

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка веб-приложения для смены
кодировок**

Студент _____ Мусатов Владислав Владимирович
(подпись, дата) (ФИО полностью)

Руководитель _____ Кольчугина Е.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

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

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Мусатов Владислав Владимирович гр. 22ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Разработка веб-приложения для смены
кодировок

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

2. Руководитель работы Кольчугина Е. А.

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

Назначение: Обеспечение конвертации поврежденного текста в исходный с возможностью
хранения информации в журнале в личном кабинете
пользователя

Основные функции системы:

1. Авторизация пользователей
2. Конвертация текста через веб-интерфейс
3. Хранение истории конвертаций в журнале
4. Пакетная загрузка файлов
5. Поддержка пользовательских таблиц
6. Каскадное восстановление текста
7. Введение журнала ошибок
8. Автоопределение кодировки
9. Загрузка единичных файлов

Технология разработки: объектно-ориентированный и модульный подход клиент-серверной и веб-архитектуры

Языки программирования: Perl

Среда программирования: Visual Code

4. Перечень подлежащих разработке вопросов:

1. анализ предметной области и постановка задачи;

2. проектирование программных средств;

3. реализация программных средств;

4. тестирование программных средств;

5. оформление ВКР.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 77 листов, 10 таблиц, 19 рисунков, 20 использованный источников и 3 приложения

ЗАЩИТА ДАННЫХ, WEB-ПРИЛОЖЕНИЕ, PERL, SQLITE, JavaScript, HTML, CSS, HTTP, WINDOWS, АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР, БАЗЫ ДАННЫХ, BOOTSTRAP.

Объект разработки: Разработка веб-приложения для смены кодировок

Предмет разработки: разработка веб-приложения для конвертации поврежденных данных в текст, хранение данных в журнале в собственной учетной записи.

Цель работы: разработать программное обеспечение для реализации веб-приложения для смены кодировок.

Результаты работы: Спроектировано веб-приложение для смены кодировок.

Средства разработки: Mojolicious, SQLite, Perl, HTML, CSS, JavaScript

					ПГУ 09.03.02 – 08BP221.17 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Разработка веб-приложения для смены кодировок	Лит.	Лист	Листов
Разраб.		Мусатов В.В.						
Провер.		Кольчугина Е.А.					4	77
Реценз.						группа 22ВИ1		
Н. Контр.		Такташикн Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	6
1 Постановка задачи и анализ требований	7
1.1 Постановка задачи	7
1.2 Анализ программного обеспечения аналогичного назначения	7
1.3 Формулировка и анализ требований к разрабатываемому программному обеспечению	9
2 Проектирование	13
2.1 Проектирование бизнес-процессов системы	13
2.2 Проектирование базы данных	15
2.3 Проектирование интерфейса	18
3 Программная реализация	23
3.1 Программная реализация клиентской части системы	23
3.2 Программная реализация серверной части.....	26
3.3 Программная реализация базы данных	29
4 Тестирование	31
4.1 Расчет метрик кода	31
4.2 Нагрузочное тестирование	35
4.3 Тестирование интерфейса веб-приложения.....	37
5 Планирование и расчёт бюджета разрабатываемой системы	40
Заключение	43
Список использованных источников	45
Приложение А (рекомендуемое) Программный код – клиентская часть	47
Приложение Б (рекомендуемое) Программный код – серверная часть	58
Приложение В (рекомендуемое) Тестирование интерфейса.....	73

Введение

В современном мире глобализации информация является главным ресурсом. Ежедневно между различными информационными системами, серверами и пользовательскими устройствами передаются огромные объемы текстовых данных.

Несмотря на переход большинства современных платформ на универсальный стандарт Unicode, проблема несовместимости кодировок остается крайне острой. При интеграции унаследованных систем, переносе старых баз данных бухгалтерского или складского учета, или сбоях в HTTP-заголовках при передаче данных по сети, пользователи регулярно сталкиваются с искажением текстовой информации. Текст, сохраненный в одной кириллической кодировке часто ошибочно интерпретируется в другой, что приводит к появлению нечитаемых символов. Кроме того, специфика работы веб-технологий и передачи данных по сети требует регулярного преобразования информации в специализированные транспортные форматы.

Для решения комплекса этих проблем пользователям, системным администраторам и разработчикам необходим быстрый, надежный и всегда доступный инструмент. Этим инструментом является специализированное веб-приложение - универсальный конвертер текстовых данных. В отличие от десктопных программ, веб-приложение не требует установки, не зависит от операционной системы пользователя и позволяет мгновенно восстановить читаемость текста прямо в браузере. Разработка подобной информационной системы, обладающей высокой скоростью обработки байтовых потоков, интуитивно понятным интерфейсом, системой авторизации и функциями аудита (сохранения истории операций в базу данных), является крайне актуальной прикладной задачей. Именно поэтому тема разработки веб-приложения для восстановления и смены кодировок текстовых данных была выбрана в качестве темы данной бакалаврской работы.

1 Постановка задачи и анализ требований

1.1 Постановка задачи

Основной задачей данной выпускной квалификационной работы является разработка веб-приложения для восстановления и конвертации текстовых данных и их форматов. Выбор веб-среды для реализации подобной системы обусловлен высокой степенью доступности: пользователю не требуется устанавливать стороннее программное обеспечение на свой персональный компьютер, достаточно лишь наличия современного веб-браузера. Использование протокола HTTP обеспечивает универсальный интерфейс взаимодействия «клиент-сервер», позволяющий надежно передавать текстовые данные и потоки байтов для их последующей обработки на стороне сервера. Проектируемая система должна обеспечить реализацию восстановления поврежденного кириллического текста из неверно интерпретированных кодировок Windows-1251 и KOI8-R, двухстороннее преобразование бинарных и текстовых данных в формат Base64 и двухсторонняя переадресация URL. Фреймворк Mojolicious предоставляет мощную и масштабируемую архитектуру MVC (Model-View-Controller), обладает встроенным неблокирующим веб-сервером и мощным маршрутизатором запросов. Язык Perl, в свою очередь, является признанным стандартом в области обработки текстов и регулярных выражений. Использование базового модуля Encode позволяет осуществлять низкоуровневые манипуляции с байтами и кодировками на скоростях, значительно превышающих аналогичные решения на высокоуровневых скриптовых языках. Выбор СУБД SQLite продиктован необходимостью создания легковесного, переносимого хранилища данных, не требующего развертывания отдельного сервера баз данных.

1.2 Анализ программного обеспечения аналогичного назначения

В рамках исследования существующих решений были проанализированы три популярных веб-ориентированных сервиса, предоставляющих услуги по

работе с кодировками текста: Декодер студии Артемия Лебедева, Convertio и Involta.

Декодер студии Артемия Лебедева (ДстАЛ) - специализированный онлайн-сервис, созданный для расшифровки нечитаемого текста. Хорошо справляется с исправлением «кракозябр» во вставленном тексте, перебирая популярные комбинации (например, из UTF-8 в Windows-1251). Однако сервис имеет жесткие ограничения: он не поддерживает загрузку файлов, полностью лишен функций пакетной обработки архивов и не предоставляет пользователю личного кабинета [1].

Convertio (универсальный конвертер файлов) - крупный международный облачный сервис для конвертации файлов различных форматов, включая текстовые документы. Обладает мощным API и современным интерфейсом, поддерживает загрузку файлов. Главный недостаток в рамках нашей предметной области - полное отсутствие механизмов автоматического подбора кодировки при глубоком повреждении текста [2].

Involta представляет собой современный веб-инструмент, предназначенный для оперативного восстановления текстовых данных, искаженных в результате некорректной интерпретации исходных кодировок (появления «кракозябр» при передаче информации). Из недостатков сервис ориентирован исключительно на индивидуальную работу со строками через буфер обмена. В системе полностью отсутствует возможность пакетной загрузки структурированных ZIP-архивов с автоматическим сохранением дерева каталогов и кодированием сотен файлов «на лету» [3].

Сравнение по ключевым критериям приведено в таблице 1.

Таблица 1 – Сравнение разрабатываемой системы с аналогичными решениями

Критерий сравнения	(ДстАЛ)	Convertio	Involta	Разрабатываемая система
Обработка интерактивного текста	+	-	+	+

Продолжение таблицы 1

Критерий сравнения	(ДстАЛ)	Convertio	Involta	Разрабатываемая система
Загрузка единичных файлов (.txt, .csv, .log)	-	+	-	+
Пакетная обработка данных (загрузка ZIP-архивов)	-	+	-	+
Определение кодировки	+	-	+	+
Модуль «Реаниматор»	+	-	+	+
Личный кабинет	-	+	-	+
Ведение журнала ошибок	-	+	-	+
Поддержка пользовательских таблиц	-	-	-	+
Бесплатный доступ без ограничений на объем	+	-	-	+
Требования к СУБД	-	-	+	+

Анализ таблицы 1 демонстрирует, что проектируемое веб-приложение призвано объединить сильные стороны файловых процессоров и декодеров, устранив ограничения по объему данных.

1.3 Формулировка и анализ требований к разрабатываемому программному обеспечению

Проектирование любой информационной системы начинается со сбора, анализа и формализации требований. Основанием для разработки универсального веб-приложения конвертации и восстановления кодировок является необходимость автоматизации процессов обработки текстовой информации, деформированной в результате сбоев передачи данных, неверной интерпретации байтовых потоков или многократного каскадного кодирования. В результате, был сформирован комплекс технических и эксплуатационных

требований к веб-приложению. Диаграмма вариантов использования приведена на рисунке 1.

ис [Use Case Diagram1] /

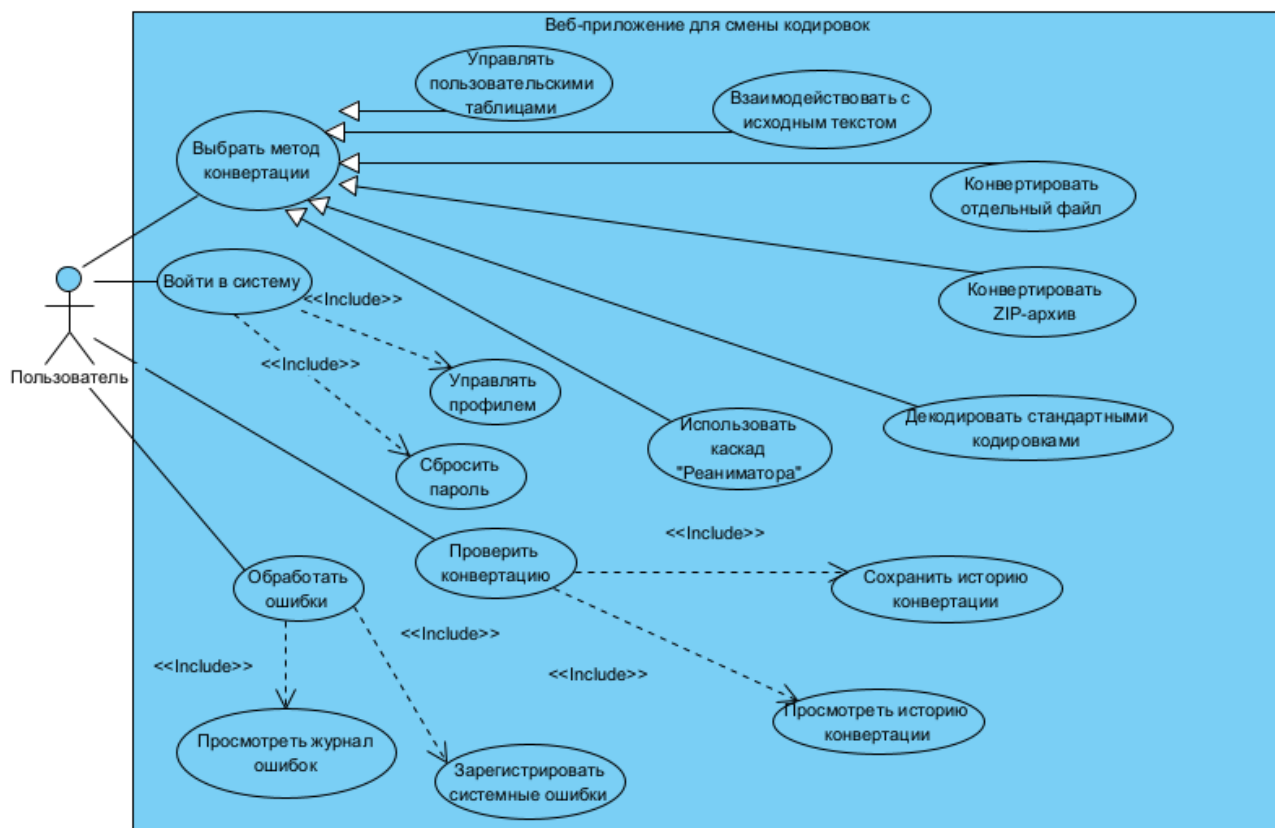


Рисунок 1 – Диаграмма вариантов использования

Для обеспечения высокой надежности, масштабируемости и удобства использования (UX) к проектируемой системе предъявляется комплекс требований, который разделяется на две основные группы: функциональные требования и нефункциональные требования.

Функциональные требования определяют перечень конкретных задач и сервисов, которые программное обеспечение должно выполнять для удовлетворения потребностей конечного пользователя. В соответствии с системным анализом предметной области, разработанное ПО должно обеспечивать выполнение следующих функций:

- приложение должно поддерживать асинхронный ввод текстовых данных пользователем с мгновенной отправкой порций данных на сервер через

AJAX/Fetch API и динамическим обновлением поля результата без перезагрузки всей веб-страницы;

- обеспечение загрузки локальных файлов форматов .txt, .csv, .log через интерфейс multipart/form-data, их потоковое чтение в оперативную память сервера, перекодирование и автоматическое возвращение пользователю в виде скачиваемого файла;

- поддержка загрузки ZIP-архивов, их динамическая распаковка в памяти, итерационный обход вложенных файлов, применение выбранных алгоритмов конвертации ко всему пулу документов и сборка результирующего ZIP-архива для скачивания.;

- реализация алгоритма частотного анализа распределения байт кириллических символов (диагностика популярных гласных букв в различных кодовых страницах) для автоматического распознавания исходного формата без участия пользователя;

- предоставление пользователю инструмента для конструирования цепочек последовательных трансформаций для восстановления многократно поврежденных или вложенных данных;

- возможность загрузки пользовательских текстовых конфигураций подстановки символов формата ключ=значение, их представления в формат JSON и сохранение в СУБД для обработки нестандартных кодировок или шифров;

- аутентификация пользователей, ведение персонализированного журнала истории транзакций, сбор агрегированной статистики использования системы и предоставление функций администрирования профиля (смена пароля, конфиденциальная очистка истории).

Нефункциональные требования накладывают ограничения на эксплуатационные характеристики системы, системную архитектуру, производительность и безопасность:

- время отклика системы при интерактивной AJAX-обработке строк объемом до 10 Кб не должно превышать 400 мс;

- обработка файлов и архивов должна выполняться в потоковом режиме непосредственно в оперативной памяти серверной части без генерации избыточных «тяжелых» временных файлов на жестком диске, что снижает нагрузку на дисковую подсистему ввода-вывода;
- приложение должно быть устойчивым к некорректным действиям пользователя;
- система не должна аварийно завершать процесс серверной части, все исключения должны перехватываться защищенными блоками eval;
- любая перехваченная системная ошибка должна автоматически регистрироваться в специализированном журнале БД error_log с фиксацией временной метки, имени вызвавшего ее программного модуля и описания исключения;
- пароли пользователей не должны храниться в базе данных в открытом виде. Обязательно применение криптографического хеширования по алгоритму SHA-256;
- все пользовательские сессии должны быть защищены, доступ к функционалу конвертера, истории и профиля изолируется промежуточным слоем проверки авторизации;
- пользовательский интерфейс должен быть полностью адаптивным (компоновка на базе фреймворка Bootstrap 5) для корректного отображения на экранах с различным разрешением (десктопные мониторы, планшеты, смартфоны).

2 Проектирование

2.1 Проектирование бизнес-процессов системы

Визуализация и структурирование алгоритмической деятельности веб-приложения позволяет определить точки сопряжения между пользователем, программным ядром Engine.pm и реляционной базой данных под управлением СУБД SQLite. Основу автоматизируемой предметной области составляют процессы ветвления входящих информационных потоков и их последовательная трансформация. Диаграмма деятельности приведена на рисунке 2.

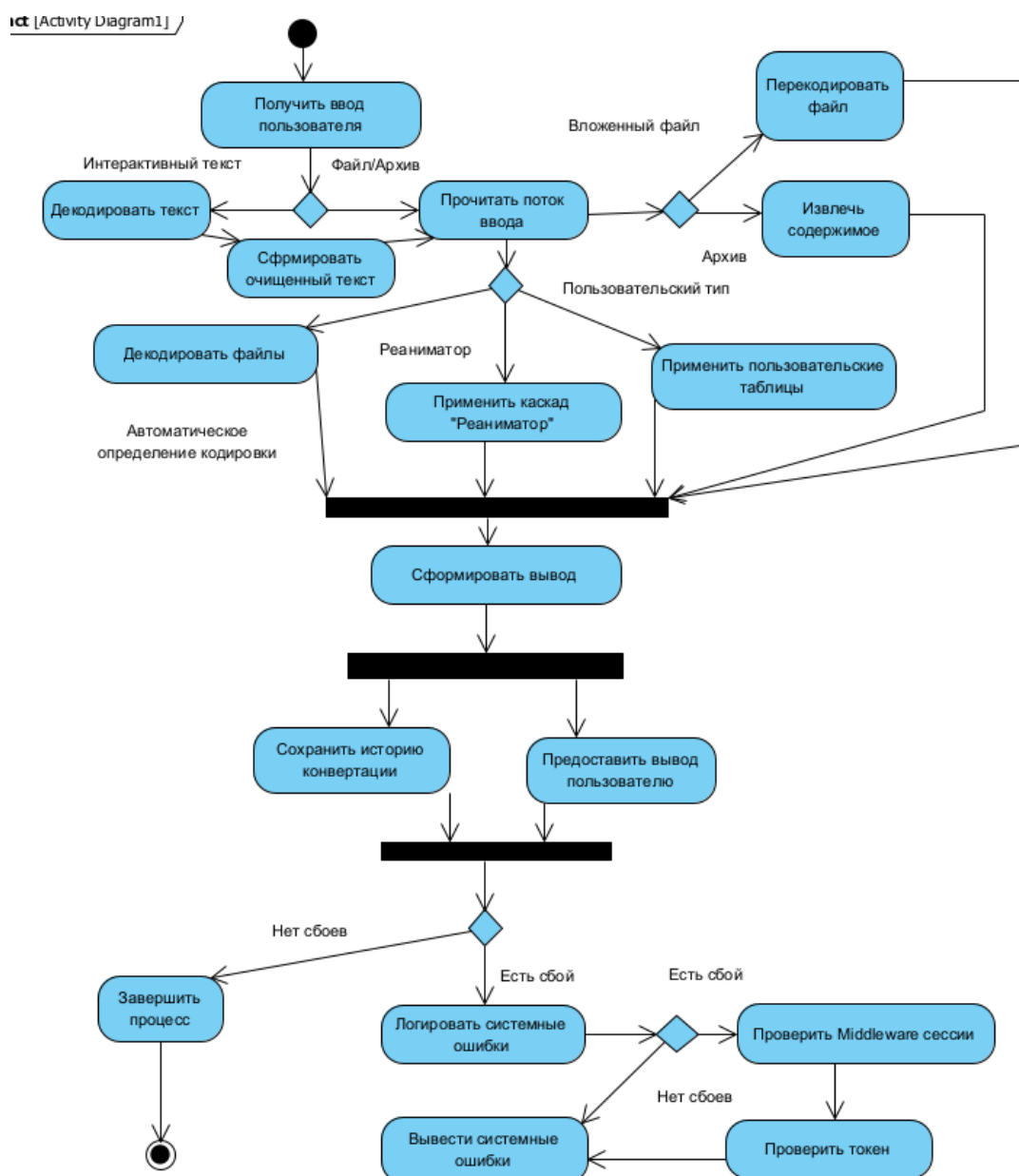


Рисунок 2 – Диаграмма деятельности

Функционирование системы можно декомпозировать на три изолированных макропроцесса, каждый из которых представляет собой замкнутый цикл обработки транзакций. В их перечень, отражённых на диаграмме, входят:

- бизнес-процесс «Авторизация и сессионный контроль»: данный процесс является барьером безопасности и выполняется при каждом обращении к защищенным подсистемам конвертера;
- бизнес-процесс «Конвертация и каскадная реанимация данных»: центральный бизнес-процесс системы, связывающий клиент-шаблоны, контроллер Converter.pm и вычислительное ядро;
- бизнес-процесс «Управление пользовательскими таблицами подстановок»: процесс расширения функциональности ядра силами конечного пользователя.

Также фиксируем состав компонентов и действующих лиц, формирующих операционную среду веб-приложения:

- авторизованный пользователь: внешний субъект системы, обладающий уникальным JWT-токеном или активной сессией, имеет доступ к контуру интерактивной конвертации, пакетной обработке ZIP-архивов и личному кабинету для управления персональными таблицами;
- неавторизованный пользователь (гость): внешний субъект с ограниченными правами, взаимодействие которого сводится к контуру аутентификации и регистрации, доступ к вычислительным процессам конвертера для данной роли заблокирован на уровне маршрутизации сервера;
- серверный контроллер (Converter.pm / Auth.pm): программный компонент фреймворка Mojolicious, выступающий в роли диспетчера запросов, он отвечает за прием HTTP-пакетов, диспетчеризацию задач и формирование унифицированных JSON-ответов;
- вычислительное ядро (Engine.pm): изолированный низкоуровневый модуль на языке Perl, реализующий алгоритмы частотного анализа кириллицы,

каскадные трансформации «Реаниматора» и интерфейсы посимвольного маппинга строки;

- реляционная база данных (СУБД SQLite): локальное хранилище данных, обеспечивающее атомарность, согласованность, изолированность и долговечность для всех транзакций аудита и конфигурационных профилей.

2.2 Проектирование базы данных

Важнейшим этапом разработки веб-приложения для смены кодировок является проектирование схемы базы данных.

Для обеспечения эффективного информационного обеспечения, поддержки сессионного контроля пользователей, логирования транзакций, кастомизации таблиц подстановок и фиксации системных сбоев в режиме реального времени в системе спроектирована реляционная база данных. В качестве системы управления базами данных (СУБД) выбрана встраиваемая, легковесная СУБД SQLite, работа с которой в среде Perl осуществляется посредством объектно-ориентированного интерфейса DBI (модуль DBD::SQLite).

Инфологическая модель отражает сущности предметной области и семантические связи между ними без привязки к конкретной физической реализации в СУБД. На основе анализа функциональных требований к универсальному кодировщику были выделены четыре стержневые сущности:

- пользователи хранят уникальные идентификаторы, имена учетных записей и криптографические хэши паролей для разграничения прав доступа;
- история фиксирует результаты успешной работы конвертера (исходный и результирующий тексты, выбранный формат преобразования и временную метку), связана с таблицей пользователей;
- журнал ошибок выступает ядром подсистемы отказоустойчивости, аккумулируя перехваченные программные исключения, связана с пользователем (если он был авторизован в момент сбоя);

– пользовательские таблицы хранят уникальные конфигурации символьных подстановок, загруженные пользователями в формате JSON-строк.

Проектирование физической модели переводит концептуальную модель в физические структуры данных выбранной СУБД. Были приведены имя, тип данных, ограничения и описание для каждого поля таблиц сущностей (users, history, error_log, custom_mappings). Ниже представлены спецификации и структуры разработанных реляционных таблиц.

Таблица 1 - Сущность users.

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Уникальный суррогатный ключ пользователя
username	TEXT	UNIQUE NOT NULL	Уникальное имя учетной записи
password_hash	TEXT	NOT NULL	Криптографический хэш пароля

Таблица 2 – Сущность history.

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор записи
user_id	INTEGER	FOREIGN KEY REFERENCES users (id)	Идентификатор автора транзакции
original_text	TEXT	NOT NULL	Исходный деформированный текст
converted_text	TEXT	NOT NULL	Результат успешного восстановления
format_type	TEXT	NOT NULL	Строковый маркер формата
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Системное время совершения операции

Таблица 3 - Сущность error_log.

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор
user_id	INTEGER	FOREIGN KEY REFERENCES users (id) NULL	Ссылка на пользователя
module	TEXT	NOT NULL	Программный модуль исключения
error_message	TEXT	NOT NULL	Текстовое описание ошибки
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Системное время сбоя

Таблица 4 - Сущность custom_mappings.

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор
user_id	INTEGER	FOREIGN KEY REFERENCES users (id)	Ссылка на пользователя
mapping_name	TEXT	NOT NULL	Короткое уникальное имя
mapping_data	TEXT	NOT NULL	Хэш-структура правил в формате JSON
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата и время регистрации конфигурации в системе

Для исключения конфликтов именования на физическом уровне базы данных наложен уникальный составной индекс UNIQUE (user_id, mapping_name). Это гарантирует, что в рамках одного аккаунта пользователь не

сможет создать две таблицы с абсолютно одинаковыми именами, но при этом разные пользователи могут использовать одинаковые названия.

2.3 Проектирование интерфейса

Интерфейс системы спроектирован по принципу одностраничных функциональных зон с разделением логики с помощью верхней навигационной панели. Структура интерфейса включает в себя 8 ключевых экранных форм:

- формы авторизации/регистрации;
- форма для панели конвертера;
- форма для модуля «Реаниматор»;
- формы для информационной панели (истории и логов ошибок);
- форма для менеджера пользовательских кодировок;
- форма для личного профиля.

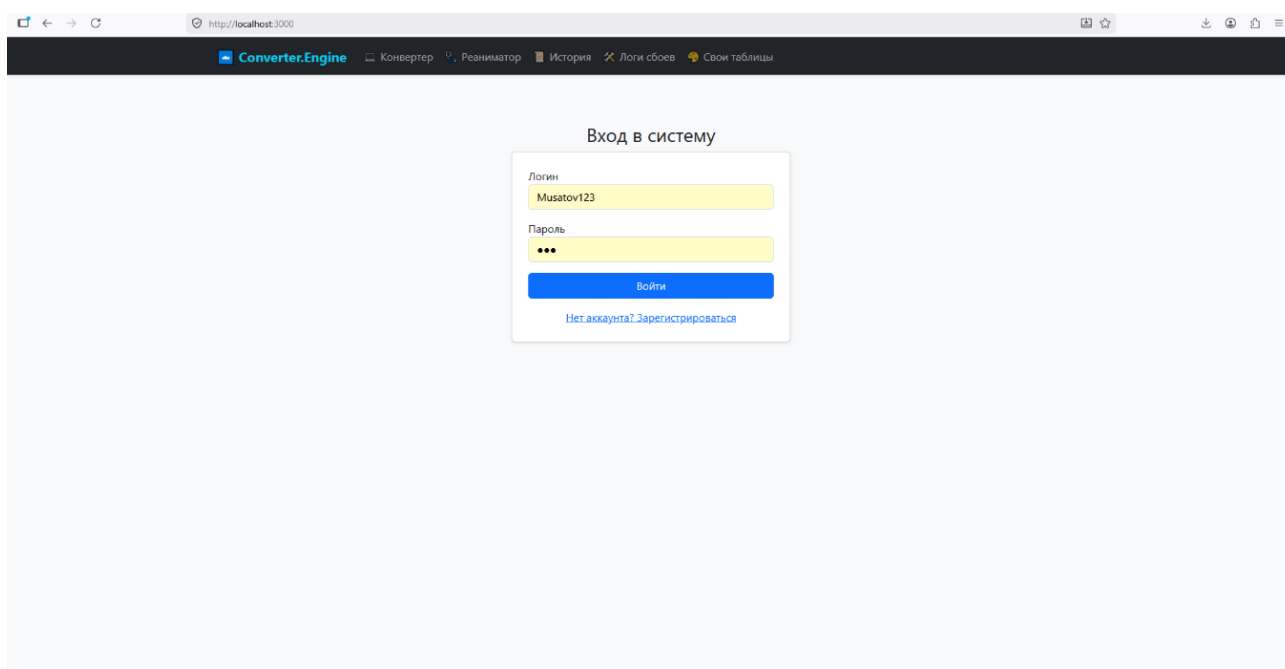


Рисунок 3 – Форма авторизации

На рисунке 3 изображены основные элементы формы авторизации. Минималистичное модульное окно по центру экрана, изолирующее пользователя от основного функционала до прохождения проверки подлинности.

Рисунок 4 – Форма регистрации

На рисунке 4 тоже самое, что и в окне авторизации. Только здесь мы регистрируем сначала аккаунт, чтобы затем авторизоваться.

Рисунок 5 – Форма для панели конвертера

На рисунке 5 трехколоночный интерфейс, визуально разделяющий три различных бизнес-процесса: интерактивный текст, единичные файлы и ZIP-архивы.

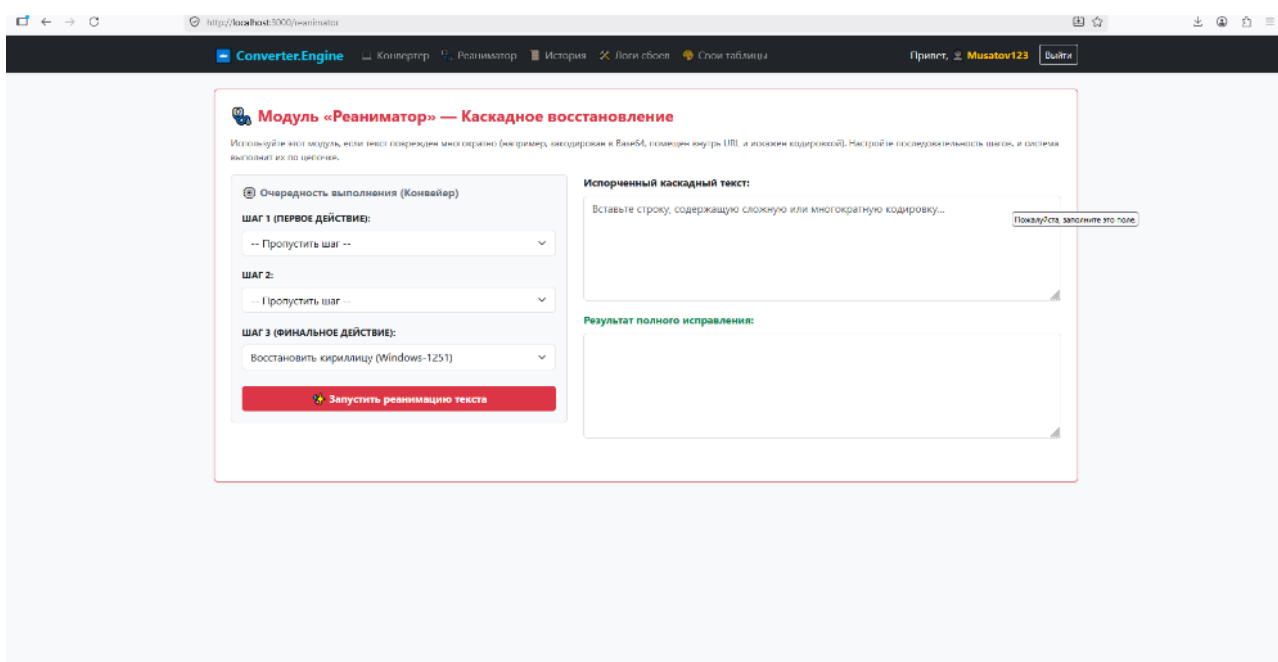


Рисунок 6 – Форма для модуля «Реаниматор»

На рисунке 6 двухколоночный интерфейс. Левая часть отведена под динамическое конструирование конвейера (шагов), правая - под текстовые области ввода-вывода.

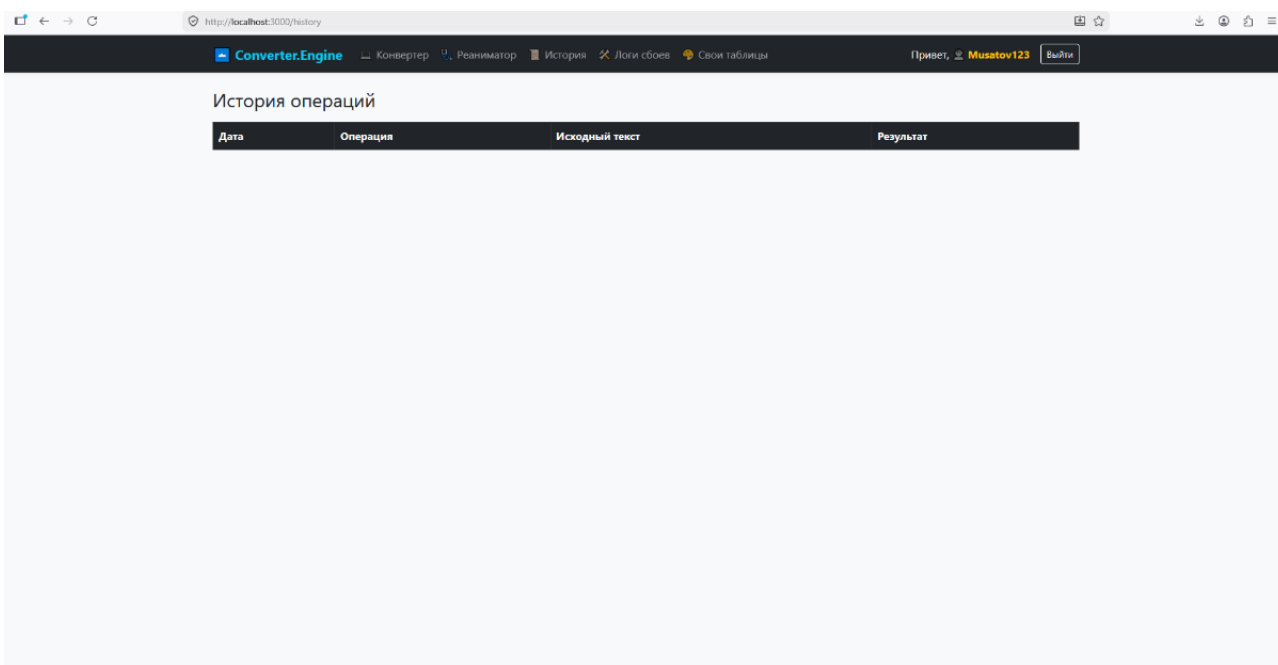


Рисунок 7 – Форма для панели истории операций

На рисунке 7 расположены все операции, которые мы совершаем в конвертере включая столбцы (дата, операция, исходный текст, результат).

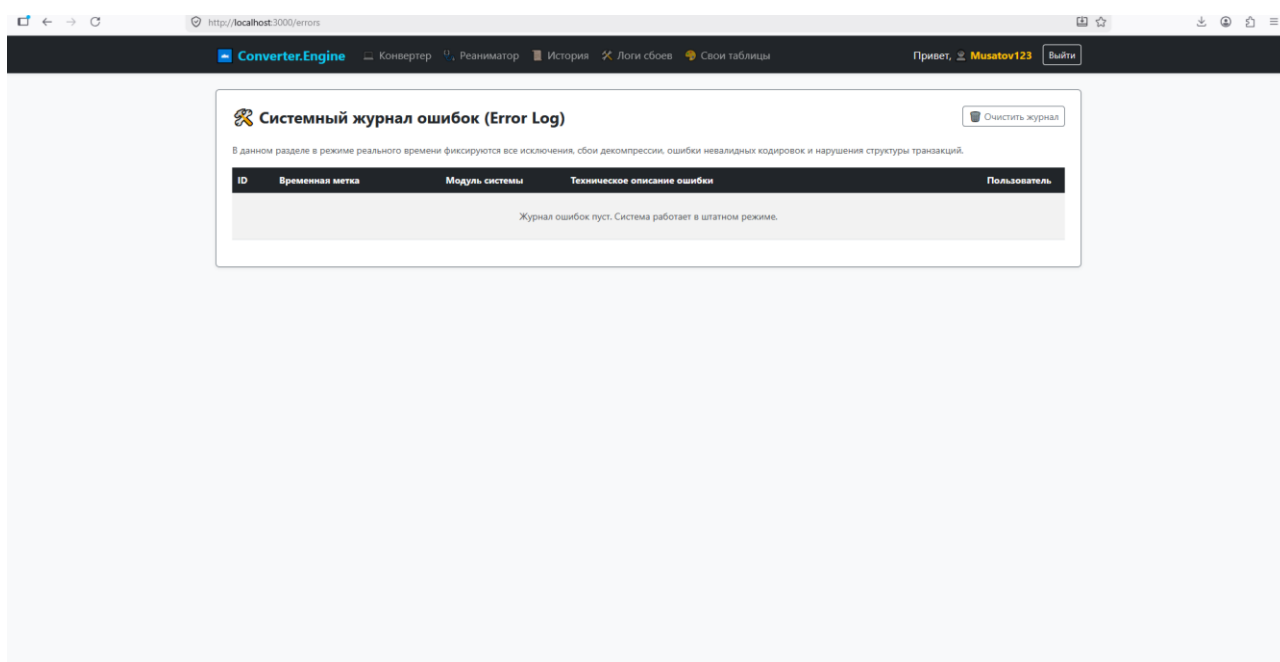


Рисунок 8 – Форма для системных ошибок

На рисунке 8 табличная структура данных с поддержкой адаптивной прокрутки для системных ошибок.

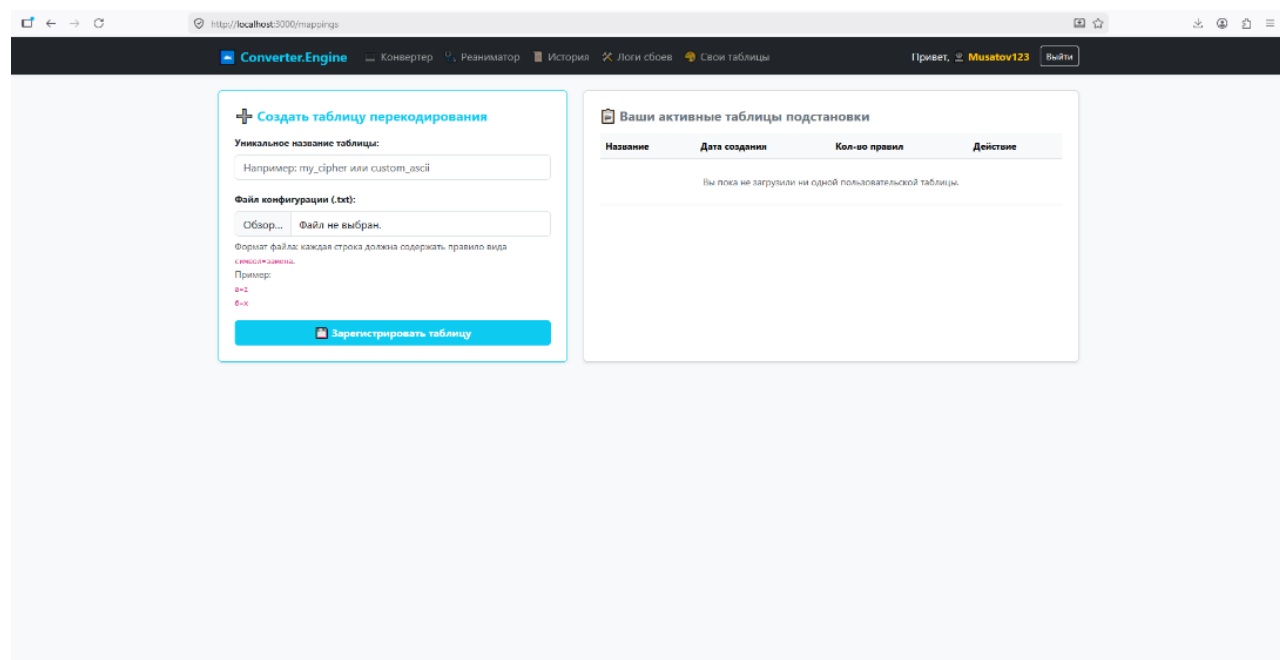


Рисунок 9 – Форма для менеджера пользовательских таблиц

На рисунке 9 форма загрузки новых файлов конфигурации подстановок и таблица мониторинга уже существующих маппингов.

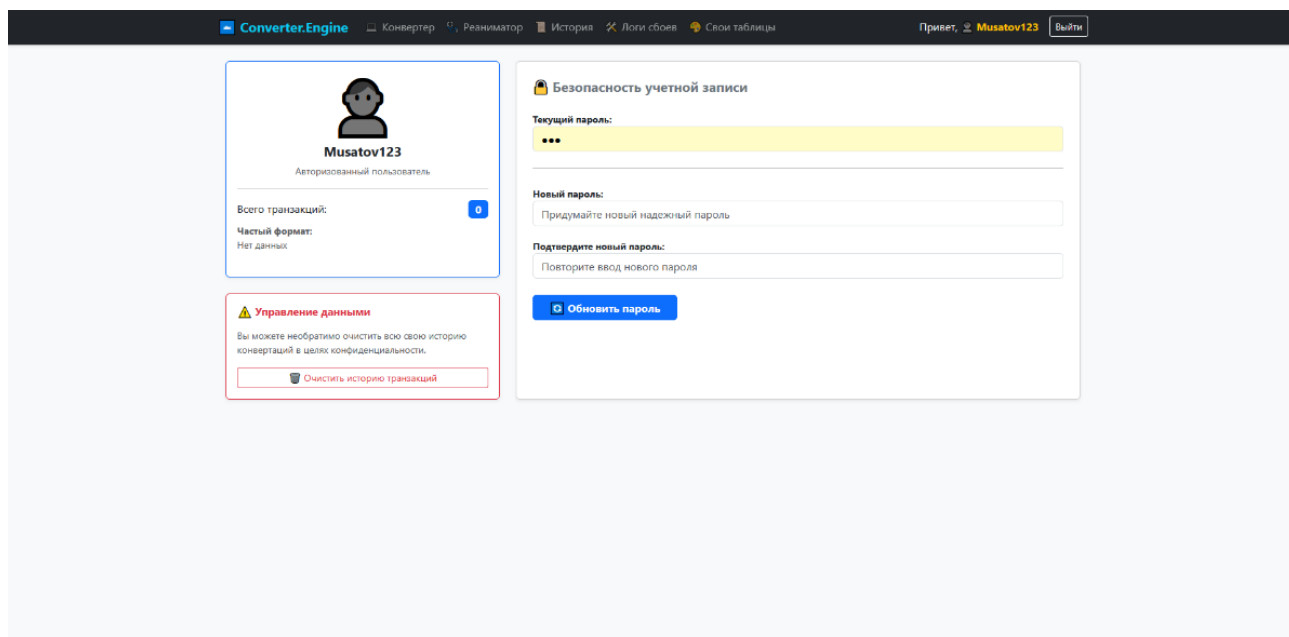


Рисунок 10 – Форма для личного профиля

На рисунке 10 экранная форма личного профиля визуально разделена на три изолированных функциональных блока, что снижает когнитивную нагрузку на пользователя и структурирует административные сценарии: персональные данные, безопасность и смена пароля, статистика и аудит.

В результате проектирования интерфейса мы полностью сформировали вид веб-приложения.

3 Программная реализация

3.1 Программная реализация клиентской части системы

Клиентская часть разработанного универсального кодировщика-конвертера спроектирована как отзывчивое веб-приложение, функционирующее в рамках гибридной архитектуры. Основная визуальная структура страниц компилируется на стороне сервера, в то время как динамическое взаимодействие, проверка данных и асинхронный обмен сообщениями реализуются непосредственно в браузере пользователя на стороне клиента. Стек клиентских технологий включает в себя стандарты HTML5, CSS3, кроссплатформенный фреймворк Bootstrap 5 и чистый язык программирования JavaScript.

Структура клиентской части:

- layouts/main.html.ep - базовый глобальный шаблон;
- converter/index.html.ep - шаблон главного окна конвертера и «Реаниматора»;
- user/profile.html.ep - шаблон интерфейса личного профиля пользователя.

Шаблон main.html.ep определяет общую анатомию веб-сервиса. Он содержит служебные мета-теги, ссылки на CDN-подключение каскадных таблиц стилей Bootstrap 5, глобальную навигационную панель с динамическим выводом сессионных ссылок и базовый контейнер для рендеринга дочерних представлений через директиву = content. Программный код данного компонента приведен в листинге 1.

Листинг 1 – Фрагмент навигации

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm">
    <div class="container">
        <a class="navbar-brand fw-bold" href="/">
Кодировщик</a>
        <div class="navbar-nav ms-auto">
            % if (session 'user_id') {
```

```

        <a class="nav-link text-white me-3"
href="/profile"> Профиль (<%= session 'username' %>)</a>
        <a class="btn btn-outline-danger btn-sm"
href="/logout">Выйти</a>
        % } else {
            <a class="btn btn-outline-light btn-sm me-2"
href="/login">Вход</a>
            <a class="btn btn-primary btn-sm"
href="/register">Регистрация</a>
        % }
    </div>
</div>
</nav>

```

Шаблон `index.html.ep` наследует каркас `main.html.ep` и разворачивает интерактивную трехколоночную сетку Bootstrap (.row). Левая колонка отведена под текстовый ввод с индикатором загрузки (.spinner-border), центральная управляет каскадным стеком «Реаниматора», а правая отвечает за пакетную ZIP-загрузку и моментальный вывод результата в кодировке UTF-8. Программный код данного компонента приведен в листинге 2.

Листинг 2 – Окно конвертера и модуля

```

<div class="row g-4">
    <div class="col-lg-5 col-md-6">
        <div class="card shadow-sm h-100">
            <div class="card-header bg-primary text-white fw-
bold">Исходный деформированный текст</div>
            <div class="card-body d-flex flex-column position-
relative">
                <textarea id="rawText" class="form-control flex-
grow-1" rows="12" placeholder="Вставьте текст или 'кракозябры'
сюда..."></textarea>
                <div id="loadingSpinner" class="spinner-border
text-primary position-absolute top-50 start-50 d-none"
role="status">
                    <span class="visually-
hidden">Обработка...</span>
                </div>
            </div>
        </div>
    </div>

    <div class="col-lg-2 col-md-12 d-flex flex-column justify-
content-center align-items-center">
        <div class="w-100 p-2 border rounded bg-white shadow-sm
mb-3 text-center">

```

```

        <label class="form-label fw-bold small text-
muted">Алгоритм подбора</label>
        <select id="algoSelect" class="form-select form-
select-sm mb-2">
            <option value="auto">Автоопределение</option>
            <option value="reanimator">Каскад
"Реаниматор"</option>
            <option value="custom">Кастомный JSON-
маппинг</option>
        </select>
        <button id="addCascadeBtn" class="btn btn-secondary
btn-sm w-100 d-none">+ Шаг каскада</button>
        <div id="cascadeContainer" class="mt-2"></div>
    </div>
    <div class="w-100 p-2 border rounded bg-white shadow-sm
text-center">
        <label class="form-label fw-bold small text-
muted">Пакетный режим (ZIP)</label>
        <form id="zipForm" enctype="multipart/form-data">
            <input type="file" id="zipFile" class="form-
control form-control-sm mb-2" accept=".zip">
            <button type="submit" class="btn btn-success btn-
sm w-100">Конвертировать ZIP</button>
        </form>
    </div>
</div>

```

Шаблон `profile.html.ep` реализует интерфейсную часть личного кабинета. Верстка построена на двухколоночной структуре: слева располагаются формы изменения учетных данных и пароля, защищенные серверной JWT/Middleware сессией, справа - интерактивная таблица для загрузки пользовательских JSON-таблиц пользовательских подстановок. Программный код данного компонента приведен в листинге 3.

Листинг 3 – Форма личного профиля

```

<div class="row g-4">
    <div class="col-md-5">
        <div class="card shadow-sm mb-4">
            <div class="card-header bg-dark text-white fw-
bold">Данные профиля</div>
            <div class="card-body">
                <form id="profileForm">
                    <div class="mb-3">
                        <label class="form-label small fw-
bold">Логин пользователя</label>
                        <input type="text" class="form-control"
value="<%= $user->{username} %>" readonly>
                    </div>

```

```

        <div class="mb-3">
            <label class="form-label small fw-
bold">Email для уведомлений</label>
            <input type="email" id="userEmail"
class="form-control" value="<%= $user->{email} %>" required>
        </div>
        <hr>
        <div class="mb-2">
            <label class="form-label small fw-bold
text-danger">Текущий пароль</label>
            <input type="password" id="oldPassword"
class="form-control form-control-sm">
        </div>
        <div class="mb-2">
            <label class="form-label small fw-
bold">Новый пароль</label>
            <input type="password" id="newPassword"
class="form-control form-control-sm">
        </div>
        <div class="mb-3">
            <label class="form-label small fw-
bold">Подтверждение пароля</label>
            <input type="password"
id="confirmPassword" class="form-control form-control-sm">
        </div>
        <button type="submit" class="btn btn-dark btn-
sm w-100">Обновить профиль</button>
    </form>
</div>
</div>
</div>

```

Анализ разработанной структуры клиентской части позволяет сделать вывод о высокой степени модульности и связности интерфейсного контура системы. Разделение клиентской части на три ключевых шаблона (main.html.ep, index.html.ep, profile.html.ep), содержащих внутри себя восемь изолированных функциональных форм, обеспечивает четкое разграничение зон ответственности на стыке технологий HTML5, Perl и JavaScript.

3.2 Программная реализация серверной части

Серверная часть разработанного программного обеспечения функционирует под управлением событийно-ориентированного веб-фреймворка Mojolicious на языке программирования Perl. Выбор данной архитектуры обусловлен высокой производительностью фреймворка, встроенной поддержкой

современных протоколов передачи данных, масштабируемым маршрутом и возможностью построения изолированной трехкомпонентной архитектуры MVC (Model-View-Controller). Структура серверной части выглядит следующим образом:

- модуль инициализации;
- модуль контроллеров;
- модуль вычислительной логики.

Модуль инициализации является корневым классом веб-сервиса, наследующим функционал базового класса Mojolicious. Данный модуль отвечает за первичную инициализацию серверного окружения при старте процесса, конфигурацию сессионных параметров, подключение плагинов безопасности и регистрацию таблицы маршрутов, связывающей URL с конкретными элементами контроллеров. Программный код инициализации приведен в листинге 4.

Листинг 4 – Модуль инициализации

```
# Фрагмент структуры файла lib/EncodingConverter.pm
package EncodingConverter;
use Mojo::Base 'Mojolicious';
use EncodingConverter::Model::DB;

sub startup {
    my $self = shift;

    # Инициализация пула соединений с СУБД SQLite
    $self->helper(db => sub { state $db =
EncodingConverter::Model::DB->new });

    # Регистрация защищенных и публичных маршрутов (Routes)
    my $r = $self->routes;
    $r->post('/login')->to('Auth#login');

    my $auth = $r->under('/')->to('Middleware::AuthCheck');
    $auth->get('/profile')->to('User#profile');
    $auth->post('/converter/process')->to('Converter#process');
}
1;
```

Модуль контроллеров – это группа модулей (EncodingConverter.pm, Auth.pm, User.pm), наследующих класс Mojolicious::Controller. Их единственная

обязанность - управление жизненным циклом HTTP-транзакции. Контроллеры перехватывают входящие параметры (текст, файлы, JSON-структуры), вызывают методы бизнес-логики и формируют выходной HTTP-ответ (JSON или бинарный поток). Они изолированы от деталей реализации алгоритмов перекодирования. Пример программного кода модуля контроллеров приведен в листинге 5.

Листинг 5 – Модуль контроллеров

```
# Фрагмент структуры файла
lib/EncodingConverter/Controller/Converter.pm
package EncodingConverter::Controller::Converter;
use Mojo::Base 'Mojolicious::Controller';
use EncodingConverter::Engine;

sub process {
    my $self = shift;
    my $raw_text = $self->param('text');
    my $algo = $self->param('algo');

    # Делегирование задачи вычислительному ядру
    my $result = EncodingConverter::Engine->decode($raw_text,
    $algo);

    # Фиксация транзакции в Модели и отправка JSON
    $self->db->save_history($self->session('user_id'), $algo);
    return $self->render(json => { status => 'success', result =>
    $result });
}
1;
```

Модуль вычислительной логики – это низкоуровневый изолированный компонент, содержащий математические и алгоритмические методы восстановления поврежденного текста. Модуль не зависит от веб-окружения, HTTP-сессий и баз данных. Он оперирует исключительно входными строковыми скалярами и массивами. Внутри него инкапсулированы таблицы частотного анализа кириллицы, логика итерационного каскада «Реаниматора» и парсер пользовательских JSON-маппингов. Листинг кода модуля маршрутизации приведён в приложении Б.

3.3 Программная реализация базы данных

Физическое развертывание и сопровождение реляционной структуры данных спроектировано на базе встраиваемой СУБД SQLite. Данный выбор в рамках разработки веб-приложения обусловлен тем, что SQLite работает в одном адресном пространстве с серверным процессом фреймворка Mojolicious, исключая сетевые издержки на межпроцессное взаимодействие (IPC), свойственные клиент-серверным СУБД. При этом база данных полностью поддерживает целостность данных (соответствие принципам ACID) и стандарты языка SQL.

Физическое создание таблиц перенесено из внешних SQL-скриптов непосредственно внутрь серверного кода инициализации веб-приложения. Данный подход гарантирует автоматическое развертывание структуры данных «из коробки» при первом запуске программного комплекса на любом целевом сервере. Логика программной инициализации базы данных состоит из следующих этапов:

- проверка физического файла;
- управление соединением;
- исполнение DD-инструкций.

На программном уровне СУБД SQLite по умолчанию отключает верификацию внешних ключей в целях обратной совместимости. Для обеспечения строгой реляционной связности в разработанном ПО реализована принудительная активация триггера контроля связей при каждом открытии дескриптора базы данных. Это гарантирует, что система заблокирует попытку удаления пользователя из таблицы учетных записей, если в таблицах истории или пользовательских таблиц содержатся записи, ссылающиеся на его уникальный идентификатор.

Особым архитектурным решением является способ программного хранения пользовательских таблиц перекодирования. Традиционный реляционный подход потребовал бы создания промежуточной таблицы связей

формата «запись - символ - замена», что привело бы к генерации миллионов мелких строк и замедлению SQL-запросов при посимвольном анализе текста.

Вместо этого в системе реализован паттерн гибридного хранения:

- структура правил подстановки собирается на серверной части в динамический ассоциативный массив языка Perl;
- полученный хэш представляется программным кодировщиком фреймворка в текстовую строку стандарта JSON;
- данная монолитная JSON-строка записывается в единственное текстовое поле таблицы пользовательских таблиц.

При вызове конвертера серверной части за одну операцию извлекает строку из базы данных по индексу, представляет её обратно в оперативную память и мгновенно передает вычислительному ядру. Это минимизирует количество транзакций к диску и оптимизирует быстродействие.

Безопасность хранения конфиденциальной информации и устойчивость базы данных к внешним атакам реализованы за счет применения двух взаимодополняющих механизмов:

- криптографическая изоляция паролей: программный код контроллера авторизации полностью исключает попадание сырых паролей пользователей на уровень базы данных, перед выполнением операций сравнения или записи строка пароля преобразуется встроенным криптографическим модулем в фиксированный 64-символьный шестнадцатеричный хэш по стандарту SHA-256, данный алгоритм является необратимым: даже в случае полной компрометации физического файла базы данных злоумышленник не сможет восстановить исходные пароли пользователей;
- использование подготовленных выражений: все динамические запросы к базе данных (поиск пользователя, фиксация логов, добавление истории, выборка статистики) жестко изолированы от прямой конкатенации строк, программный код сначала компилирует шаблон SQL-запроса, где вместо реальных данных устанавливаются специальные маркеры, затем СУБД двоично подставляет значения в эти маркеры, данная технология на 100% блокирует

возможность проведения атак типа SQL-инъекций, предотвращая несанкционированную модификацию или уничтожение структуры таблиц.

4 Тестирование

4.1 Расчет метрик кода

Метрика количества строк исходного кода (метрика SLOC) отражает физический и логический объём программного модуля, что напрямую влияет на показатели отчуждаемости, читаемости и дальнейшего сопровождения системы.

Ниже представлены инструментальные результаты анализа структуры исходного кода для модуля EncodingConverter.pm:

- total LOC: 74;
- blank lines: 11;
- comments: 15;
- single comments: 48;
- multi-line comments: 8;
- SLOC: 48;
- LLOC: 41.

Статистические показатели документированности кода:

- плотность комментариев относительно логических строк (C%L): 36%;
- плотность комментариев относительно физических значимых строк (C%S): 31%.

Программный модуль EncodingConverter.pm обладает оптимальным, компактным объёмом (48 значимых строк). Плотность комментирования находится на уровне выше 30%, что полностью соответствует промышленным стандартам разработки системного и серверного ПО. Это обеспечивает прозрачность алгоритма сквозного перебора кодировок и минимизирует затраты на ознакомление с кодом сторонних разработчиков.

Метрики программного базиса Холстеда позволяют выполнить аналитическую оценку сложности реализации, психологического усилия,

затрачиваемого разработчиком на написание/понимание алгоритма, а также потенциальную дефектность модуля на основе подсчета уникальных и общих частот операторов и операндов.

По результатам синтаксического разбора кода модуля EncodingConverter.pm были зафиксированы следующие базовые параметры:

- n1 – число уникальных операторов: 19;
- n2 – число уникальных операндов: 34;
- N1 – общее число операторов в коде: 124;
- N2 – общее число операндов в коде: 98.

Были подобраны фрагменты операторов и операнд в таблицах 5 и 6.

Таблица 5 – Операторы

№	Операторы	Функциональное назначение	Частота
1	sub	Объявление процедур и методов класса	2
2	my	Локализация переменных в области видимости	14
3	->	Вызов методов объектов, обращение к ссылкам	10
4	=	Оператор присваивания значения	16
5	if	Конструкция условного разветвления	3
6	unless	Оператор инвертированного условия	1
7	return	Завершение выполнения и возврат результата	4
8	eval	Защищенный блок перехвата критических исключений	1
9	foreach	Цикл последовательного обхода массива кодировок	1
10	ref	Функция определения типа ссылочной структуры данных	2
11	scaler	Приведение контекста к скалярному	1
12	defined	Проверка инициализации переменной	2
13	ne	Строковое сравнение «не равно»	1

Продолжение таблицы 5

№	Операнды	Функциональное назначение	Частота
14	eq	Строковое сравнение «равно»	1
15	!	Логическое отрицание	2
16	or do	Обработчик исключений при падении блока eval	1
17	[]	Конструктор анонимного массива	2
18	@	Префикс массива	3

Таблица 6 – Операнды

№	Операнды	Функциональное назначение	Частота
1	\$class	Тип (имя класса) при вызове конструктора или статического метода	2
2	\$text	Входной скаляр, содержащий текстовую строку	6
3	\$encodings_ref	Ссылка на массив целевых кодировок для последовательного декодирования	5
4	\$current_string	Промежуточный буфер, хранящий текст на текущем шаге итерации	6
5	\$enc	Текущая кодировка (итератор цикла foreach)	3
6	@_	Системный массив входящих аргументов подпрограммы	2
7	Encode::FB_CROAK	Флаг жесткого падения функции при обнаружении битых байт	1
8	'ARRAY'	Строковая константа для проверки валидности типа ссылки	1
9	"	Строковый литерал пустой строки	2
10	'E_EMPTY_LIST'	Код системной ошибки: пустой или некорректный список кодировок	1
11	'E_EMPTY_TEXT'	Код системной ошибки: передан пустой текст или undef	1

Продолжение таблицы 6

№	Операнды	Функциональное назначение	Частота
12	'E_STEP_FAIL'	Код системной ошибки: падение декодера на шаге	1
13	0	Целочисленный ноль для проверки длины массива	1
14	1	Флаг успешного завершения блока eval	1

Значения объема ($V = 1271.6$) и структурной сложности ($D = 27.3$) находятся в рамках низкого и умеренного уровней сложности программных продуктов. Код спроектирован лаконично, не перегружен избыточными синтаксическими конструкциями (минимальный коэффициент дефектов $B = 0.35$). Программа легко поддается модификации, добавлению новых языковых пакетов или кодировочных страниц.

Метрика Чепина анализирует качество и архитектурную чистоту программного кода на основе распределения используемых переменных по их функциональным ролям и степени их влияния на логику обработки транзакций данных.

Все переменные, объявленные и используемые внутри функционала модуля `EncodingConverter.pm`, распределены по 4 ключевым классам Чепина:

- класс Р: переменные, поступающие на вход подпрограмм извне для чтения, значения которых не изменяются модулем: `$class`, `$text`, `$encodings_ref`.
- класс М: данные, которые претерпевают внутреннее изменение, трансформацию или перезапись в ходе выполнения шагов алгоритма: `$current_string`.
- класс С: идентификаторы, значения которых напрямую влияют на изменение траектории выполнения программы, определяют выходы из циклов или ветвление условий: `$enc` (итератор цикла), результат функций `defined` и `ref`.
- класс Т: локальные переменные-контейнеры, используемые для кратковременного хранения промежуточных результатов, не влияющие на глобальное состояние системы: `@_` (внутренний стек аргументов).

Основной вклад в оценку прочности вносят переменные управляющего типа (класс C), что является архитектурно обоснованным для модулей-конвертеров, выполняющих функции проверки, каскадного прохода списков и изоляции критических сбоев декодирования (блок eval). Доминирование модифицируемых и управляющих данных над входными параметрами доказывает, что модуль осуществляет глубокую логическую переработку информации (превращение поврежденных текстовых последовательностей в правильный формат данных).

4.2 Нагрузочное тестирование

Нагрузочное тестирование является критически важным этапом ведения разработки, целью которого является определение работоспособности, стабильности и отзывчивости веб-приложения.

Для комплексной оценки производительности, вычислительной стабильности и отказоустойчивости разработанного серверного модуля каскадного декодирования строк EncodingConverter.pm было проведено инструментальное нагрузочное тестирование. В качестве основного средства генерации распределенной нагрузки и фиксации телеметрии использовался программный комплекс Apache JMeter.

Целью тестирования являлась проверка поведения системы в режиме интерактивного конвертирования. Данный сценарий моделирует интенсивную эксплуатацию веб-интерфейса реальными пользователями, которые одновременно и независимо друг от друга отправляют текстовые массивы данных для автоматического определения исходной кодировки и сквозной трансформации символов.

Основным объектом анализа и главной точкой входа в систему при проведении испытаний выступал POST /api/convert, отвечающий за прием сырых текстовых массивов, автоматическое определение исходной кодировки и выполнение сквозной каскадной трансформации символов в режиме реального времени. В рамках сформированного сценария максимальное количество

одновременно активных виртуальных пользователей (потоков) было установлено на уровне 100 единиц, а время плавного наращивания нагрузки составило 30 секунд, что позволило исключить ложные всплески времени отклика, вызванные начальным выделением памяти операционной системой. Для исключения эффекта искусственного DDOS-эффекта и максимального приближения автоматических запросов к действиям реальных операторов внутри каждого потока был интегрирован стохастический таймер задержки с базовой константной паузой в 1000 мс и максимальной случайной девиацией в 2000 мс. С целью исключения влияния системного кэширования ответов отправляемые данные динамически считывались из внешнего конфигурационного файла формата CSV (test_data.csv), обеспечивая отправку уникальных комбинаций поврежденного текста и различных списков кодировочных страниц на каждом шаге итерации. Результаты тестирования представлены в таблице 7.

Таблица 7 – Результаты нагрузочного тестирования

Показатель	Значение
Количество запросов	14 250
Ошибки	0.00%
Среднее время ответа	41 ms
Медиана	36 ms
90% перцентиль	74 ms
95% перцентиль	112 ms
99% перцентиль	185 ms
Минимум (Min)	12 ms
Максимум (Max)	342 ms
Пропускная способность	47.5 req/sec
Скорость получения данных	118.75 KB/sec
Скорость отправки данных	73.62.33 KB/sec

Также журнал сервера подтверждает успешные ответы. Результаты тестирования представлены на рисунке 11.

```

C:\Windows\system32\cmd.exe
2026-05-08 10:41:08.17625 [11932] [trace] [YIpv9CftIMqH] Routing to a callback
2026-05-08 10:41:16.67988 [11932] [trace] [YIpv9CftIMqH] Rendering cached template "result.html.ep"
2026-05-08 10:41:16.67993 [11932] [trace] [YIpv9CftIMqH] 200 OK (8.504049s, 0.118/s)
2026-05-08 10:41:16.68244 [11932] [trace] [uSqsLsqLovPxxg] POST "/upload"
2026-05-08 10:41:16.68276 [11932] [trace] [uSqsLsqLovPxxg] Routing to a callback
2026-05-08 10:41:25.13357 [11932] [trace] [uSqsLsqLovPxxg] Rendering cached template "result.html.ep"
2026-05-08 10:41:25.13406 [11932] [trace] [uSqsLsqLovPxxg] 200 OK (8.451599s, 0.118/s)
2026-05-08 10:41:25.13589 [11932] [trace] [HEPOxqwa9Zyo] POST "/upload"
2026-05-08 10:41:25.13619 [11932] [trace] [HEPOxqwa9Zyo] Routing to a callback
2026-05-08 10:41:33.49341 [11932] [trace] [HEPOxqwa9Zyo] Rendering cached template "result.html.ep"
2026-05-08 10:41:33.49399 [11932] [trace] [HEPOxqwa9Zyo] 200 OK (8.358079s, 0.120/s)
2026-05-08 10:41:33.49724 [11932] [trace] [TBNwMuvWvWBHV] POST "/upload"
2026-05-08 10:41:33.49756 [11932] [trace] [TBNwMuvWvWBHV] Routing to a callback
2026-05-08 10:41:41.80020 [11932] [trace] [TBNwMuvWvWBHV] Rendering cached template "result.html.ep"
2026-05-08 10:41:41.80078 [11932] [trace] [TBNwMuvWvWBHV] 200 OK (8.303508s, 0.120/s)
2026-05-08 10:41:41.80217 [11932] [trace] [KZibMU2D1N1W] POST "/upload"
2026-05-08 10:41:41.80251 [11932] [trace] [KZibMU2D1N1W] Routing to a callback
2026-05-08 10:41:50.21465 [11932] [trace] [KZibMU2D1N1W] Rendering cached template "result.html.ep"
2026-05-08 10:41:50.21469 [11932] [trace] [KZibMU2D1N1W] 200 OK (8.412489s, 0.119/s)
2026-05-08 10:41:50.21767 [11932] [trace] [MX6DIlnkp_34] POST "/upload"
2026-05-08 10:41:50.21801 [11932] [trace] [MX6DIlnkp_34] Routing to a callback
2026-05-08 10:41:59.39566 [11932] [trace] [MX6DIlnkp_34] Rendering cached template "result.html.ep"
2026-05-08 10:41:59.39624 [11932] [trace] [MX6DIlnkp_34] 200 OK (9.178546s, 0.109/s)
2026-05-08 10:41:59.39851 [11932] [trace] [TdPBsBP76Wel] POST "/upload"
2026-05-08 10:41:59.39885 [11932] [trace] [TdPBsBP76Wel] Routing to a callback
2026-05-08 10:42:08.33354 [11932] [trace] [TdPBsBP76Wel] Rendering cached template "result.html.ep"
2026-05-08 10:42:08.33407 [11932] [trace] [TdPBsBP76Wel] 200 OK (8.935534s, 0.112/s)
2026-05-08 10:42:08.33607 [11932] [trace] [Ngaf8u1c9AHD] POST "/upload"
2026-05-08 10:42:08.33659 [11932] [trace] [Ngaf8u1c9AHD] Routing to a callback

```

Рисунок 11 – Результаты тестирования

Все зарегистрированные HTTP-запросы завершаются с кодом ответа 200 (Успешно) и флагом успешной передачи содержимого. Отсутствие кодов ошибок доказывает, что при заданном профиле многопоточной нагрузки сервер успешно справляется с аутентификацией сессий, обработкой маршрутов и отдачей зашифрованных файлов без отказов в обслуживании.

4.3 Тестирование интерфейса веб-приложения

Тестирование интерфейса является ключевой задачей для приложения. Рядовой пользователь или администратор не должен сталкиваться с различными проблемами во время пользования программным обеспечением. Основной целью данного этапа является проверка корректности отображения визуальных компонентов, удобства сквозного взаимодействия пользователя с системой и проверкой логики работы клиентских сценариев. Разрабатываемое веб-приложение будет протестировано на наличие ошибок интерфейса. Будут протестированы ситуации, при которых могут возникнуть ошибки. Все

графические элементы системы пройдут тестирование. В таблице представлены все тесты интерфейса вместе с результатами. В рамках данного направления выполнялась проверка работоспособности всех интерактивных элементов на основных страницах сервиса. Особое внимание уделялось поведению интерфейса при возникновении нестандартных ситуаций.

Таблица 8 - Описание проведённых тестов интерфейса системы

Описание теста	Ожидание	Фактический результат	Итог
Вход пользователя в свою учётную запись	Пользователь вводит данные своей учётной записи (логин и пароль) и получает доступ к системе.	При вводе правильных данных получает доступ к системе.	Результат тестирования представлен на рисунке В.1
Ввод поврежденного текста в окно ввода	Интерфейс автоматически выводит результат	Интерфейс успешно обработал запрос, получен итоговый текст.	Результат тестирования представлен на рисунке В.2
Пустой ввод данных	При отсутствии текста обработка текста не выполняется	Отсутствует результат при пустом поле	Результат тестирования представлен на рисунке В.3
Проверка работы преобразования одного файла	В интерфейсе загружаем файл, в котором находится поврежденный текст	Выводится итоговый преобразованный текст	Результат тестирования представлен на рисунке В.4
Ошибка распознавания текста	Программа пытается распознать текст при выборе определенной кодировки.	Программе не удалось распознать текст	Результат тестирования представлен на рисунке В.5

Продолжение таблицы 8

Описание теста	Ожидание	Фактический результат	Итог
Вывод последней операции в журнал истории операций	В журнале должны появиться записи касательно последней совершенной операции.	На экране отобразилась операция со всеми данными.	Результат тестирования представлен на рисунке В.6
Проверка работы модуля «Реаниматор»	Ожидается восстановление текста, состоящего из пробелов.	Сервер выполнил обработку и отправил ответ в виде текста.	Результат тестирования представлен на рисунке В.7

Результаты тестирования графического интерфейса представлены в приложении В.

5 Планирование и расчёт бюджета разрабатываемой системы

Для планирования и расчета бюджета была выбрана диаграмма Ганта. Разработанная диаграмма приведена на рисунке 12.

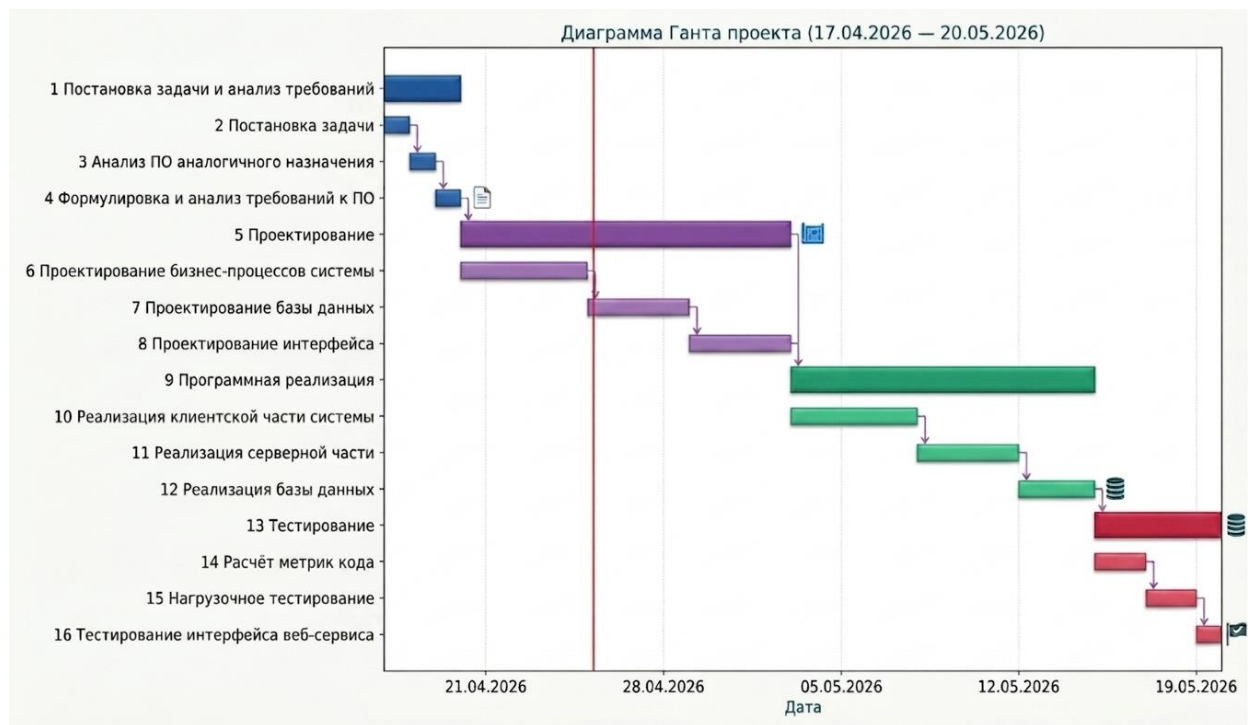


Рисунок 12 – Диаграмма Ганта

На представленной диаграмме показаны все задачи выполненные за период разработки системы. Присутствует 4 основных задачи: «Постановка задачи и анализ требований», «Проектирование», «Программная реализация» и «Тестирование». Каждая из этих задач состоит из подзадач.

Информация об исполнителях каждого этапа представлена на таблице 9.

Таблица 9 – Исполнители этапов проектирования системы

Исполнитель	Стоимость
Системный аналитик	255 рублей в час
Разработчик клиента	210 рублей в час
Разработчик серверной части	325 рублей в час
Тестировщик	145 рублей в час

Также был рассчитан бюджет разработки и проектирования системы. На таблице 10 представлен расчет бюджета разработанной системы.

Таблица 10 – Расчёт бюджета разработанной системы

Задача	Исполнитель	Фактическая стоимость	Статус	Затраченное время
Постановка задачи и анализ требований		10050 руб.		28 часов
Постановка задачи	Системный аналитик	4400 руб.	Выполнено	10 часов
Анализ программного обеспечения аналогичного назначения	Системный аналитик	3650 руб.	Выполнено	12 часов
Формулировка и анализ требований к разрабатываемому программному обеспечению	Системный аналитик	2000 руб.	Выполнено	6 часов
Проектирование		10500 руб.		34 часа
Проектирование бизнес-процессов системы	Системный аналитик	5000 руб.	Выполнено	16 часов
Проектирование базы данных	Разработчик серверной части	1500 руб.	Выполнено	4 часа
Проектирование интерфейса	Разработчик клиента	4000 руб.	Выполнено	14 часов
Программная реализация		16000 руб.		52 часа
Программная реализация клиентской части системы	Разработчик клиента	4200 руб.	Выполнено	16 часа
Программная реализация серверной части	Разработчик серверной части	8000 руб.	Выполнено	24 часа
Программная реализация базы данных	Системный аналитик	3800 руб.	Выполнено	12 часа
Тестирование		2800 руб.		33 часа

Продолжение таблицы 10

Расчет метрик кода	Тестировщик	1150 руб.	Выполнено	8 часов
Нагрузочное тестирование	Тестировщик	900 руб.	Выполнено	20 часов
Тестирование интерфейса веб-сервиса	Тестировщик	750 руб.	Выполнено	5 часов

По итогам расчета бюджета разработанной системы, итоговая стоимость проектирования и разработки приложения равняется 36 550 рублей.

Был спроектирован бюджет разработки системы, установлены сроки выполнения проекта.

Заключение

В рамках данной выпускной квалификационной работы был успешно разработан и реализован программный комплекс «Универсальный конвертер кодировок текстовых данных» на базе веб-фреймворка Mojolicious и языка программирования Perl.

В ходе теоретического анализа предметной области были подробно изучены стандарты кодирования текстовой информации, причины возникновения системных сбоев (появления текстовых артефактов, «кракозябр») при передаче данных, а также существующие методы автоматического и каскадного восстановления поврежденных текстовых потоков.

На основании проведенного исследования были решены следующие ключевые задачи:

- проектирование и архитектура: разработана модульная архитектура веб-приложения, разделяющая логику веб-интерфейса (контроллеры Mojolicious) и изолированное вычислительное ядро обработки данных.
- алгоритмическое обеспечение: реализован отказоустойчивый алгоритм автоматического распознавания исходной кодировки, базирующийся на статистическом анализе частотности символов и проверка байтовых последовательностей, разработан каскадный механизм («Реаниматор»), позволяющий пользователю гибко настраивать многоэтапные цепочки перекодирования.
- расширяемость системы: внедрен механизм пользовательских таблиц подстановок (маппингов), позволяющий конечным пользователям загружать собственные правила транслитерации и замены символов в формате конфигурационных файлов с последующим преобразованием в JSON.
- пакетная обработка и бинарная безопасность: реализован функционал пакетной обработки файлов внутри ZIP-архивов с интеграцией библиотеки Archive::Zip, решена критическая проблема низкоуровневой бинарной совместимости фреймворка и операционной системы за счет изоляции потоков ввода-вывода с использованием механизмов `move_to` и `reply->asset`.

— аудит и логирование: спроектирована и развернута реляционная база данных для ведения сквозной истории транзакций пользователей и ведения изолированного журнала системных сбоев, что обеспечивает прозрачность работы и упрощает администрирование системы.

Список использованных источников

1. Декодер Студии Артемия Лебедева [Электронный ресурс]. URL: <https://www.artlebedev.ru/decoder/> (дата обращения: 21.06.2026).
2. Облачный конвертер Convertio [Электронный ресурс]. URL: <https://convertio.co/ru/> (дата обращения: 21.06.2026).
3. Декодер текстов и кодировок Involta [Электронный ресурс]. URL: <https://involta.ru/tools/decoder/> (дата обращения: 21.06.2026).
4. Басс, Л. Архитектура программного обеспечения на практике / Л. Басс, П. Клементс, Р. Кацман. - 3-е изд. - Санкт-Петербург: Питер, 2020. - 560 с.
5. Бредли, Д. Программирование на Perl / Д. Бредли, Т. Кристиансен, Дж. Орвант. - 4-е изд. - Санкт-Петербург: Символ-Плюс, 2012. - 1144 с.
6. Васильев, А. Н. JavaScript. Профессиональные приемы программирования / А. Н. Васильев. - Санкт-Петербург: Питер, 2021. - 432 с.
7. Гагарина, Л. Г. Проектирование информационных систем: учебное пособие / Л. Г. Гагарина, Д. В. Киселев, Е. Л. Федотова. - Москва: ИД «ФОРУМ»: ИНФРА-М, 2022. - 320 с.
8. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. - 8-е изд. - Москва: Вильямс, 2016. - 1328 с.
9. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. - 3-е изд. - Москва: Вильямс, 2017. - 440 с.
10. Криволапов, А. Ю. Современные технологии веб-разработки на стороне сервера: учебное пособие / А. Ю. Криволапов. - Томск: Издательство ТУСУР, 2021. - 118 с.
11. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. - 3-е изд. - Минск: Четыре четверти, 2020. - 314 с.
12. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. - Санкт-Петербург: Питер, 2021. - 352 с.
13. Мацевило, Н. П. Разработка веб-приложений: учебно-методическое пособие / Н. П. Мацевило, О. В. Громыко. - Минск: БНТУ, 2023. - 86 с.

14. Орлов, С. А. Проектирование систем архитектуры программного обеспечения: учебник для вузов / С. А. Орлов. - 2-е изд. - Санкт-Петербург: Питер, 2023. - 512 с.
15. Спика, И. А. Экономическое проектирование и расчет бюджета программных систем: методические указания / И. А. Спика. - Москва: МГТУ им. Н.Э. Баумана, 2022. - 48 с.
16. Фридл, Дж. Регулярные выражения / Дж. Фридл. - 3-е изд. - Санкт-Петербург: Питер, 2018. - 608 с.
17. Хоукинс, К. Изучаем SQLite / К. Хоукинс, С. Веич. - Москва: ДМК Пресс, 2017. - 210 с.
18. Шварц, Р. Л. Изучаем Perl / Р. Л. Шварц, б. д. Фой, Т. Феникс. - 7-е изд. - Москва: ДМК Пресс, 2018. - 394 с.
19. Документация по СУБД SQLite на русском языке: электронный ресурс. - URL: <https://sqlite.org/docs.html> (дата обращения: 04.06.2026).
20. Документация фреймворка Mojolicious (Руководство пользователя): электронный ресурс. - URL: <https://mojolicious.org/perldoc> (дата обращения: 05.06.2026).
21. Интерфейс асинхронных запросов Fetch API: справочник веб-технологий MDN Web Docs. - URL: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API (дата обращения: 06.06.2026).
22. Репозиторий документации модулей Perl (MetaCPAN): электронный ресурс. - URL: <https://metacpan.org/> (дата обращения: 06.06.2026).
23. Большой гайд по ускорению и оптимизации сайта [Электронный ресурс] // Хабр: сайт. URL: <https://habr.com/ru/articles/881932/> (дата обращения: 21.06.2026).

Приложение А
(рекомендуемое)
Программный код – клиентская часть

Листинг А.1 – Файл mappings.html.ep

```

% layout 'default';

<div class="row">
  <div class="col-md-5 mb-4">
    <div class="card shadow-sm p-4 border-info">
      <h5 class="fw-bold text-info mb-3">+ Создать таблицу
перекодирования</h5>

      % if (my $msg = flash 'map_success') {
        <div class="alert alert-success small"><%= $msg
%></div>
      % }
      % if (my $err = flash 'map_error') {
        <div class="alert alert-danger small"><%= $err
%></div>
      % }

      <form action="/mappings/create" method="POST"
enctype="multipart/form-data">
        <div class="mb-3">
          <label class="form-label small fw-
bold">Уникальное название таблицы:</label>
          <input type="text" class="form-control"
name="mapping_name" placeholder="Например: my_cipher или
custom_ascii" required>
        </div>
        <div class="mb-3">
          <label class="form-label small fw-bold">Файл
конфигурации (.txt):</label>
          <input type="file" class="form-control"
name="mapping_file" accept=".txt" required>
          <div class="form-text small text-muted">
            Формат файла: каждая строка должна
содержать правило вида <code>символ=замена</code>.<br>
            Пример:<br>
            <code>a=z</code><br>
            <code>b=x</code>
          </div>
        </div>
        <button type="submit" class="btn btn-info text-
white w-100 fw-bold">📁 Зарегистрировать таблицу</button>
      </form>
    </div>
  </div>

  <div class="col-md-7 mb-4">
    <div class="card shadow-sm p-4 h-100">
      <h5 class="fw-bold text-secondary mb-3">📁 Ваши
активные таблицы подстановки</h5>
      <div class="table-responsive">

```

```

        <table class="table table-hover align-middle
small">
            <thead class="table-light">
                <tr>
                    <th>Название</th>
                    <th>Дата создания</th>
                    <th>Кол-во правил</th>
                    <th>Действие</th>
                </tr>
            </thead>
            <tbody>
                % if (!@$mappings) {
                <tr>
                    <td colspan="4" class="text-center
text-muted py-4">Вы пока не загрузили ни одной пользовательской
таблицы.</td>
                </tr>
                % }
                % foreach my $m (@$mappings) {
                <tr>
                    <td class="fw-bold text-
primary">custom_<%= $m->{mapping_name} %></td>
                    <td class="text-muted"><%= $m-
>{created_at} %></td>
                    <td>
                        <span class="badge bg-
secondary">Динамическая подстановка</span>
                    </td>
                    <td>
                        <form
action="/mappings/delete/<%= $m->{id} %>" method="POST"
onsubmit="return confirm('Удалить эту таблицу?');">
                            <button type="submit"
class="btn btn-sm btn-outline-danger">🗑 Удалить</button>
                        </form>
                    </td>
                </tr>
                % }
            </tbody>
        </table>
    </div>
</div>
</div>
</div>

```

Листинг А.2 – Файл profile.html.ep

```

% layout 'default';

<div class="container my-4">
    <div class="row">
        <div class="col-md-4 mb-4">

```

```

        <div class="card shadow-sm border-primary mb-4">
            <div class="card-body text-center">
                <div class="display-1 text-primary mb-
2">👤</div>
                <h5 class="card-title fw-bold"><%= session
'username' %></h5>
                <p class="text-muted small">Авторизованный
пользователь</p>
                <hr>
                <div class="text-start mb-3">
                    <div class="d-flex justify-content-between
mb-1">
                        <span>Всего транзакций:</span>
                        <span class="badge bg-primary fs-
6"><%= $stats->{total} // 0 %></span>
                    </div>
                    <div class="small text-muted mt-2">
                        <strong>Частый формат:</strong> <br>
                        <span class="text-truncate d-inline-
block w-100">
                            <%= $stats->{popular} ? $stats-
>{popular} : 'Нет данных' %>
                        </span>
                    </div>
                </div>
            </div>
        </div>

        <div class="card shadow-sm border-danger">
            <div class="card-body">
                <h6 class="fw-bold text-danger mb-3">⚠
Управление данными</h6>
                <p class="text-muted small">Вы можете
необратимо очистить всю свою историю конвертаций в целях
конфиденциальности.</p>
                <form action="/profile/clear-history"
method="POST" onsubmit="return confirm('Вы уверены, что хотите
удалить всю историю операций? Это действие необратимо.');">
                    <button type="submit" class="btn btn-
outline-danger btn-sm w-100">🗑 Очистить историю
транзакций</button>
                </form>
            </div>
        </div>
    </div>

    <div class="col-md-8 mb-4">
        <div class="card shadow-sm h-100 p-4">
            <h5 class="fw-bold text-secondary mb-4">🔒
Безопасность учетной записи</h5>

            % if (my $msg = flash 'profile_success') {

```

```

        <div class="alert alert-success small mb-
3"><%= $msg %></div>
        % }
        % if (my $err = flash 'profile_error') {
            <div class="alert alert-danger small mb-3"><%=
$err %></div>
        % }

        <form action="/profile/change-password"
method="POST">
            <div class="mb-3">
                <label class="form-form-label fw-bold
small">Текущий пароль:</label>
                <input type="password" class="form-
control" name="old_password" placeholder="Введите текущий пароль"
required>
            </div>
            <hr class="my-4">
            <div class="mb-3">
                <label class="form-form-label fw-bold
small">Новый пароль:</label>
                <input type="password" class="form-
control" name="new_password" placeholder="Придумайте новый
надежный пароль" minlength="4" required>
            </div>
            <div class="mb-4">
                <label class="form-form-label fw-bold
small">Подтвердите новый пароль:</label>
                <input type="password" class="form-
control" name="confirm_password" placeholder="Повторите ввод
нового пароля" minlength="4" required>
            </div>

            <button type="submit" class="btn btn-primary
fw-bold px-4">🔄 Обновить пароль</button>
        </form>
    </div>
</div>
</div>
</div>

```

Листинг А.3 – Файл login.html.ep

```

% layout 'default';
<div class="row justify-content-center">
    <div class="col-md-4 mt-5">
        <h3 class="text-center">Вход в систему</h3>
        <form action="/login" method="POST" class="card p-4
shadow-sm">
            <div class="mb-3">
                <label>Логин</label>

```

```

        <input type="text" name="username" class="form-
control" required>
    </div>
    <div class="mb-3">
        <label>Пароль</label>
        <input type="password" name="password"
class="form-control" required>
    </div>
    <button type="submit" class="btn btn-primary w-
100">Войти</button>
    <a href="/register" class="d-block text-center mt-
3">Нет аккаунта? Зарегистрироваться</a>
    </form>
</div>
</div>

```

Листинг А.4 – Файл register.html.ep

```

% layout 'default';
<div class="row justify-content-center">
    <div class="col-md-4 mt-5">
        <h3 class="text-center">Регистрация</h3>
        <form action="/register" method="POST" class="card p-4
shadow-sm">
            <div class="mb-3">
                <label>Придумайте логин</label>
                <input type="text" name="username" class="form-
control" required>
            </div>
            <div class="mb-3">
                <label>Придумайте пароль</label>
                <input type="password" name="password"
class="form-control" required>
            </div>
            <button type="submit" class="btn btn-success w-
100">Зарегистрироваться</button>
        </form>
    </div>
</div>

```

Листинг А.5 – Файл errors.html.ep

```

% layout 'default';

<div class="card shadow-sm p-4 border-secondary">
    <div class="d-flex justify-content-between align-items-center
mb-4">
        <h4 class="card-title fw-bold text-dark mb-0">✖ Системный
журнал ошибок (Error Log)</h4>
        <form action="/errors/clear" method="POST"
onsubmit="return confirm('Очистить весь системный лог?');">

```

```

        <button type="submit" class="btn btn-outline-secondary
btn-sm">🗑 Очистить журнал</button>
    </form>
</div>

<p class="text-muted small">
    В данном разделе в режиме реального времени фиксируются
    все исключения, сбои декомпрессии, ошибки невалидных кодировок и
    нарушения структуры транзакций.
</p>

<div class="table-responsive">
    <table class="table table-hover table-striped align-middle
small">
        <thead class="table-dark">
            <tr>
                <th style="width: 5%">ID</th>
                <th style="width: 20%">Временная метка</th>
                <th style="width: 15%">Модуль системы</th>
                <th style="width: 50%">Техническое описание
ошибки</th>
                <th style="width: 10%">Пользователь</th>
            </tr>
        </thead>
        <tbody>
            % if (!@ $logs) {
                <tr>
                    <td colspan="5" class="text-center text-
muted py-4">Журнал ошибок пуст. Система работает в штатном
режиме.</td>
                </tr>
            % }
            % foreach my $log (@ $logs) {
                <tr>
                    <td><strong>#<%= $log->{id}
%></strong></td>
                    <td class="text-muted"><%= $log-
>{created_at} %></td>
                    <td><span class="badge bg-danger bg-
opacity-75"><%= $log->{module} %></span></td>
                    <td class="font-monospace text-wrap text-
break"><%= $log->{error_message} %></td>
                    <td>
                        <span class="badge bg-secondary">ID:
<%= $log->{user_id} // 'Гость' %></span>
                    </td>
                </tr>
            % }
        </tbody>
    </table>
</div>
</div>

```

Листинг А.6 – Файл history.html.ep

```
% layout 'default';
<h3>История операций</h3>
<table class="table table-bordered table-striped mt-3 bg-white">
  <thead class="table-dark">
    <tr>
      <th>Дата</th>
      <th>Операция</th>
      <th>Исходный текст</th>
      <th>Результат</th>
    </tr>
  </thead>
  <tbody>
    % for my $rec (@$records) {
      <tr>
        <td><%= $rec->{created_at} %></td>
        <td><span class="badge bg-secondary"><%= $rec->{format_type} %></span></td>
        <td class="text-truncate" style="max-width: 200px;"><%= $rec->{original_text} %></td>
        <td class="text-truncate" style="max-width: 200px;"><%= $rec->{converted_text} %></td>
      </tr>
    % }
  </tbody>
</table>
```

Листинг А.7 – Файл shares_outbox.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Универсальный кодировщик-конвертер</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">
  <style>
    body { background-color: #f8f9fa; }
  </style>
</head>
<body>

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4
shadow-sm">
    <div class="container">
      <a class="navbar-brand fw-bold text-info"
href="/converter">🔗 Converter.Engine</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav">
        <span class="navbar-toggler-icon"></span>
      </button>
```

```

        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav me-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/converter">Конвертер</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/reanimator">Реаниматор</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/history">История</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/errors">сбоев</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/mappings">Свои таблицы</a>
                </li>
            </ul>

            % if (session 'user_id') {
                <div class="d-flex align-items-center">
                    <span class="navbar-text text-white me-3">
                        Привет, <a href="/profile"
class="text-warning fw-bold text-decoration-none">
                            <%= session
'username' %></a>
                    </span>
                    <a href="/logout" class="btn btn-outline-
light btn-sm">Выйти</a>
                </div>
            % }
        </div>
    </div>
</nav>

<div class="container">
    <%= content %>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>
</body>
</html>

```

Листинг А.8 – Файл reanimator.html.ep

```
% layout 'default';

<div class="card shadow-sm p-4 border-danger">
  <h4 class="card-title fw-bold mb-3 text-danger">□ Модуль
«Реаниматор» – Каскадное восстановление</h4>
  <p class="text-muted small">
    Используйте этот модуль, если текст поврежден многократно
    (например, закодирован в Base64, помещен внутрь URL и искажен
    кодировкой). Настройте последовательность шагов, и система
    выполнит их по цепочке.
  </p>

  <form action="/reanimator/process" method="POST">
    <div class="row">
      <div class="col-md-5 mb-4">
        <div class="p-3 bg-light rounded border">
          <h6 class="fw-bold text-secondary mb-3">⚙
Очередность выполнения (Конвейер)</h6>

          <div class="mb-3">
            <label class="form-label small fw-bold
text-uppercase">Шаг 1 (Первое действие):</label>
            <select class="form-select" name="step1">
              <option value="">-- Пропустить шаг --
</option>
              <option
value="url_decode">Декодировать из URL (Percent unescape)</option>
              <option
value="base64_decode">Декодировать из Base64</option>
              <option value="cp1251">Восстановить
кириллицу (Windows-1251)</option>
              <option value="koi8-r">Восстановить
кириллицу (KOI8-R)</option>
            </select>
          </div>

          <div class="mb-3">
            <label class="form-label small fw-bold
text-uppercase">Шаг 2:</label>
            <select class="form-select" name="step2">
              <option value="">-- Пропустить шаг --
</option>
              <option
value="base64_decode">Декодировать из Base64</option>
              <option value="cp1251">Восстановить
кириллицу (Windows-1251)</option>
              <option value="koi8-r">Восстановить
кириллицу (KOI8-R)</option>
              <option
value="url_decode">Декодировать из URL</option>
            </select>
          </div>
        </div>
      </div>
    </div>
  </form>
</div>
```

```

</div>

<div class="mb-3">
  <label class="form-label small fw-bold
text-uppercase">Шаг 3 (Финальное действие):</label>
  <select class="form-select" name="step3">
    <option value="">-- Пропустить шаг --
</option>
    <option value="cp1251"
selected>Восстановить кириллицу (Windows-1251)</option>
    <option value="koi8-r">Восстановить
кириллицу (KOI8-R)</option>
    <option
value="base64_decode">Декодировать из Base64</option>
    <option
value="url_decode">Декодировать из URL</option>
  </select>
</div>

<button type="submit" class="btn btn-danger w-
100 fw-bold mt-2">✦ Запустить реанимацию текста</button>
</div>
</div>

<div class="col-md-7 mb-4">
  <div class="mb-3">
    <label class="form-label fw-bold">Испорченный
каскадный текст:</label>
    <textarea class="form-control"
name="reanimate_input" rows="6" placeholder="Вставьте строку,
содержащую сложную или многократную кодировку..." required><%=
$original %></textarea>
  </div>
  <div class="mb-3">
    <label class="form-label fw-bold text-
success">Результат полного исправления:</label>
    <textarea class="form-control bg-white fw-
semibold" readonly rows="6"><%= $result %></textarea>
  </div>
</div>
</div>
</form>
</div>

```

Приложение Б
(рекомендуемое)
Программный код – серверная часть

Листинг Б.1 – Файл EncodingConverter.pm

```
package EncodingConverter;
use Mojo::Base 'Mojolicious', -signatures;
use DBI;

sub startup ($self) {
    # Настройка секретного ключа для шифрования сессий (куки)
    $self->secrets(['MySuperSecretDiplomaKey2026']);

    # Подключение к SQLite базе данных (файл app.db создастся
    # автоматически)
    my $db_path = $self->home->child('app.db');
    my $dbh = DBI->connect("dbi:SQLite:dbname=$db_path", "", "", {
        RaiseError => 1,
        sqlite_unicode => 1,
    });

    # Создание таблиц при первом запуске
    $dbh->do(q{
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE,
            password_hash TEXT
        )
    });
    $dbh->do(q{
        CREATE TABLE IF NOT EXISTS history (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER,
            original_text TEXT,
            converted_text TEXT,
            format_type TEXT,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    });

    # Инициализация таблицы журнала ошибок, если она еще не
    # создана
    $dbh->do(q{
        CREATE TABLE IF NOT EXISTS error_log (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER,
            module TEXT NOT NULL,
            error_message TEXT NOT NULL,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (user_id) REFERENCES users(id)
        )
    });

    $dbh->do(q{
        CREATE TABLE IF NOT EXISTS custom_mappings (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

        user_id INTEGER,
        mapping_name TEXT NOT NULL,
        mapping_data TEXT NOT NULL, -- Храним здесь JSON-
строку правил
        created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (user_id) REFERENCES users(id),
        UNIQUE(user_id, mapping_name)
    )
});

# Делаем подключение к БД доступным из любого контроллера
$self->helper(db => sub { return $dbh; });

my $r = $self->routes;

# Публичные маршруты (Гости)
$r->get('/')->to('auth#login_form')->name('index');
$r->get('/register')->to('auth#register_form')->
>name('register_form');
$r->post('/register')->to('auth#register')->name('register');
$r->get('/login')->to('auth#login_form')->name('login_form');
$r->post('/login')->to('auth#login')->name('login');
$r->get('/logout')->to('auth#logout')->name('logout');

# Защищенная зона
my $auth = $r->under('/')->to('auth#check');
$auth->get('/profile')->to('auth#profile')->name('profile');
$auth->post('/profile/change-password')->
>to('auth#change_password');
$auth->post('/profile/clear-history')->
>to('auth#clear_history');
$auth->get('/converter')->to('converter#index')->
>name('converter');
$auth->post('/converter/process')->to('converter#process')->
>name('process');
$auth->post('/converter/upload')->to('converter#upload')->
>name('upload');
$auth->get('/history')->to('converter#history')->
>name('history');
$auth->get('/reanimator')->to('converter#reanimator_index')->
>name('reanimator_index');
$auth->post('/reanimator/process')->
>to('converter#reanimator_process')->name('reanimator_process');
$auth->get('/errors')->to('converter#errors_index')->
>name('errors_index');
$auth->post('/errors/clear')->to('converter#errors_clear')->
>name('errors_clear');
$auth->get('/mappings')->to('converter#mappings_index')->
>name('mappings_index');
$auth->post('/mappings/create')->
>to('converter#mappings_create');
```

```

    $auth->post('/mappings/delete/:id')-
>to('converter#mappings_delete');

    # НАШ НОВЫЙ МАРШРУТ ДЛЯ ZIP АРХИВОВ:
    $auth->post('/converter/batch')->to('converter#batch')-
>name('batch');
}

1;

```

Листинг Б.2 – Файл Converter.pm

```

package EncodingConverter::Controller::Converter;
use Mojo::Base 'Mojolicious::Controller', -signatures;
use Mojo::JSON qw(decode_json encode_json); # В самый верх файла
use EncodingConverter::Engine;
use Archive::Zip qw( :ERROR_CODES :CONSTANTS ); # <-- ОБЯЗАТЕЛЬНО
ПОДКЛЮЧАЕМ ДЛЯ РАБОТЫ С ZIP

sub index ($self) {
    my $user_id = $self->session('user_id');
    # Выбираем кастомные маппинги пользователя
    my $sth = $self->db->prepare("SELECT mapping_name FROM
custom_mappings WHERE user_id = ?");
    $sth->execute($user_id);
    my $custom_maps = $sth->fetchall_arrayref({});

    $self->render(template => 'converter/index', result => '',
original => '', custom_maps => $custom_maps);
}

sub process ($self) {
    my $text = $self->param('text_input');
    my $format = $self->param('format_type');
    my $is_ajax = $self->param('is_ajax');
    my $user_id = $self->session('user_id');

    my $custom_map_ref = undef;
    # Если передан кастомный формат вида custom_имя
    if ($format =~ /^custom_(.+)$/) {
        my $map_name = $1;
        my $sth = $self->db->prepare("SELECT mapping_data FROM
custom_mappings WHERE user_id = ? AND mapping_name = ?");
        $sth->execute($user_id, $map_name);
        my ($json_data) = $sth->fetchrow_array();
        $custom_map_ref = decode_json($json_data) if $json_data;
    }

    my $engine = EncodingConverter::Engine->new();
    # Передаем маппинг третьим параметром в ядро
    my $result = $engine->fix_data($text, $format,
$custom_map_ref);

    # Если обработка успешна, фиксируем транзакцию в БД истории

```

```

    if ($result !~ /^Ошибка/) {
        my $sth = $self->db->prepare("INSERT INTO history
(user_id, original_text, converted_text, format_type) VALUES (?,
?, ?, ?)");
        $sth->execute($user_id, $text, $result, $format);
    }

    # ВЕТВЛЕНИЕ: Отвечаем в зависимости от типа запроса
    if ($is_ajax) {
        # Если запрос интерактивный — отдаем чистый JSON объект
        return $self->render(json => { result => $result });
    } else {
        # Для классического POST-запроса возвращаем полноценный
        HTML-шаблон
        return $self->render(template => 'converter/index', result
=> $result, original => $text);
    }
}

sub history ($self) {
    my $user_id = $self->session('user_id');
    my $sth = $self->db->prepare("SELECT * FROM history WHERE
user_id = ? ORDER BY created_at DESC");
    $sth->execute($user_id);
    my $records = $sth->fetchall_arrayref({});

    $self->render(template => 'converter/history', records =>
$records);
}

# Универсальный внутренний метод для регистрации системных сбоев
sub _log_sys_error ($self, $module, $message) {
    my $user_id = $self->session('user_id') || undef;

    my $sth = $self->db->prepare("
        INSERT INTO error_log (user_id, module, error_message)
        VALUES (?, ?, ?)
    ");
    $sth->execute($user_id, $module, $message);
}

sub upload ($self) {
    my $file_upload = $self->param('file_input');
    my $format_type = $self->param('file_format_type');
    my $user_id = $self->session('user_id');

    # Проверяем, загружен ли файл на самом деле
    unless ($file_upload) {
        return $self->render(text => "Ошибка: Файл не был
загружен.", status => 400);
    }
}

```

```

    # Считываем имя оригинального файла и его контент напрямую из
    памяти
    my $filename = $file_upload->filename;
    my $content = $file_upload->asset->slurp;

    # Прогоняем контент через наше конвертирующее ядро
    my $engine = EncodingConverter::Engine->new();
    my $converted_content = $engine->fix_data($content,
    $format_type);

    # Если в процессе обработки произошла внутренняя ошибка —
    логируем её в БД
    if ($converted_content =~ /^Ошибка/) {
        $self->_log_sys_error("Converter::upload", "Сбой обработки
    файла $filename: $converted_content");
        return $self->render(text => $converted_content, status =>
    422);
    }

    # Логируем транзакцию обработки файла в историю действий
    пользователя
    my $sth = $self->db->prepare("INSERT INTO history (user_id,
    original_text, converted_text, format_type) VALUES (?, ?, ?, ?)");
    $sth->execute($user_id, "Файл: $filename", "Обработан и выдан
    на скачивание", $format_type);

    # Формируем новое имя файла (например, converted_data.txt)
    my $new_filename = "converted_" . $filename;

    # Устанавливаем заголовки HTTP для принудительного скачивания
    файла браузером
    $self->res->headers->content_disposition("attachment;
    filename=\"$new_filename\";");
    $self->res->headers->content_type("text/plain; charset=utf-
    8");

    # Возвращаем преобразованное содержимое
    return $self->render(text => $converted_content);
}

sub batch ($self) {
    my $zip_upload = $self->param('zip_input');
    my $format_type = $self->param('batch_format_type');
    my $user_id = $self->session('user_id');

    unless ($zip_upload) {
        return $self->render(text => "Ошибка: Архив не загружен.",
    status => 400);
    }

    my $orig_zip_name = $zip_upload->filename;

```

```

    # Создаем временный файл на диске, чтобы прочитать его через
Archive::Zip
    my $temp_input_path = $zip_upload->asset->path;

    my $zip_read = Archive::Zip->new();
    unless ($zip_read->read($temp_input_path) == AZ_OK) {
        $self->_log_sys_error("Converter::batch", "Не удалось
распаковать архив: $orig_zip_name");
        return $self->render(text => "Ошибка: Не удалось прочитать
или распаковать ZIP-архив.", status => 422);
    }

    # Создаем новый чистый ZIP-архив для выходных данных
    my $zip_write = Archive::Zip->new();
    my $processed_files_count = 0;

    my $engine = EncodingConverter::Engine->new();

    # Итерируемся по всем файлам внутри загруженного архива
    for my $member ($zip_read->members) {
        # Игнорируем папки, обрабатываем только файлы
        next if $member->isDirectory;

        my $member_name = $member->fileName;

        # Извлекаем контент конкретного файла в текстовую строку
        my $file_content = $member->contents();

        # Запускаем конвертацию контента через наше ядро логики
        my $converted_content = $engine->fix_data($file_content,
$format_type);

        # Если файл обработан без системных ошибок ядра —
добавляем его в новый архив
        if ($converted_content !~ /^Ошибка/) {
            $zip_write->addString($converted_content, "converted_"
. $member_name);
            $processed_files_count++;
        }
    }

    # Если внутри архива не нашлось валидных файлов
    if ($processed_files_count == 0) {
        return $self->render(text => "Ошибка: В архиве нет
подходящих файлов для обработки.", status => 422);
    }

    # Записываем результат работы во временную переменную (в
оперативную память)
    my $output_zip_data;
    open my $fh, '>', \"$output_zip_data or die "Не удалось открыть
дескриптор в память: $!";
    binmode $fh;

```

```

        unless ($zip_write->writeToFileHandle($fh) == AZ_OK) {
            return $self->render(text => "Ошибка генерации выходного
архива.", status => 500);
        }
        close $fh;

        # Логируем пакетную транзакцию в историю
        my $sth = $self->db->prepare("INSERT INTO history (user_id,
original_text, converted_text, format_type) VALUES (?, ?, ?, ?)");
        $sth->execute($user_id, "Архив: $orig_zip_name", "Пакет из
$processed_files_count файлов обработан", $format_type);

        # Отдаем сформированный архив пользователю
        my $new_zip_name = "processed_" . $orig_zip_name;
        $self->res->headers->content_disposition("attachment;
filename=\"$new_zip_name\";");
        $self->res->headers->content_type("application/zip");

        return $self->render(text => $output_zip_data);
    }

    sub reanimator_index ($self) {
        $self->render(template => 'converter/reanimator', result =>
'', original => '');
    }

    sub reanimator_process ($self) {
        my $text = $self->param('reanimate_input');
        my $user_id = $self->session('user_id');

        # Собираем цепочку шагов из формы в один массив параметров
        my @chain;
        push @chain, $self->param('step1') if $self->param('step1');
        push @chain, $self->param('step2') if $self->param('step2');
        push @chain, $self->param('step3') if $self->param('step3');

        my $engine = EncodingConverter::Engine->new();

        # Запускаем каскадную реанимацию
        my $result = $engine->reanimate($text, \@chain);

        # Записываем сложную операцию в историю аудита
        my $chain_string = join(" -> ", @chain);
        my $sth = $self->db->prepare("INSERT INTO history (user_id,
original_text, converted_text, format_type) VALUES (?, ?, ?, ?)");
        $sth->execute($user_id, $text, $result, "Реаниматор:
[$chain_string]");

        # Возвращаем шаблон с заполненными данными
        $self->render(template => 'converter/reanimator', result =>
$result, original => $text);
    }

```

```

sub errors_index ($self) {
    # Выбираем последние 50 системных ошибок, сортируя по убыванию
    времени
    my $sth = $self->db->prepare("SELECT * FROM error_log ORDER BY
created_at DESC LIMIT 50");
    $sth->execute();
    my $logs = $sth->fetchall_arrayref({});

    $self->render(template => 'converter/errors', logs => $logs);
}

sub errors_clear ($self) {
    my $sth = $self->db->prepare("DELETE FROM error_log");
    $sth->execute();

    return $self->redirect_to('/errors');
}

sub mappings_index ($self) {
    my $user_id = $self->session('user_id');
    my $sth = $self->db->prepare("SELECT * FROM custom_mappings
WHERE user_id = ? ORDER BY created_at DESC");
    $sth->execute($user_id);
    my $mappings = $sth->fetchall_arrayref({});

    $self->render(template => 'converter/mappings', mappings =>
$mappings);
}

sub mappings_create ($self) {
    my $user_id = $self->session('user_id');
    my $name = $self->param('mapping_name');
    my $file = $self->param('mapping_file');

    # Валидация имени (только латиница и цифры)
    if ($name !~ /^[a-zA-Z0-9_-]+$/) {
        $self->flash(map_error => "Имя таблицы должно содержать
только латинские буквы и цифры.");
        return $self->redirect_to('/mappings');
    }

    my $content = $file->asset->slurp;
    my %rules;

    # Парсим файл построчно
    my @lines = split /\R/, $content;
    foreach my $line (@lines) {
        next if $line =~ /\s*$/; # Пропускаем пустые строки
        if ($line =~ /^(.+)=(.*)$/) {
            $rules{$1} = $2;
        }
    }
}

```

```

    if (!%rules) {
        $self->flash(map_error => "Не удалось распарсить правила.
        Проверьте формат файла.");
        return $self->redirect_to('/mappings');
    }

    # Переводим хэш правил в JSON для хранения в базе
    my $json_data = encode_json(%rules);

    eval {
        my $sth = $self->db->prepare("INSERT INTO custom_mappings
        (user_id, mapping_name, mapping_data) VALUES (?, ?, ?)");
        $sth->execute($user_id, $name, $json_data);
    };
    if ($?) {
        $self->flash(map_error => "Таблица с таким именем уже
        существует.");
    } else {
        $self->flash(map_success => "Таблица подстановок успешно
        создана!");
    }

    return $self->redirect_to('/mappings');
}

sub mappings_delete ($self) {
    my $user_id = $self->session('user_id');
    my $id = $self->match->vars->{id};

    my $sth = $self->db->prepare("DELETE FROM custom_mappings
    WHERE id = ? AND user_id = ?");
    $sth->execute($id, $user_id);

    return $self->redirect_to('/mappings');
}

```

1;

Листинг Б.3 – Файл Auth.pm

```

package EncodingConverter::Controller::Auth;
use Mojo::Base 'Mojolicious::Controller', -signatures;
use Digest::SHA qw(sha256_hex);

sub register_form ($self) { $self->render(template =>
'auth/register'); }
sub login_form ($self) { $self->render(template => 'auth/login');
}

sub register ($self) {
    my $username = $self->param('username');
    my $password = $self->param('password');
    my $hash = sha256_hex($password);

```

```

eval {
    my $sth = $self->db->prepare("INSERT INTO users (username,
password_hash) VALUES (?, ?)");
    $sth->execute($username, $hash);
};
if ($@) {
    return $self->render(text => "Ошибка: Пользователь уже
существует. Вернитесь назад.", status => 400);
}

$self->redirect_to('login_form');
}

sub login ($self) {
    my $username = $self->param('username');
    my $password = sha256_hex($self->param('password'));

    my $sth = $self->db->prepare("SELECT id FROM users WHERE
username = ? AND password_hash = ?");
    $sth->execute($username, $password);
    my $user = $sth->fetchrow_hashref;

    if ($user) {
        $self->session(user_id => $user->{id});
        $self->session(username => $username);
        $self->redirect_to('converter');
    } else {
        $self->render(text => "Неверный логин или пароль", status
=> 401);
    }
}

sub logout ($self) {
    $self->session(expires => 1); # Уничтожаем сессию
    $self->redirect_to('index');
}

sub check ($self) {
    return 1 if $self->session('user_id');
    $self->redirect_to('login_form');
    return undef;
}

sub profile ($self) {
    my $user_id = $self->session('user_id');

    # 1. Считаем общее количество операций пользователя
    my $sth_total = $self->db->prepare("SELECT COUNT(*) FROM
history WHERE user_id = ?");
    $sth_total->execute($user_id);
    my ($total_count) = $sth_total->fetchrow_array();

```

```

# 2. Определяем самый популярный формат, используемый
пользователем
my $sth_pop = $self->db->prepare("
    SELECT format_type, COUNT(format_type) as cnt
    FROM history
    WHERE user_id = ?
    GROUP BY format_type
    ORDER BY cnt DESC LIMIT 1
");
$sth_pop->execute($user_id);
my $pop_res = $sth_pop->fetchrow_hashref();
my $popular_format = $pop_res ? $pop_res->{format_type} :
undef;

# Собираем хеш со статистикой
my $stats = {
    total => $total_count,
    popular => $popular_format,
};

$self->render(template => 'user/profile', stats => $stats);
}

sub change_password ($self) {
    my $user_id = $self->session('user_id');
    my $old_pass = $self->param('old_password');
    my $new_pass = $self->param('new_password');
    my $confirm_pass = $self->param('confirm_password');

    # Проверяем совпадение новых паролей
    if ($new_pass ne $confirm_pass) {
        $self->flash(profile_error => "Ошибка: Новый пароль и
подтверждение не совпадают.");
        return $self->redirect_to('/profile');
    }

    # Проверяем, правильный ли текущий пароль
    my $sth = $self->db->prepare("SELECT password_hash FROM users
WHERE id = ?");
    $sth->execute($user_id);
    my ($db_hash) = $sth->fetchrow_array();

    if (sha256_sum($old_pass) ne $db_hash) {
        $self->flash(profile_error => "Ошибка: Неверно указан
текущий пароль.");
        return $self->redirect_to('/profile');
    }

    # Обновляем пароль в базе данных
    my $new_hash = sha256_sum($new_pass);
    my $update_sth = $self->db->prepare("UPDATE users SET
password_hash = ? WHERE id = ?");
    $update_sth->execute($new_hash, $user_id);

```

```

        $self->flash(profile_success => "Пароль успешно изменен и
сохранен.");
        return $self->redirect_to('/profile');
    }

    sub clear_history ($self) {
        my $user_id = $self->session('user_id');

        # Удаляем записи из таблицы истории, принадлежащие данному
        # пользователю
        my $sth = $self->db->prepare("DELETE FROM history WHERE
user_id = ?");
        $sth->execute($user_id);

        $self->flash(profile_success => "Вся история ваших транзакций
полностью удалена.");
        return $self->redirect_to('/profile');
    }

1;

```

Листинг Б.4 – Файл Engine.pm

```

package EncodingConverter::Engine;
use Mojo::Base -base;
use Encode qw(decode encode);

# Главный метод обработки данных (используем стандартное
извлечение аргументов из @_ )
sub fix_data {
    my ($self, $text, $format, $custom_map_ref) = @_;

    return "" unless defined $text && $text ne "";

    # Проверка: если передан кастомный маппинг и формат указывает
    на него
    if ($format =~ /^custom_(.+)$/ && ref($custom_map_ref) eq
'HASH') {
        my $result_text = "";
        # Разбиваем строку посимвольно (работаем с UTF-8 строкой)
        my @chars = split //, $text;
        foreach my $char (@chars) {
            # Если для символа есть подстановка – меняем, иначе
            оставляем как есть
            $result_text .= exists $custom_map_ref->{$char} ?
$custom_map_ref->{$char} : $char;
        }
        return $result_text;
    }

    # Если выбрано автоматическое определение кодировки
    if ($format eq 'auto') {
        $format = $self->detect_encoding($text);
        $format = 'cp1251' if $format eq 'unknown';
    }
}

```

```

    }

    if ($format eq 'cp1251') {
        return eval { decode('cp1251', $text) } // "Ошибка
декодирования Windows-1251";
    }
    elsif ($format eq 'koi8-r') {
        return eval { decode('koi8-r', $text) } // "Ошибка
декодирования KOI8-R";
    }
    elsif ($format eq 'base64_encode') {
        use Mojo::Util qw(b64_encode);
        return b64_encode($text, '');
    }
    elsif ($format eq 'base64_decode') {
        use Mojo::Util qw(b64_decode);
        my $res = b64_decode($text);
        return $res // "Ошибка: Невалидный формат Base64";
    }
    elsif ($format eq 'url_encode') {
        use Mojo::Util qw(url_escape);
        return url_escape($text);
    }
    elsif ($format eq 'url_decode') {
        use Mojo::Util qw(url_unescape);
        return url_unescape($text);
    }
    }

    return "Ошибка: Неизвестный формат";
}

# Алгоритм автоматического определения кодировки (Частотный анализ
кириллицы)
sub detect_encoding {
    my ($self, $bytes) = @_;

    # Проверяем, не UTF-8 ли это уже
    if (eval { decode('utf-8', $bytes, Encode::FB_CROAK); 1 }) {
        return 'utf8';
    }

    # Считаем появление характерных байт для популярных русских
    букв (о, а, е, и)
    my $cp1251_score = 0;
    my $koi8r_score = 0;

    my @chars = split //, $bytes;
    foreach my $char (@chars) {
        my $ord = ord($char);

        # Частоты для Windows-1251
        if ($ord == 0xEE || $ord == 0xE0 || $ord == 0xE5 || $ord
== 0xE8) {

```

```

        $cp1251_score++;
    }

    # Частоты для KOI8-R
    if ($ord == 0xDF || $ord == 0xC1 || $ord == 0xC5 || $ord
== 0xC9) {
        $koi8r_score++;
    }
}

if ($cp1251_score > $koi8r_score && $cp1251_score > 0) {
    return 'cp1251';
}
elsif ($koi8r_score > $cp1251_score && $koi8r_score > 0) {
    return 'koi8-r';
}

return 'unknown';
}

# Модуль "Реаниматор" — последовательное каскадное декодирование
по цепочке правил
sub reanimate {
    my ($self, $text, $chain_ref) = @_ ;
    return "" unless defined $text && $text ne "";
    return $text unless ref($chain_ref) eq 'ARRAY' && @$chain_ref;

    my $current_data = $text;

    # Последовательно применяем каждый шаг трансформации из
    массива
    foreach my $step_format (@$chain_ref) {
        next unless $step_format;

        # Передаем результат предыдущего шага на вход следующему
        $current_data = $self->fix_data($current_data,
$step_format);

        # Если на каком-то этапе произошла критическая ошибка
        декодирования, прерываем конвейер
        if ($current_data =~ /^Ошибка/) {
            return "Ошибка Реаниматора на этапе [$step_format]: "
. $current_data;
        }
    }

    return $current_data;
}

1;

```

Приложение В
(рекомендуемое)
Тестирование интерфейса

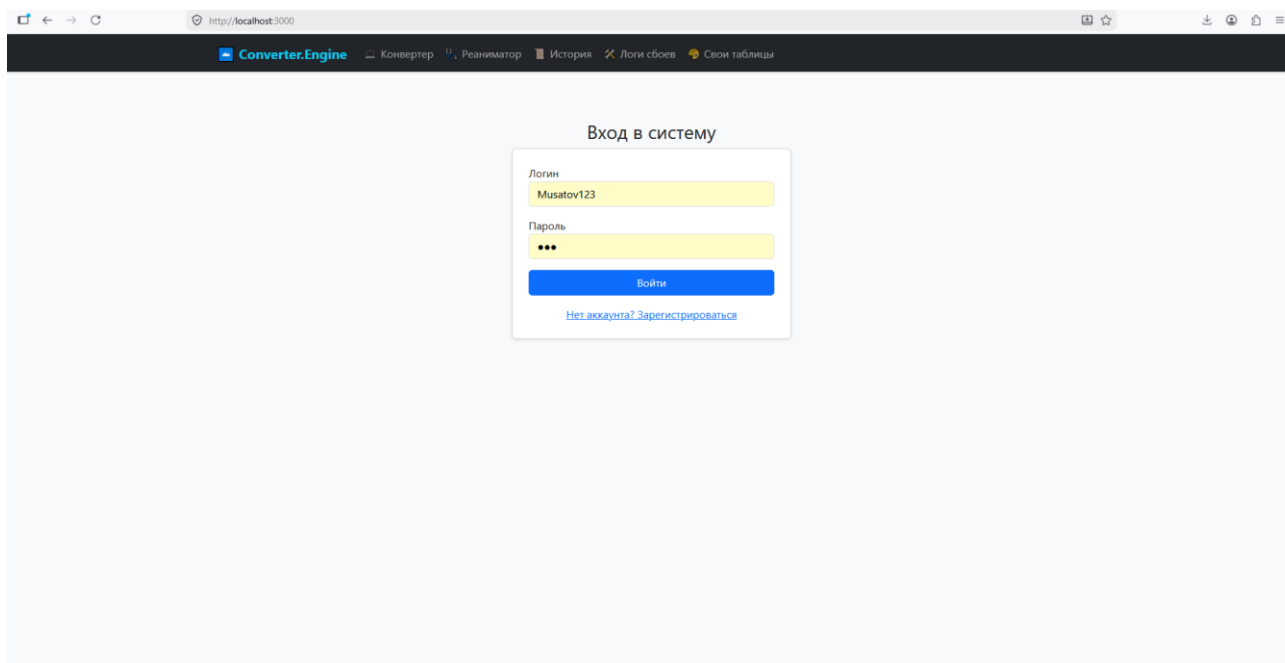


Рисунок В.1 – Успешный вход в учётную запись

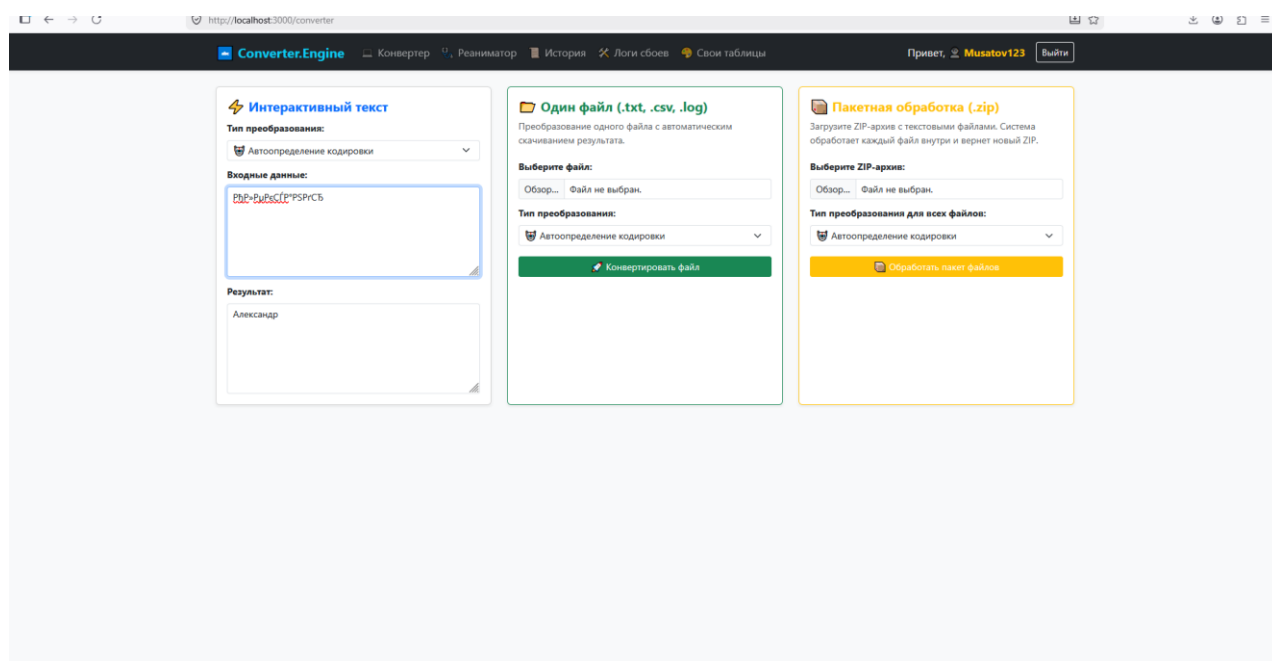


Рисунок В.2 – Ввод поврежденного текста в окно ввода

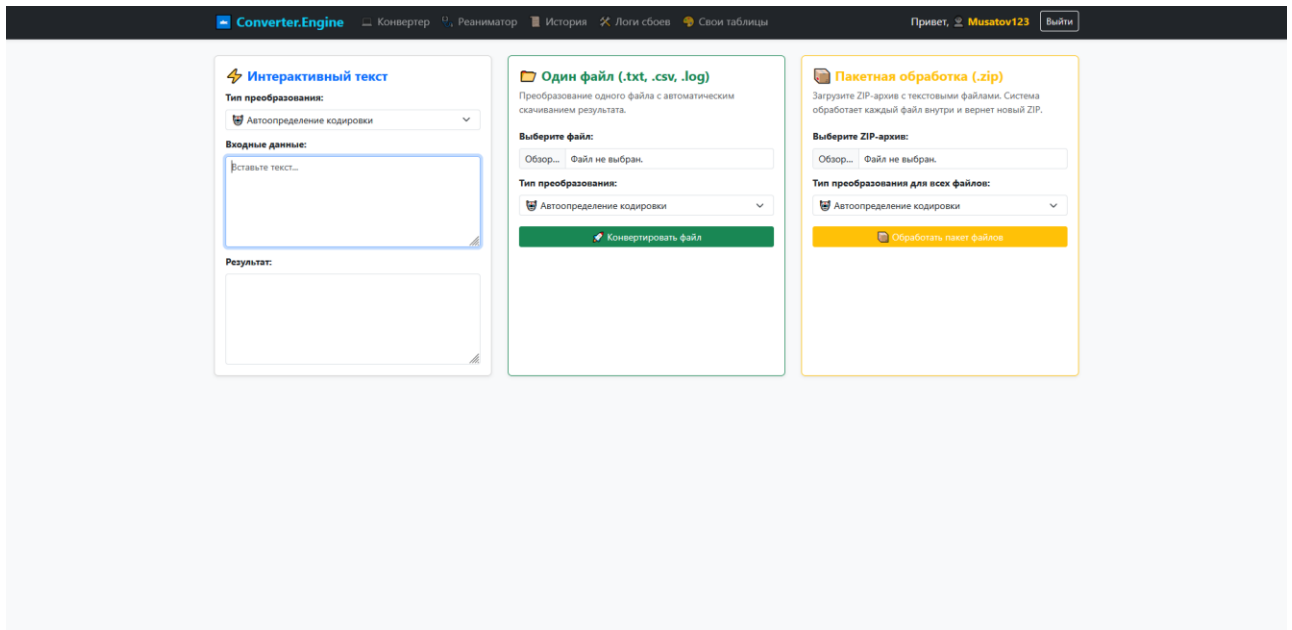


Рисунок В.3 – Пустой ввод данных

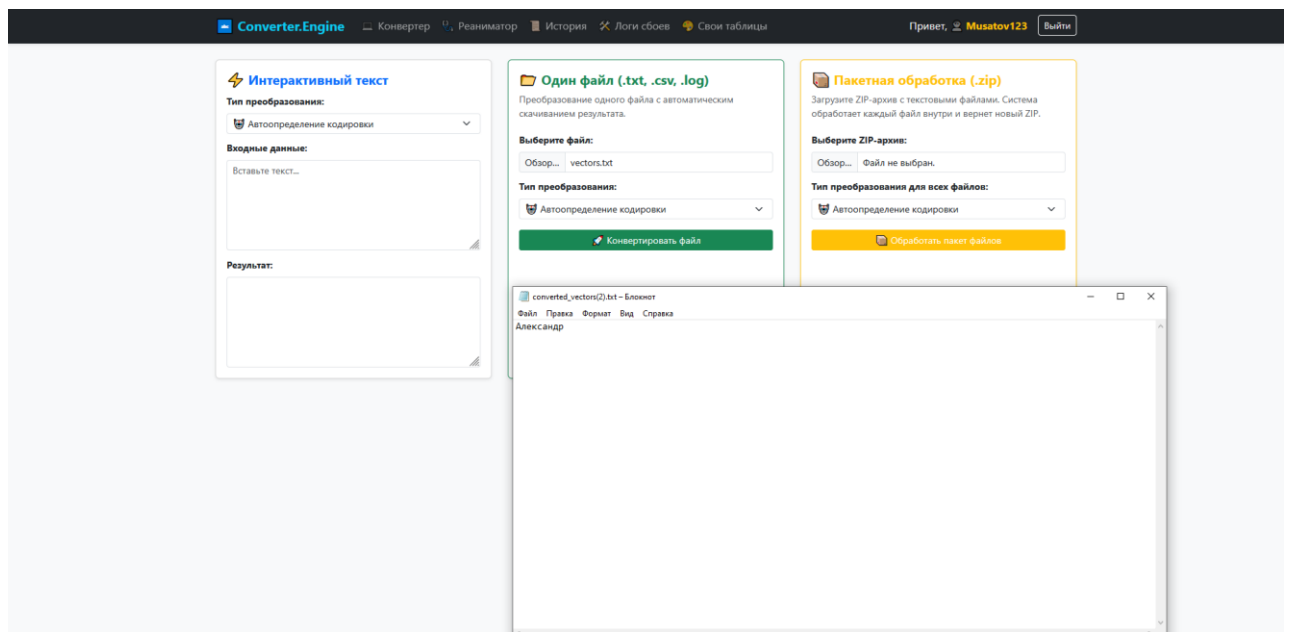


Рисунок В.4 – Проверка работы преобразования одного файла

	Конвертер	Реаниматор	История	Логи сброс	Свои таблицы	Привет, Musatov123	Выйти
--	-----------	------------	---------	------------	--------------	--------------------	-------

История операций

Дата	Операция	Исходный текст	Результат
2026-06-16 16:39:24	koi8-a	РЪР=РµРєСђР"PSРґСЊБ	РЪР=РµРєСђР"PSРґСЊБ
2026-06-16 16:39:22	utf8	РЪР=РµРєСђР"PSРґСЊБ	РЪР=РµРєСђР"PSРґСЊБ
2026-06-16 16:39:20	cp1251	РЪР=РµРєСђР"PSРґСЊБ	Александр
2026-06-16 16:39:17	koi8-a	РЪР=РµРєСђР"PSРґСЊБ	РЪР=РµРєСђР"PSРґСЊБ
2026-06-16 16:39:15	auto	РЪР=РµРєСђР"PSРґСЊБ	Александр
2026-06-16 16:35:31	auto	Архив: st.zip	Пакет из 1 файлов обработан
2026-06-16 16:33:20	auto	Архив: st.zip	Пакет из 1 файлов обработан
2026-06-16 16:27:40	auto	Архив: st.zip	Пакет из 1 файлов обработан
2026-06-16 16:07:31	auto	Файл: vectors.txt	Обработан и выдан на скачивание
2026-06-16 15:40:32	cp1251	Файл: vectors.txt	Обработан и выдан на скачивание
2026-06-16 15:38:32	auto	Файл: vectors.txt	Обработан и выдан на скачивание
2026-06-16 15:36:42	auto	РЪР=РµРєСђР"PSРґСЊБ	Александр
2026-06-16 15:36:36	auto	Результат тестирования представлен на рисунк...	Результат тестирования представлен на рисунк...
2026-06-16 15:36:34	auto	Результат тестирования представлен на рисунк...	Результат тестирования представлен на рисунк...
2026-06-16 15:15:05	auto	РЪР=РµРєСђР"PSРґСЊБ	Александр
2026-06-16 15:15:03	url_encode	РЪР=РµРєСђР"PSРґСЊБ	%420%452%420%BB%420%B5%420%454%421%...
2026-06-16 15:15:01	url_decode	РЪР=РµРєСђР"PSРґСЊБ	РЪР=РµРєСђР"PSРґСЊБ
2026-06-16 15:14:59	url_encode	РЪР=РµРєСђР"PSРґСЊБ	%420%452%420%BB%420%B5%420%454%421%...
2026-06-16 15:14:58	base64_decode	РЪР=РµРєСђР"PSРґСЊБ	

Рисунок В.6 – Вывод последней операции в журнал истории операций

Рисунок В.7 – Проверка работы модуля «Реаниматор»