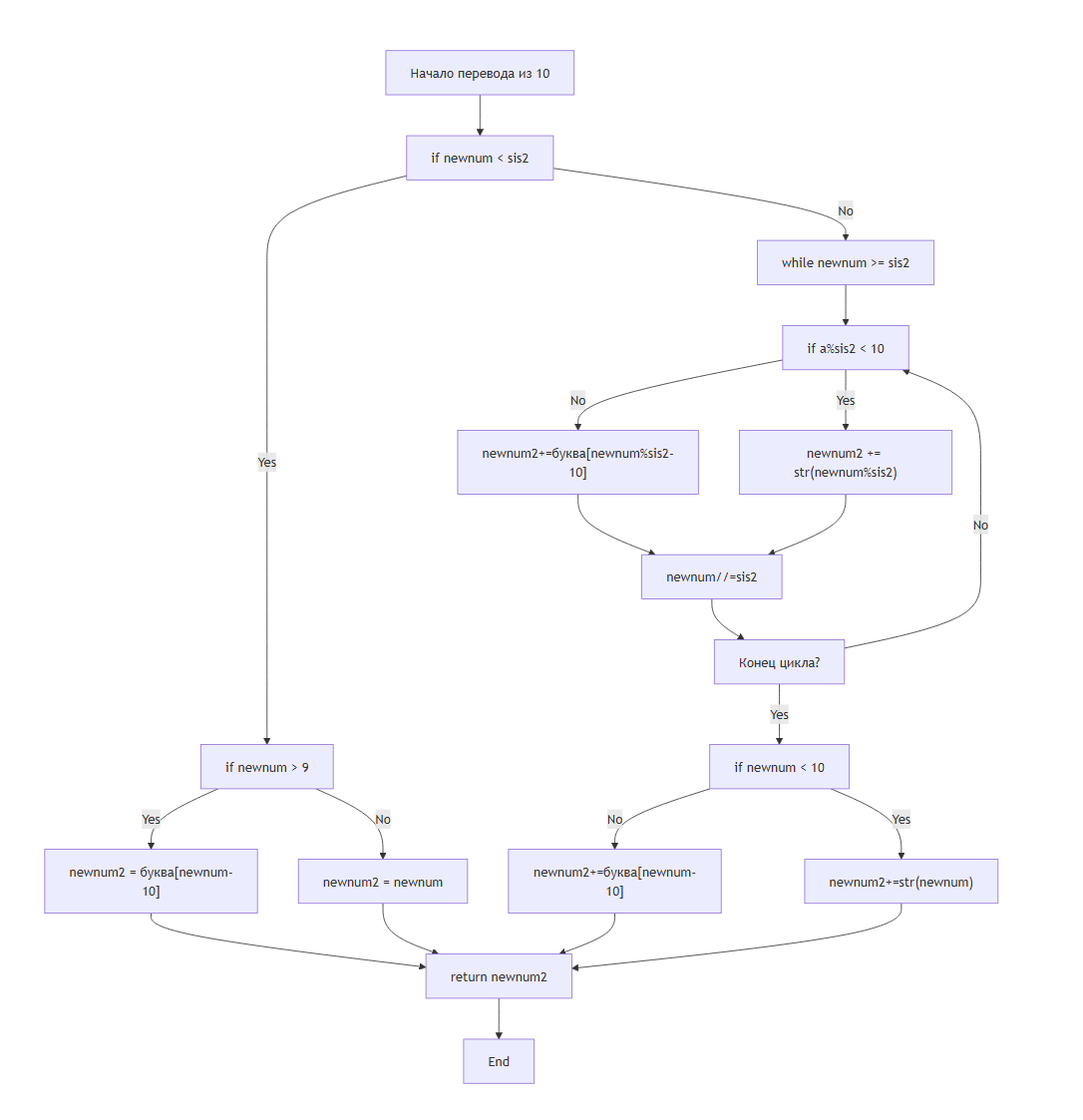


Рисунок . Перевод числа из 10-й системы

Вторая важная функция – это calculator. Эта функция обеспечивает выполнение простых операций над числами в выбранной системе счисления. Основана на функции perevod. Для выполнения операций числа переводятся в 10-ю систему, а ответ – в исходную. Основная функция калькулятора приведена на рисунке 5.

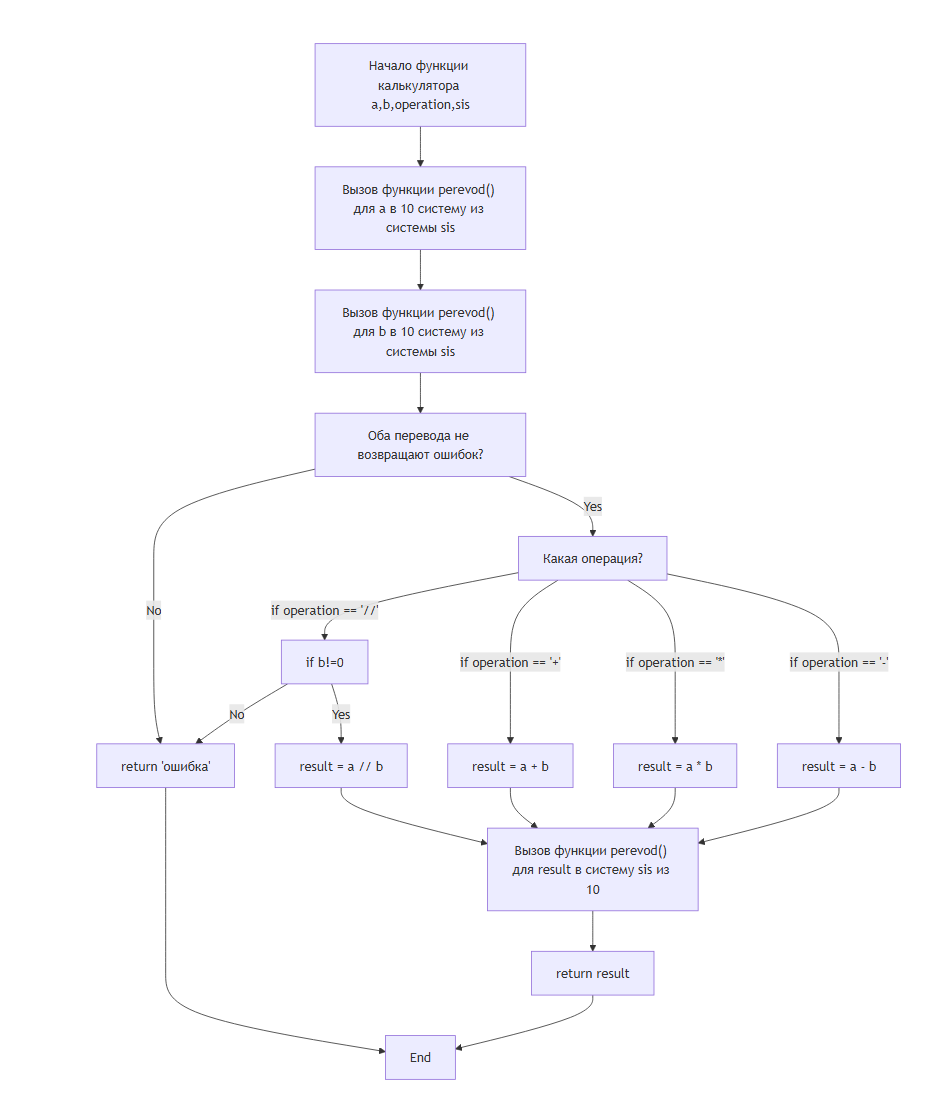


Рисунок . Функция калькулятора

Суть алгоритма: начинается функция с перевода обоих чисел из выбранной системы счисления в 10-ю. Далее производятся нужные вычисления и валидации и в конце обратное преобразование результата в выбранную систему счисления.

# **Оформление сайта (BackEnd+FrontEnd)**

## **3.1 Структура сайта**

Проект состоит из нескольких файлов, которые отвечают за работу сайта. HTML заготовки страниц находятся в папке /templates. CSS стили находятся в файле /static/style.css. JS код находится в файле /static/test.js. Картинки видов Пермского края находятся в папке /static/img4-20.jpg.

Сайт написан на фреймворке Flask для Python. Чтобы установить Flask (для запуска проекта на компьютере), необходимо выполнить:

**pip install Flask**

Проект выгружен на бесплатный хостинг для учебных Python проектов: <https://petyacalculator.pythonanywhere.com/>

Flask обеспечивает прием запросов с клиента (браузера), разбор адресной строки (маршрутизация запросов), генерация html страницы (с помощью шаблонизатора Jinja). Далее, html страница уходит на клиента (браузер) и отображается средствами браузера с учетом стилей и JS кода.

В данном проекте есть 4 страницы:

1. Переводчик: /per
2. Калькулятор: /calc
3. Таблицы умножения: /table
4. Тренажер (проверка знаний): /test

Маршрутизация и логика генерации страниц, а также логика обработки данных, введенных пользователем на сервере, находится в файле **app.py**

## **3.2 Страница Переводчик**

Страница Переводчик работает благодаря обмену информацией между Flask и клиентским браузером. Браузер запрашивает страницу /per, ему возвращается страница с полями ввода, заполненными по умолчанию. Алгоритм работы страницы указан на рисунке 6.

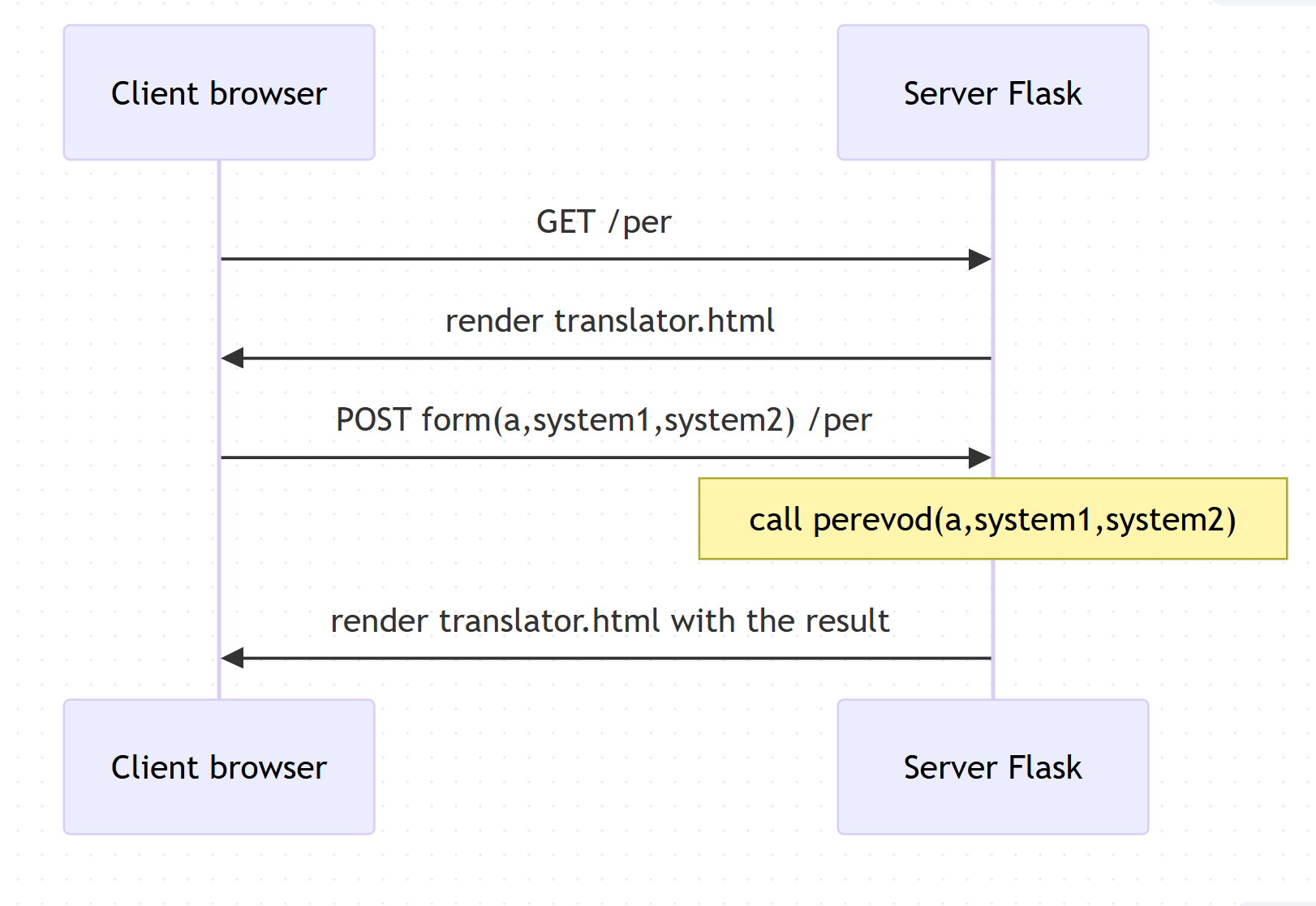


Рисунок . Работа страницы переводчик

Стартовый вид страницы Переводчик приведен на рисунке 7.

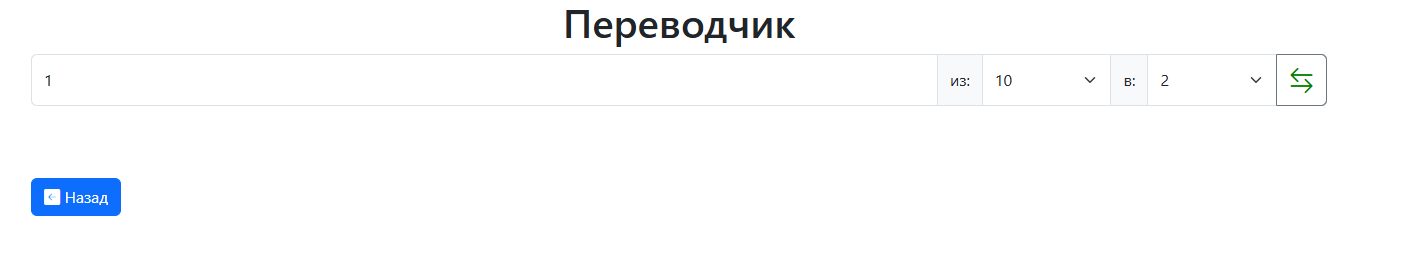


Рисунок . Страница Переводчик. Стартовый вид

Далее, пользователь вводит нужные ему значения и нажимает кнопку. Данные отправляются в виде POST формы на сервер, где производится перевод в нужную систему счисления (функция perevod) и возвращается страница с заполненным ответом:

Если успешно, то видим результат зеленым цветом (в генераторе html проставляется css класс “success”).

На рисунка 8 и 9 приведены варианты работы страницы Переводчик.

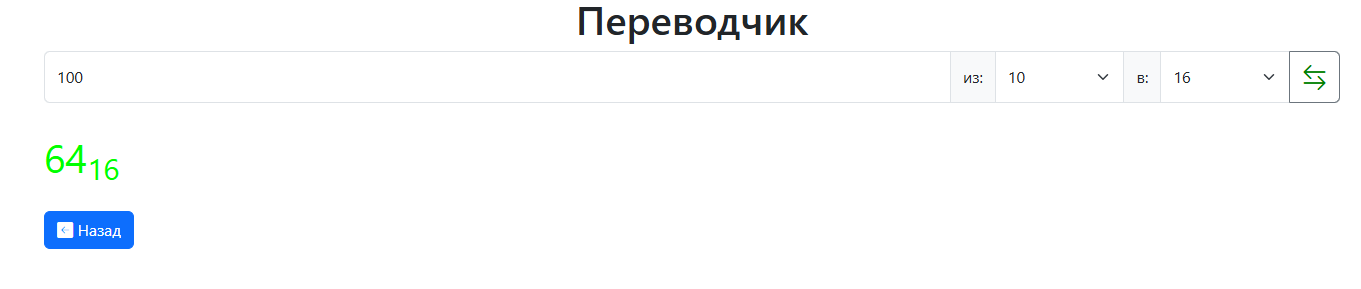


Рисунок . Страница переводчик. Вариант работы

Если есть ошибка введенных данных, видимо текст ошибки красным цветом (в генераторе html проставляется класс “notsuccess”):

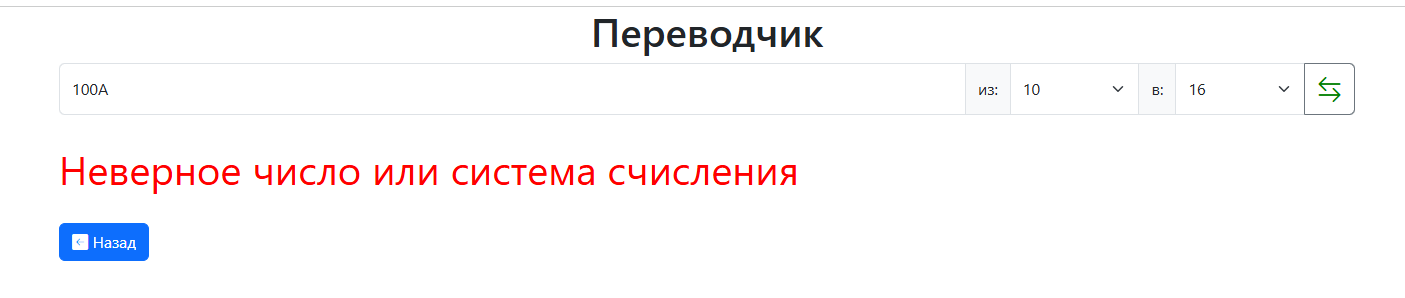


Рисунок . Страница Переводчик. Ошибка

## **3.3 Страница Калькулятор**

Страница Калькулятор работает благодаря обмену информацией между Flask и клиентским браузером. Браузер запрашивает страницу /calc, ему возвращается страница с полями ввода, заполненными по умолчанию. Алгоритм работы страницы указан на рисунке 10.

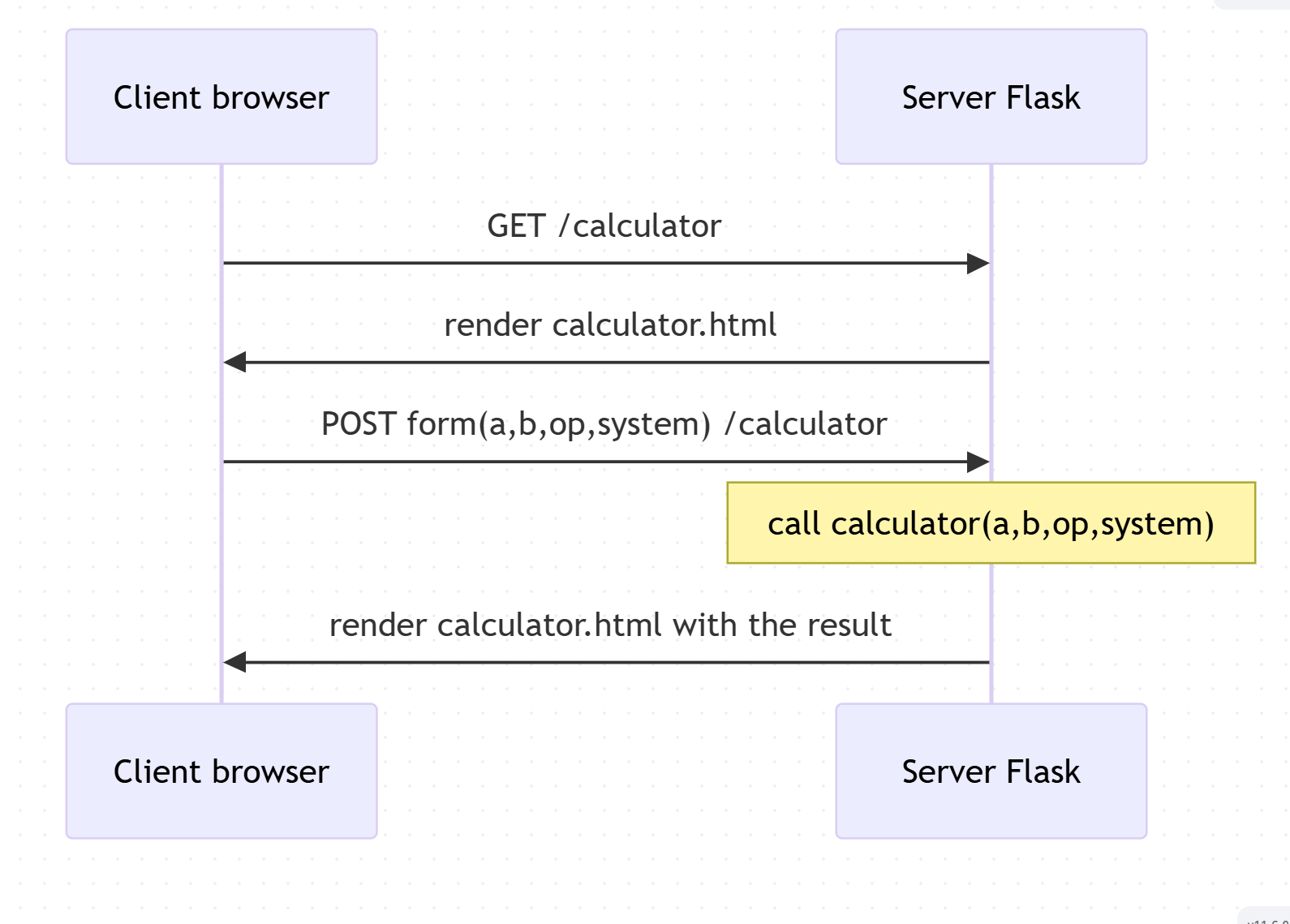


Рисунок . Работа страницы калькулятор

Стартовый вид страницы Калькулятор приведен на рисунке 11.

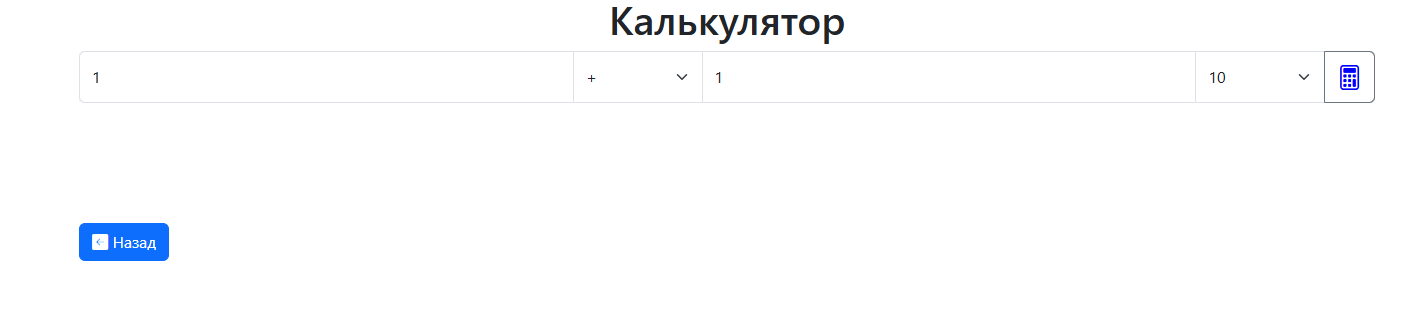


Рисунок . Калькулятор. Стартовый вид

На стартовом экране пользователь вводит нужные ему значения и нажимает кнопку. Данные отправляются в виде POST формы на сервер, где производится вычисление в нужной системе счисления (функция calculator) и возвращается страница с заполненным ответом:

Если успешно, то видим результат зеленым цветом (в генераторе html проставляется css класс “success”). Если есть ошибка введенных данных, видимо текст ошибки красным цветом (в генераторе html проставляется класс “notsuccess”).

Варианты обработки на странице Калькулятор приведены на рисунках 12 и 13.

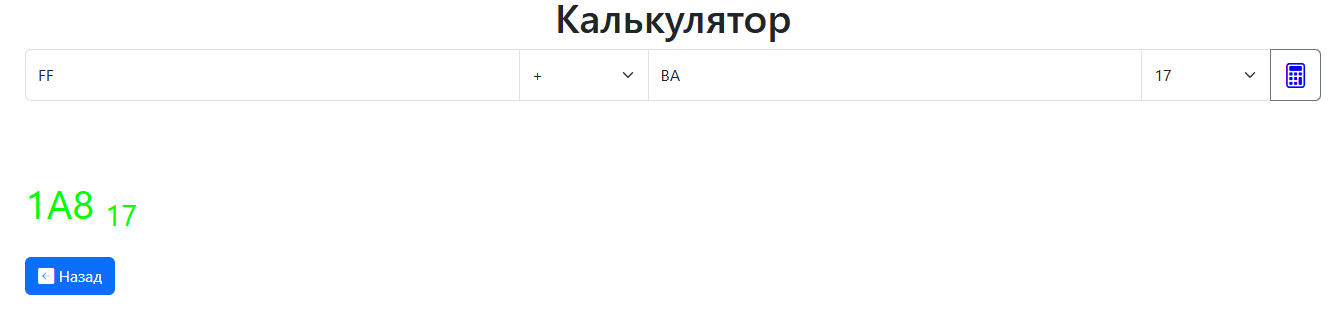


Рисунок . Калькулятор. Успешная обработка

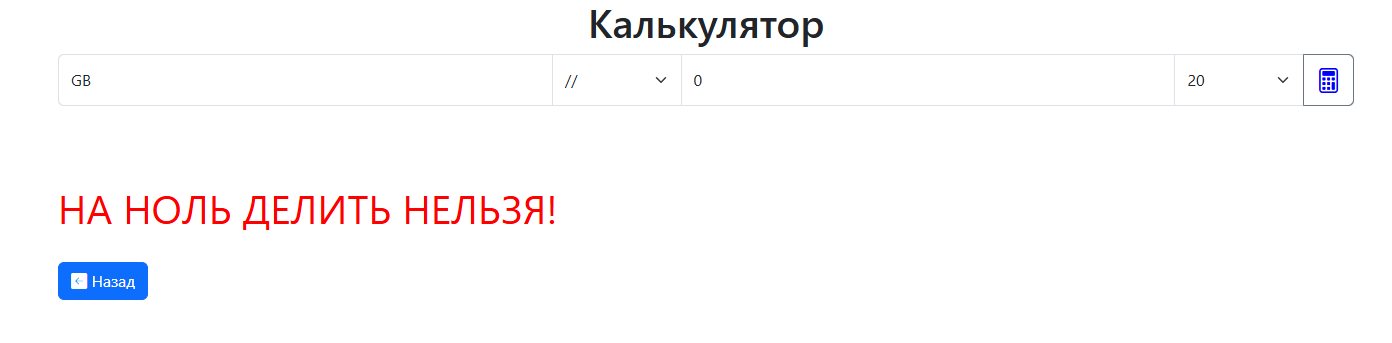


Рисунок . Калькулятор. Ошибка

## **Страница Таблицы Умножения**

Страница Таблицы умножения работает благодаря обмену информацией между Flask и клиентским браузером. Браузер запрашивает страницу /table, ему возвращается страница с выбором системы счисления. Алгоритм работы страницы указан на рисунке 14.

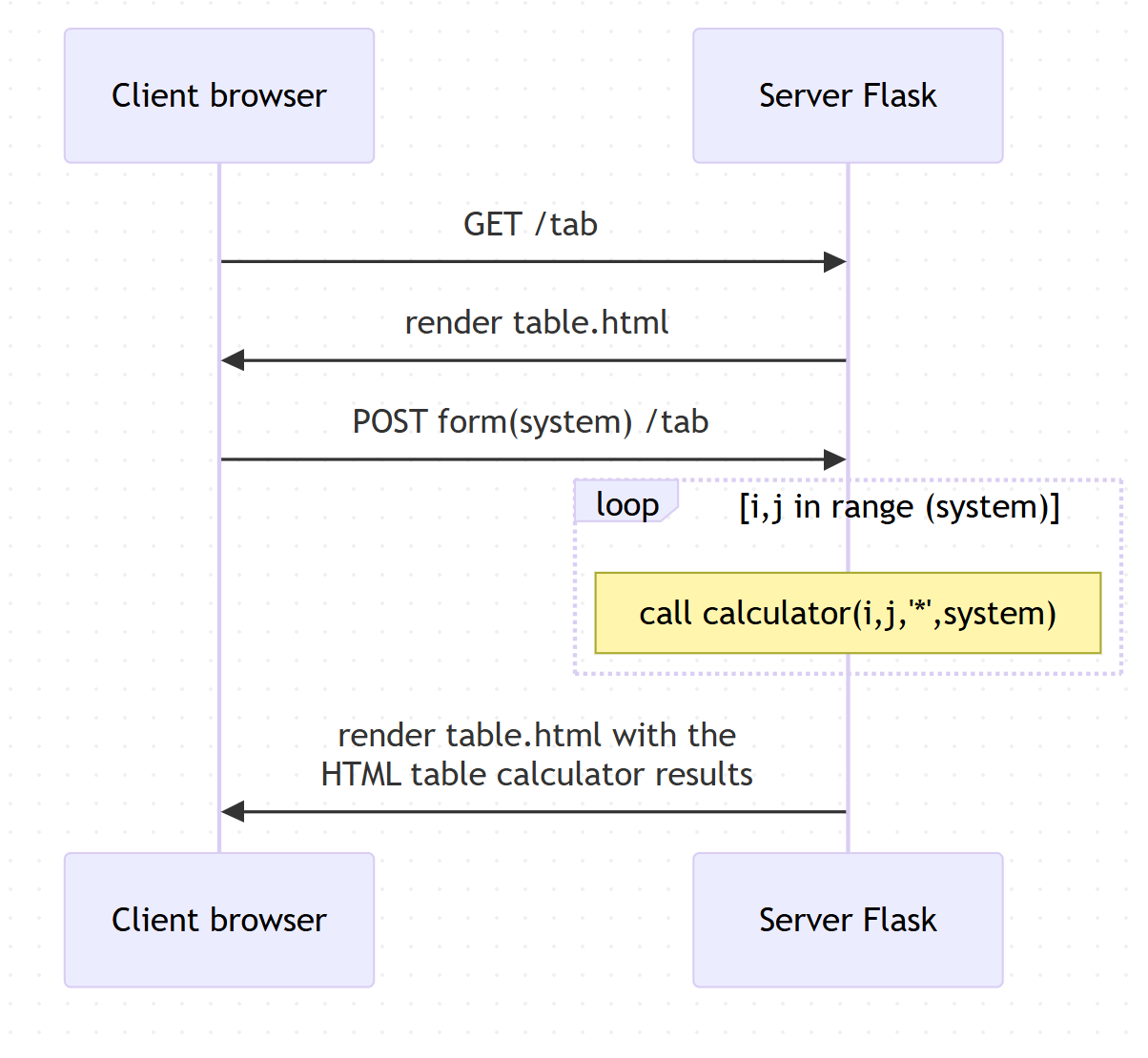


Рисунок . Работа страницы таблицы умножения

Стартовый вид страницы приведен на рисунке 15.

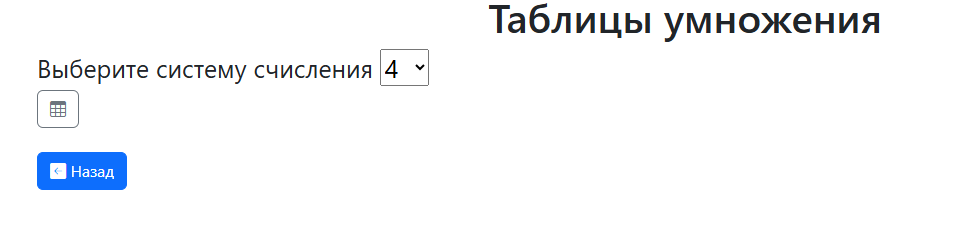


Рисунок . Таблицы умножения. Стартовый вид страницы

На стартовой странице пользователь вводит нужную систему счисления и нажимает кнопку. Данные отправляются в виде POST формы на сервер, где производится вычисление таблицы (всех возможных вариантов умножения чисел 1..system в нужной системе счисления (через функцию calculator) и возвращается страница с HTML таблицей для этой системы. На рисунке 16 приведен вариант отображения результата обработки для системы с основанием 16.



Рисунок . Таблицы умножения. Результат обработки

## **3.5 Страница Тренажер**

Страница Таблицы умножения работает благодаря обмену информацией между Flask и клиентским браузером. Браузер запрашивает страницу /test, ему возвращается страница с выбором системы счисления. Алгоритм работы страницы указан на рисунке 9.

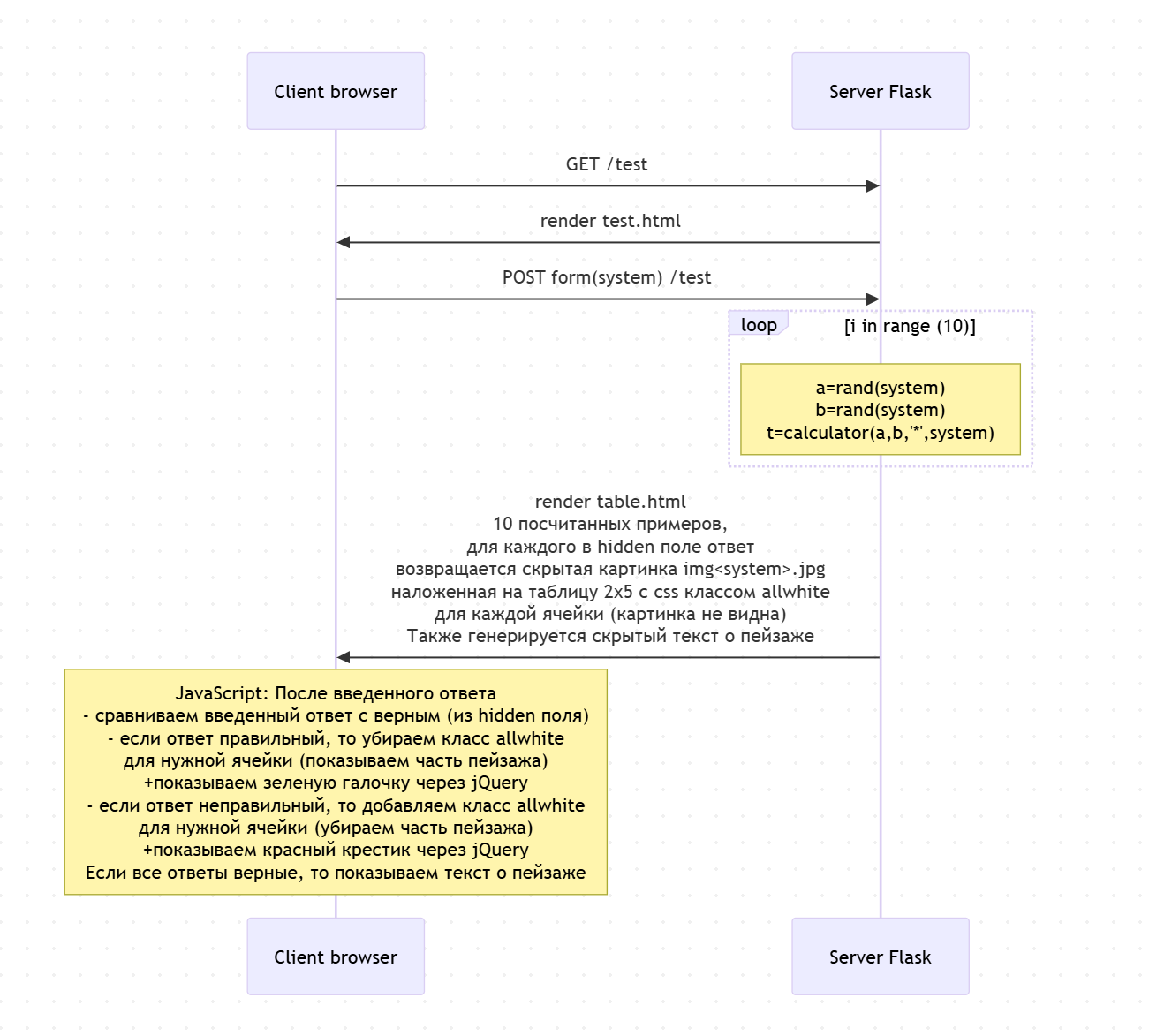


Рисунок . Работа страницы Тренажер

Стартовый вид страницы Тренажер приведен на рисунке 18.

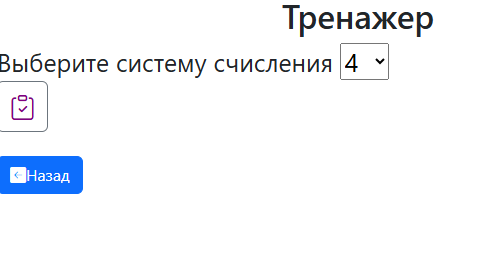


Рисунок . Тренажер. стартовый вид

После выбора, генерируются случайно 10 вопросов по таблице умножения. Рядом с полем ввода также генерируется скрытое (type=hidden) поле с правильным ответом. Также сразу отправляется в html нужная картинка пейзажа (зависит от системы счисления) в виде html таблицы с 10 ячейками. В начале, все ячейки таблицы имеют CSS класс “allwhite”, т.е. картинка не видна:

На рисунке 19 приведен пример сгенерированных вопросов.

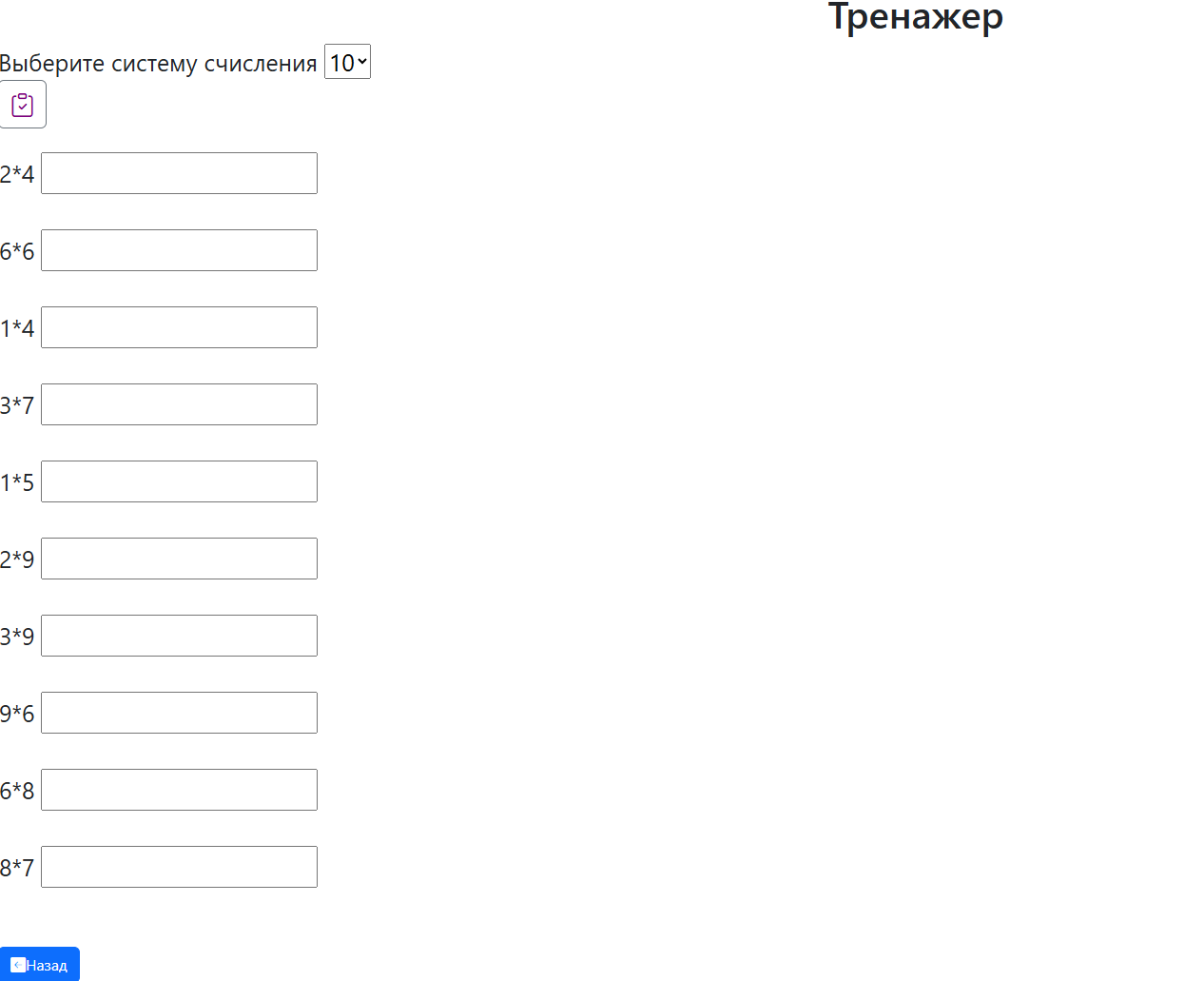


Рисунок . Пример вопросов

Далее на уровне JS и jQuery отслеживается изменение значения и для каждого изменения производится:

- проверяем, если ответ правильный, то открываем часть картинки (у нужной ячейки убираем класс “allwhite”) и отображаем скрытую зеленую галочку через jQuery

- проверяем, если ответ неправильный, то скрываем часть картинки (у нужной ячейки добавляем класс “allwhite”) и отображаем скрытую красный крестик через jQuery

- если все ответы правильные, также показываем скрытый изначально текст с описанием пейзажа.

На рисунках 20 и 21 приведены варианты генерации картинки при различных ответах пользователя.

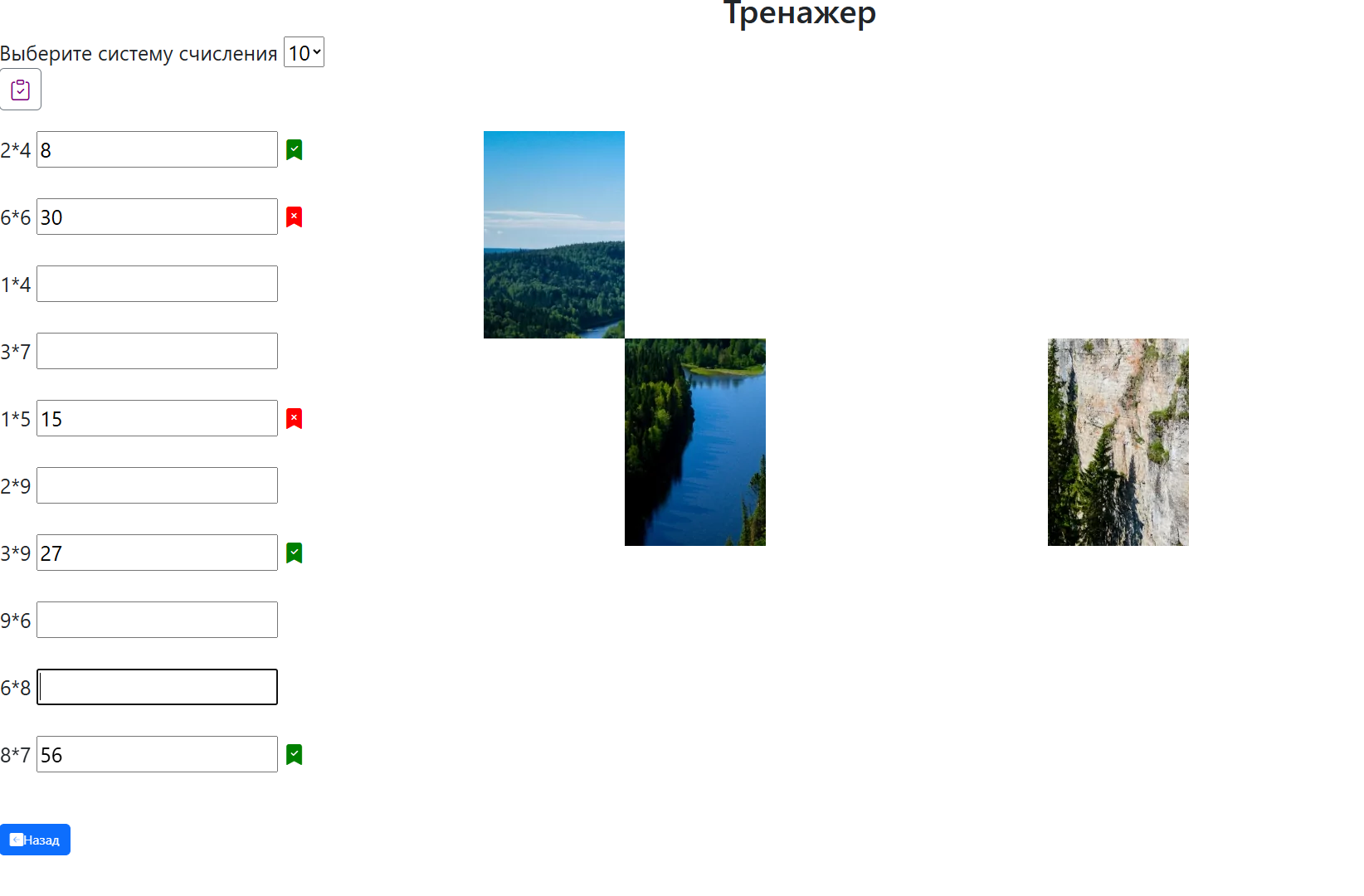


Рисунок . Тренажер. Ввод с ошибками

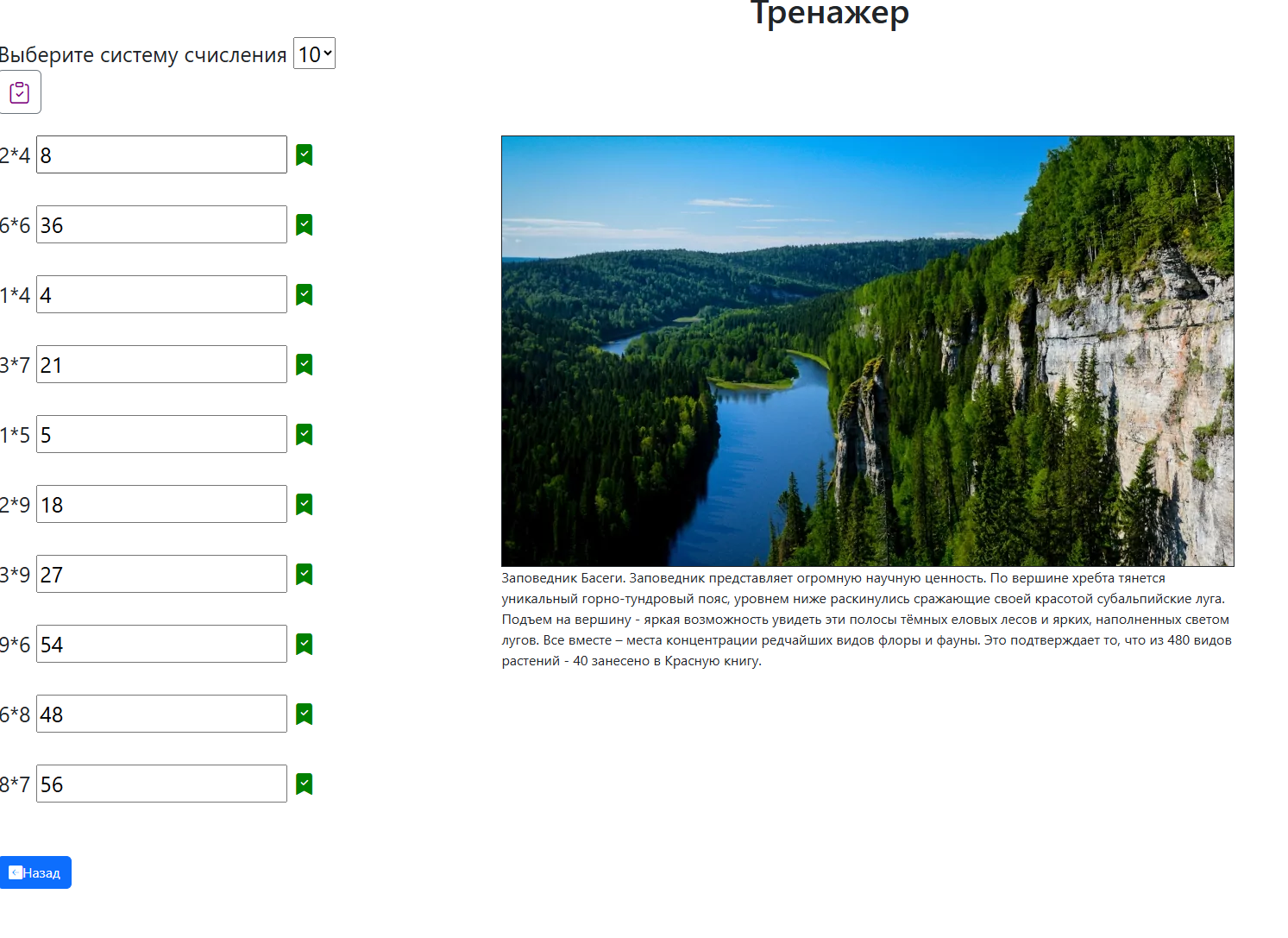


Рисунок . Тренажер. Ввод без ошибок

Для того, чтобы посмотреть все пейзажи, необходимо выполнить задания тренажера для каждой из систем счисления (4-20)

# **Тестирование сайта**

Целью тестирования программного обеспечения является обнаружение ошибок, дефектов и несоответствий в работе программы, а также для проверки соответствия программы требованиям заказчика.

Тестирование помогает установить надежность, стабильность и качество программы, а также повысить уровень удовлетворенности пользователей ее работой. [1]

Тестирование может происходить самыми разными способами, однако не стоит забывать о самом процессе и стратегии тестирования. От него зависит последовательность ваших действий. На сегодняшний день, специалисты по тестированию веб-сайтов применяют такие виды как:

* Функциональное тестирование
* Тестирования удобства пользования (юзабилити)
* Тестирование производительности
* Тестирование интерфейса пользователя (UI testing)
* Тестирование безопасности. [2]

[Функциональное тестирование](https://ibs-qa.ru/services/functional-testing/) — процесс проверки, как работает программное обеспечение или система в соответствии с заданными требованиями. Главная цель — убедиться, что все функции программы выполняются так, как ожидалось. [3]

[Функциональное тестирование](https://ibs-qa.ru/services/functional-testing/) можно делать вручную, но можно применять специальное ПО.

Тестирование удобства пользования (юзабилити) – это вид тестирования, который делает для сайта удобство и практичность в использовании. [2] В рамках проекта тестирование юзабилити не проводилось.

Тестирование производительности — в основном это нагрузочное тестирование. [Нагрузочное тестирование](https://ibs-qa.ru/services/highload/) оценивает производительность системы под определенной нагрузкой. Цель теста — проверить, как приложение или сервер реагируют на большое количество запросов, обрабатывают данные и поддерживают стабильную работу. [4] Разработанный сайт не предполагает использование большим количеством пользователем, тестирование производительности не проводилось.

Тестирование интерфейса пользователя - это тестирование графического интерфейса пользователя, которая предполагает проверить сайт на соответствия требованиям к графическому интерфейсу, профессионально ли оно выглядит, выполнено ли оно в едином стиле. [2] Тестирование интерфейса также за рамками данной работы.

Тестирование безопасности - «это ключ к надежности веб-сайтов. Основные правила этого тестирования — это проверка на уязвимость разных видов атак. [2] Данный учебный проект не предполагает промышленное использование и тестирование безопасности также не проводилось.

Таким образом, исходя из задач проекта выявлена полезность функционального тестирования, которое было проведено вручную и автоматически.

Ручное тестирование проводилось по сценариям тестирования.

Сценарий тестирования – это документ произвольной формы, в котором описаны основные шаги тестирования. Важный момент, что сценарий создается до начала тестирования и даже, в идеальном случае, до начала разработки. Пример используемого сценариев тестирования приведены в рисунке 22.

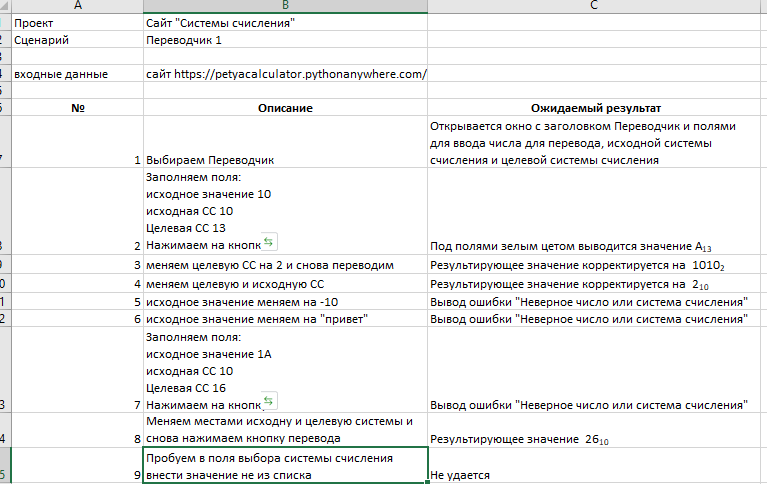


Рисунок . Пример сценария тестирования

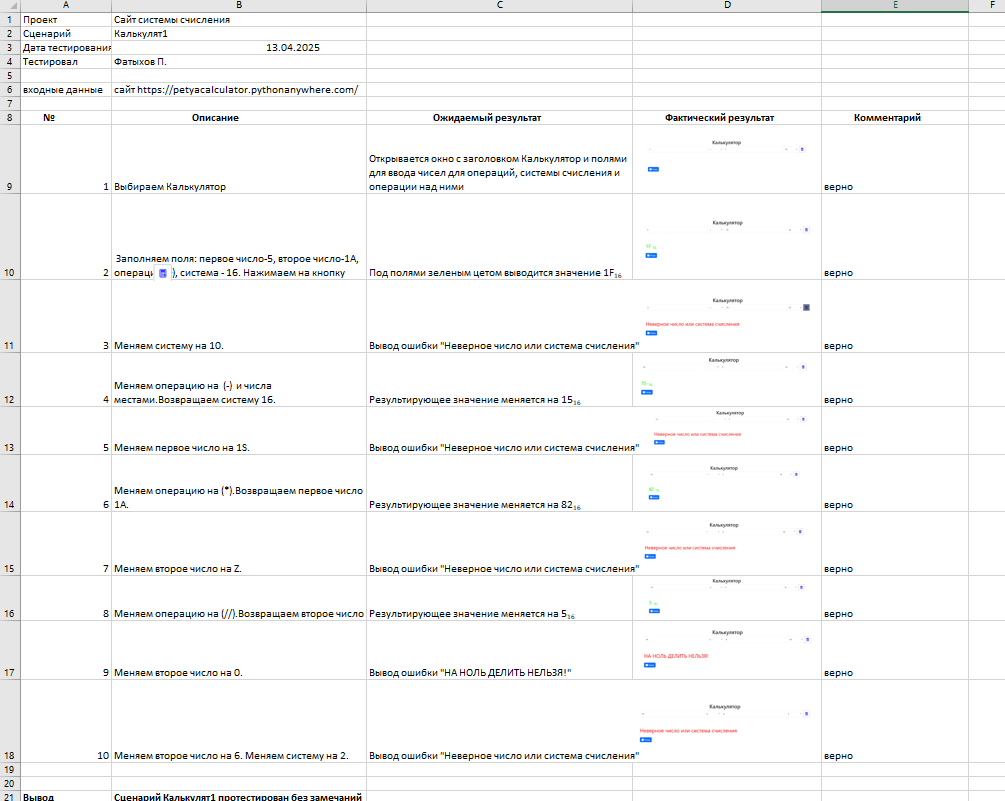
При прохождении сценария был подготовлен протокол тестирования. Протокол тестирования также является документом свободной формы, разрабатываем на базе сценария тестирования. Основное отличие сценария от протоколов тестирования, что в нем зафиксирован результат тестирования и, при наличии, ошибки и необходимые доработки. Пример протокола тестирования приведен на рисунке 23.

Рисунок 23. Пример протокола тестирования

Для тестирования расчетных алгоритмов были разработаны автотесты.

Автотесты имеют ряд преимуществ перед ручным тестированием:

1. **Раннее обнаружение ошибок** – проблемы выявляются на этапе разработки, а не в production.
2. **Упрощение отладки** – если тест падает, сразу понятно, где искать ошибку.
3. **Безопасность рефакторинга (изменения кода)**  – уверенность, что изменения не сломали существующую логику.
4. **Документирование кода** – тесты показывают, как должен использоваться модуль.
5. **Ускорение разработки** – меньше времени на ручное тестирование, быстрая проверка изменений.

Тесты – это различные варианты вызова функций, сделанные т.о., чтобы каждая строчка исходных функций была вызвана хотя бы раз.

В этом проекте все расчетные функции покрыты автотестами.

Для автотестирования python функций было использовано 2 инструмента -

модуль unittest для Python и инструмент coverage.py для анализа процента покрытия кода.

Для тестирования расчетных функций было написано 32 теста.

Результаты анализа покрытия кода (код приведен на рисунке 24) даны на рисунке 25.

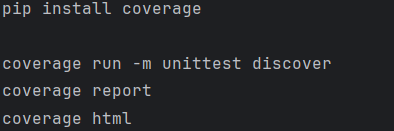


Рисунок . Код для анализа покрытия кода

В результате получили 100% покрытия тестами.

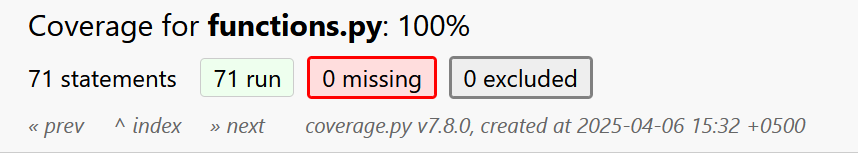


Рисунок . Результаты покрытия кода

По результатам полученного опыта тестирования был сделан следующий вывод: Автотесты в большей степени покрывают алгоритмическую часть реализации, функциональность доступная пользователю этим инструментом автоматического тестирования не проверяется. Ручные тесты, напротив, удобны для тестирования пользовательской функциональности.

# **Заключение**

Сайт по системам счисления - годовой проект, на котором реализованы 4 функции: Переводчик, Калькулятор, Таблицы и Тренажер, проведено тестирование ручным и автоматическим способами. Отличительной особенностью сайта являются виды Пермского края. При решении тестов в тренажере вам будут открываться части картинки (В каждой системе своя) и при завершении всех тестов в одной системе счисления появится название места и интересный факт о нём.

Разработка сайта довольно долгая работа, но она очень интересная и развивающая, а также очень востребованная в нашем современном мире. В процессе работы над проектом познакомился с технологиями: HTML, CSS, JS, Flask, инструментом для рисования диаграмм mermaid и инструментом для автоматического тестирования unittest для Python.

Для создания сайта я познакомился с понятием архитектуры программных систем, изучил технологии разработки и выбрал инструменты для реализации. Выбранный стек технологий оптимально подошёл для решаемой задачи.

Все необходимые расчетные алгоритмы реализованы и покрыты автотестами, проведен анализ покрытия автотестами, результат составил 100%.

Подготовлен сайт с необходимой функциональностью и размещен на бесплатном хостинге. Функциональность разработанного сайта протестирована с помощью технологии ручного тестирования.

В работе над проектом мне удалось попробовать свои силы в таких задачах создания сайтов как проектирование, разработка и тестирование. Такие задачи как дизайн, наполнение сайта, продвижение и поддержка не вошли в сферу интересов текущего проекта.

Все заявленные цели проекта выполнены, сайт работоспособен и может использоваться для обучения системам счисления, самообучения и подготовки к ОГЭ по информатике.

# **Список литературы**

1. Тестирование ПО: что это и зачем оно нужно [Электронный ресурс]: [Медиаблог](file:///C:\Users\tfatykhova\Downloads\Telegram%20Desktop\Медиаблог). – URL:<https://www.tgu-dpo.ru/news/2023/04/23/chto-takoe-testirovanie-po>
2. Нарыжный Д. Этапы тестирование вэб- сайтов [Электронный ресурс]: Сайт digital-агенства Art Lemon. – URL: <https://art-lemon.com/site-test>
3. Что такое функциональное тестирование [Электронный ресурс]: Блог IBS. – URL: <https://ibs-qa.ru/media/chto-takoe-funktsionalnoe-testirovanie/#block0>
4. Нагрузочное тестирование: что это и как его провести. Виды и инструменты [Электронный ресурс]: Блог IBS. URL: <https://ibs-qa.ru/media/nagruzochnoe-testirovanie-chto-eto-i-kak-ego-provesti-vidy-i-instrumenty/>
5. Python Unit Testing and Code Coverage [Электронный ресурс]:  контент-платформа LinkedIn Pulse.– URL: <https://www.linkedin.com/pulse/python-unit-testing-code-coverage-sourav-kumar-hazra>
6. The PythonAnywhere help pages [Электронный ресурс]:  Справочник по среде PythonAnywhere.– URL: https://help.pythonanywhere.com/pages/
7. Diagram Syntax. Mermaid [Электронный ресурс]:  Справочник по среде рисования диаграмм Mermaid.– URL: https://mermaid.js.org/intro/syntax-reference.html