ИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ(ТУ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Автоматизированные системы обработки информации и управления | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного  проектирования и управления | | |
| Учебная дисциплина | |  | **Интернет-технологии** | | |
| Курс | III | | | Группа | 404 |

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| **Тема:** | **Разработка веб-приложения для шифрования текста** |
| **Задача:** | **Разработать программный комплекс с возможностью шифровать текст** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Д.К. Азаров |
| Руководитель |  |  |  | А.А. Иванов |
| Лектор |  |  |  | А.С. Разыграев |
| Оценка за курсовой проект |  |  |  |  |

**СОДЕРЖАНИЕ**

[1 Аналитический обзор. 3](#_Toc132830780)

[1.1 Описание предметной области. Формирование функциональных требований к информационной системе. 3](#_Toc132830781)

[1.2 Обзор и обоснование выбора технологий верстки макетов. 4](#_Toc132830782)

[1.3 Обзор и обоснование выбора архитектуры информационной системы, способа клиент-серверного взаимодействия, используемой СУБД, инструментальных средств разработки. 9](#_Toc132830783)

[2 Технологическая часть – Технология разработки программного комплекса. 11](#_Toc132830784)

[2.1 Разработка функциональной структуры информационной системы с учетом клиент-серверной архитектуры. Постановка задачи разработки. 11](#_Toc132830785)

[2.2 Разработка UML-диаграммы вариантов использования программного комплекса. 13](#_Toc132830786)

[2.3 Разработка макета графического интерфейса промышленной информационной системы. 13](#_Toc132830787)

[2.4 Разработка блок-схемы алгоритма решения задачи, которую решает информационная система. 33](#_Toc132830788)

[2.5 Проектирование серверного интерфейса взаимодействия с клиентом и СУБД. 34](#_Toc132830789)

[2.6 Разработка базы данных информационной системы. 53](#_Toc132830790)

[2.7 Разработка программного обеспечения информационной системы. 54](#_Toc132830791)

[2.8 Тестирование программного комплекса на примере конкретного объекта (информационного, математического, аппаратно-технологического оформления). 72](#_Toc132830792)

[ВЫВОДЫ ПО РАБОТЕ 75](#_Toc132830793)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 76](#_Toc132830794)

# Аналитический обзор.

# Описание предметной области. Формирование функциональных требований к информационной системе.

Шифрование – обратимое преобразование [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) в целях сокрытия от [неавторизованных](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) лиц с предоставлением в это же время [авторизованным](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) пользователям доступа к ней. Главным образом, шифрование служит для соблюдения [конфиденциальности](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) передаваемой информации. Важной особенностью любого алгоритма шифрования является использование [ключа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)), который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.

Пользователи являются авторизованными, если они обладают определённым [аутентичным](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) [ключом](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)). Вся сложность и, собственно, задача шифрования состоит в том, как именно реализован этот процесс. Шифрование состоит из двух составляющих: зашифрование и расшифрование.

С помощью шифрования обеспечиваются три состояния безопасности информации:

* Конфиденциальность – шифрование используется для скрытия информации от неавторизованных пользователей при передаче или при хранении.
* Целостность – шифрование используется для предотвращения изменения информации при передаче или хранении.
* Идентифицируемость – шифрование используется для аутентификации источника информации и предотвращения отказа отправителя информации от того факта, что данные были отправлены именно им.

Для того, чтобы прочитать зашифрованную информацию, принимающей стороне необходимы [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) и [дешифратор](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%88%D0%B8%D1%84%D1%80%D0%B0%D1%82%D0%BE%D1%80) (устройство, реализующее алгоритм расшифровывания). Идея шифрования состоит в том, что злоумышленник, перехватив зашифрованные данные и не имея к ним ключа, не может ни прочитать, ни изменить передаваемую информацию. Кроме того, в современных [криптосистемах (с открытым ключом)](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC) для шифрования, расшифрования данных могут использоваться разные ключи. Однако, с развитием криптоанализа, появились методики, позволяющие дешифровать закрытый текст без ключа. Они основаны на математическом анализе переданных данных [1].

**Цель проекта:** разработать интерфейса веб-приложения для шифрования текста.

**Функциональные требования к разрабатываемому веб-приложению:**

1) Отображать алгоритмы.

2) Возможность ввода ключа.

3) Возможность Ввода исходного текста.

4) Отображать описание алгоритма.

# Обзор и обоснование выбора технологий верстки макетов.

**Верстание** — формирование структуры будущего сайта в HTML-документе и перенос дизайнерского наброска в такой документ. Перенос макета в веб-страницу происходит благодаря использованию HTML, CSS, JavaScript и клиентских сценариев. Цель верстки сайта — сделать так, чтобы макет был перенесен в реальный сайт максимально точно и близко дизайнерской задумке, с сохранением расположения текста, заголовков, изображений, других медиафайлов, таблиц [[2]](#_СПИСОК_ЛИТЕРАТУРЫ).

Говоря о верстании веб-страниц, нельзя не упомянуть конкретные методы, которые используются для переноса дизайн-макета в документ. Не стоит путать методы с видами — это разные термины. Итак, в 2023 году активно используются четыре метода верстания страниц: фреймы, табличный способ, блочный способ (div-контейнерами) и слои. Каждая техника имеет свой набор достоинств / недостатков. Давайте посмотрим, какие именно.

1. **Верстание фреймами** — убирает повторы в коде и делает итоговый файл гораздо легче по размеру, но такие страницы плохо индексируется ПС (ведь они не имеют уникального адреса) [[2]](#_СПИСОК_ЛИТЕРАТУРЫ).

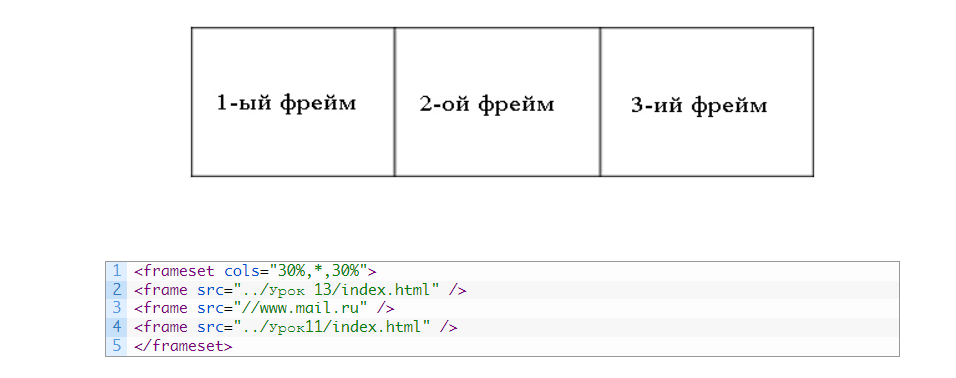


Рисунок 1 – Фреймовая верстка

1. **Табличный метод** верстания долгое время был стандартом в вебе, но сегодня — он активно сдает позиции (ведь этот метод понижает скорость загрузки сайта, ухудшает индексацию, да и сам код, в итоге, выглядит не оптимизированным, так как предусматривает большое количество пустых таблиц).

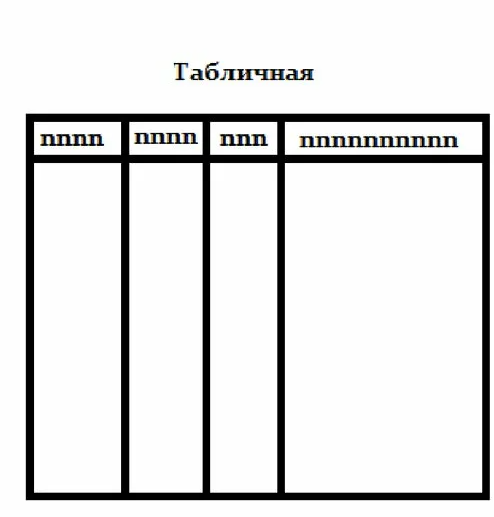


Рисунок 2 – Табличная верстка

1. **Блочный метод верстания** — считается одним из лучших, однако, он ухудшает фактор кроссбраузерности. Зато именно блочный метод предусматривает минимальный размер кода, его высокую читабельности. Именно блочный метод верстания страниц развязывает руки дизайнерам — можно внедрить любое, даже сложное решение. Также сайты с блочной версткой лучше всего индексируется, по сравнению с теми, которые используют иные методы верстания страниц [[2]](#_СПИСОК_ЛИТЕРАТУРЫ).

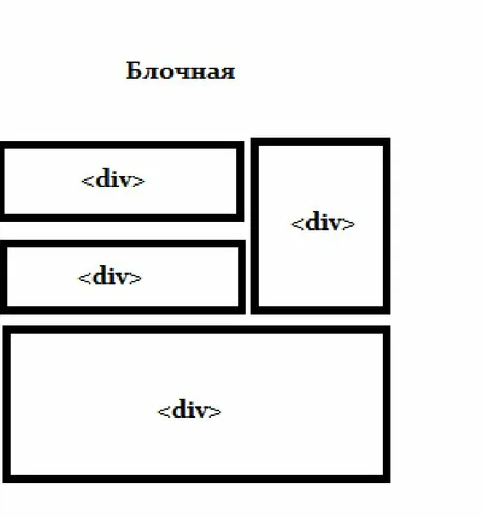


Рисунок 3 – Табличная верстка

# Обзор и обоснование выбора архитектуры информационной системы, способа клиент-серверного взаимодействия, используемой СУБД, инструментальных средств разработки.

**Многослойная архитектура** работает по принципу разделения ответственностей. Программное обеспечение разделено на слои, лежащие друг на друге, и каждый из них выполняет определенную обязанность. Преимущества: более простая реализация по сравнению с другими подходами; предлагает абстракцию благодаря разделению ответственностей между уровнями; изолирование защищает одни слои от изменений других; повышает управляемость программного обеспечения за счет слабой связанности. Недостатки: не предлагает большой масштабируемости; программное обеспечение, созданное с таким подходом, будет иметь монолитную структуру, усложняющую внесение модификаций; данные должны проходить по каждому слою, даже если нет необходимости передавать их с определенных слоев.

**Многоуровневая архитектура** разделяет комплекс программного обеспечения на уровни по принципу взаимодействия “клиент-сервер”. Архитектура может иметь один, два и больше уровней, разделяющих ответственности между поставщиком данных и потребителем. Этот подход использует шаблон Request Response для связи между уровнями. В отличие от многослойной архитектуры, он предлагает масштабируемость, которая может быть как горизонтальной (масштабирование сети с помощью высокопроизводительных узлов), так и вертикальной (масштабирование каждого узла путем повышения его производительности).

**Сервис-ориентированная архитектура** состоит из компонентов и приложений, которые связываются друг с другом с помощью четко определенных сервисов. Клиент отправляет запрос с использованием стандартного протокола и формата данных по сети. Этот запрос обрабатывается ESB (enterprise service bus — сервисная шина предприятия), которая считается сердцем сервис-ориентированной архитектуры и отвечает за оркестровку и маршрутизацию. С помощью сервисного репозитория ESB направляет запрос в специальный сервис, который может взаимодействовать с другими сервисами и базами данных, чтобы составить полезную нагрузку (данные) ответа.

**Mикросервисная архитектура** разрабатывается как набор небольших сервисов, каждый из которых работает в собственном процессе и связывается с легковесными механизмами, обычно API для HTTP-ресурса. Преимущества:

предлагает слабую связанность благодаря высокой степени изоляции; повышает модульность; cбой в одном сервисе не затронет всю систему, поскольку они изолированы; предлагает высокую гибкость и масштабируемость; простота модификации может ускорить итерации; позволяет реализовать улучшенную систему обработки ошибок; решает проблемы с потоками данных, которые бывают у многослойной архитектуры. Недостатки: повышенный риск сбоя при обмене данными между сервисами; большим количеством сервисов трудно управлять; требует решения таких проблем, как задержки в сети, балансировка нагрузки и прочих трудностей, свойственных распределенной архитектуре; нуждается в комплексном тестировании в распределенной среде; на реализацию потребуется гораздо больше времени [3].

Для реализации проекта выбрана трёхуровневая архитектура приложения, представленная на рисунке 4.

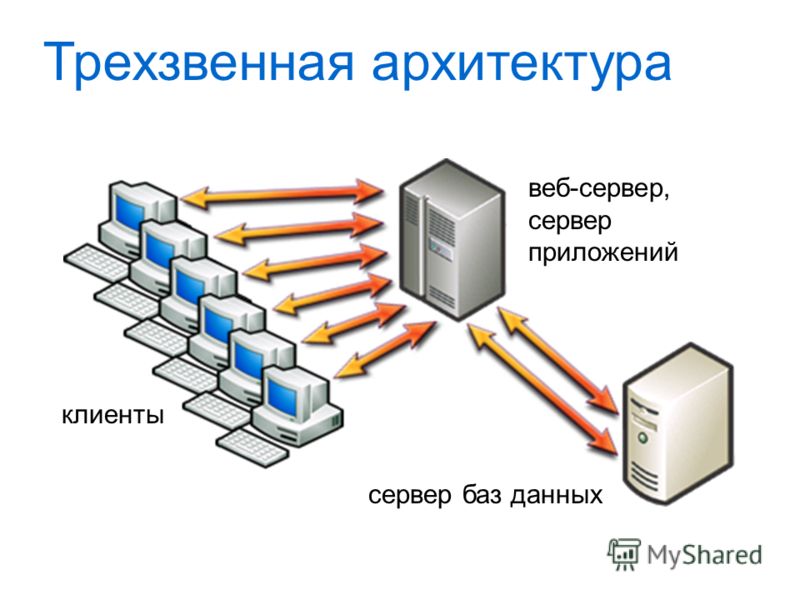


Рисунок 4 – Схема трехуровневой архитектура

***Выбор базы данных для:***

Таблица 1 – Сравнение систем управления базами данных

|  |  |  |
| --- | --- | --- |
| Критерий | MySQL | PostgreSQL |
| Открытый код | + | + |
| ACID | Частично | + |
| Высокая производительность | + | Только при чтении данных |
| Поддержка NoSQL функции | - | + |
| Хранимые процедуры | - | + |

Для реализации в проекте хранения данных выбрана СУБД PostgreSQL обладающая весомыми преимуществами в рамках, а именно:

– имеет поддержку хранимых процедур, которые повысят скорость выполнения запросов, в отличие от MySQL;

– является более производительным и имеет полную поддержку ACID (atomicity, consistency, isolation, durability) [4].

# Технологическая часть – Технология разработки программного комплекса.

# Разработка функциональной структуры информационной системы с учетом клиент-серверной архитектуры. Постановка задачи разработки.

Функциональная структура веб-приложения для шифрования, дешифрования данных и вывода частоты использования алгоритмов администратору представлена на рисунке 5. Функциональная структура представлена в виде модели клиент-сервер. Клиент может отправлять на сервер HTTP запросы Ruquest и получать от сервера HTTP ответы Response.

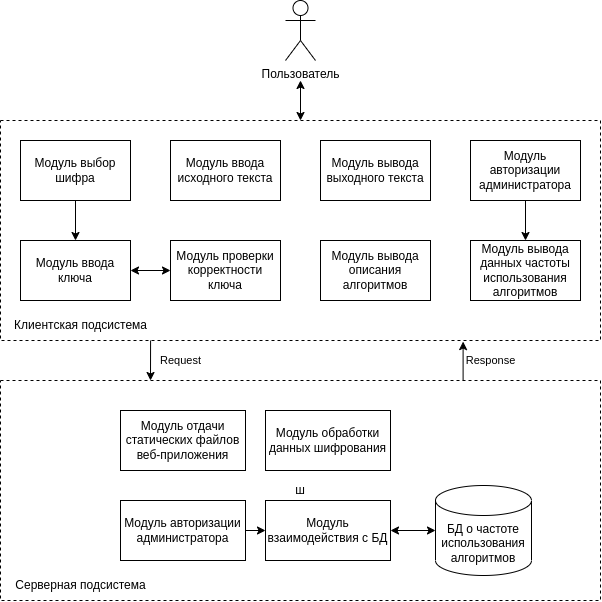


Рисунок 5 – Функциональная структура веб-приложения

# Разработка UML-диаграммы вариантов использования программного комплекса.

На рисунке 6 представлена UML-диаграммы вариантов использования веб-приложения.

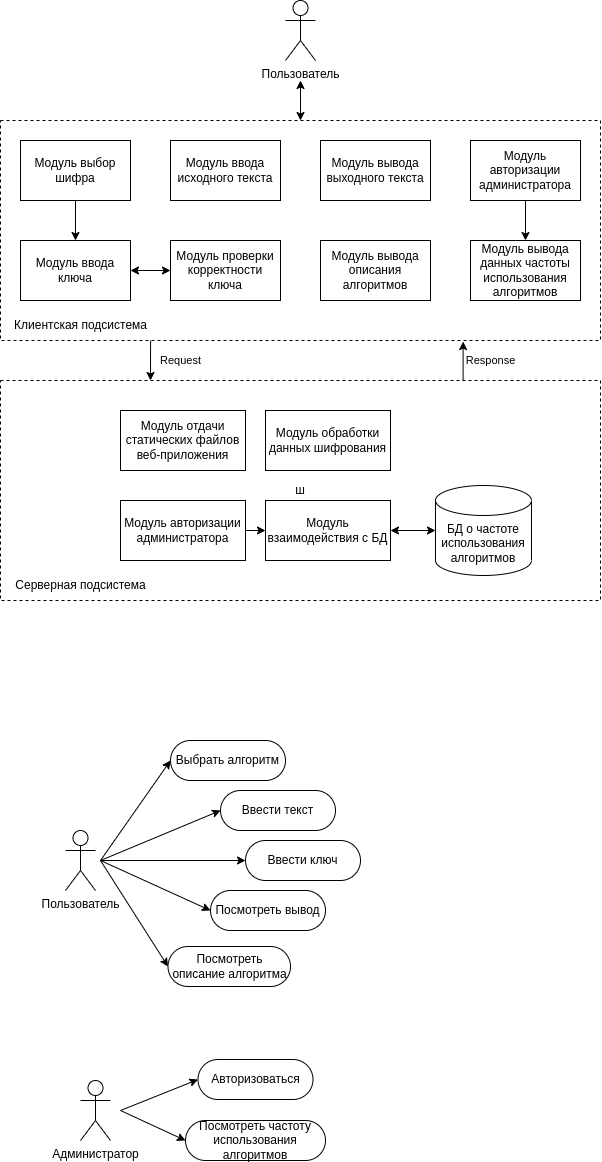


Рисунок 6 – UML-диаграммы вариантов использования веб-приложения

# Разработка макета графического интерфейса промышленной информационной системы.

Разрабатываемое веб-приложение будет состоять из заголовка с алгоритмами шифрования и страницами с исходным текстом, текстом результата, ключем и описание алгоритма. На рисунке 7 представлен схематичный макет страницы веб-приложения пользовательского интерфейса при открытой вкладки страниц.

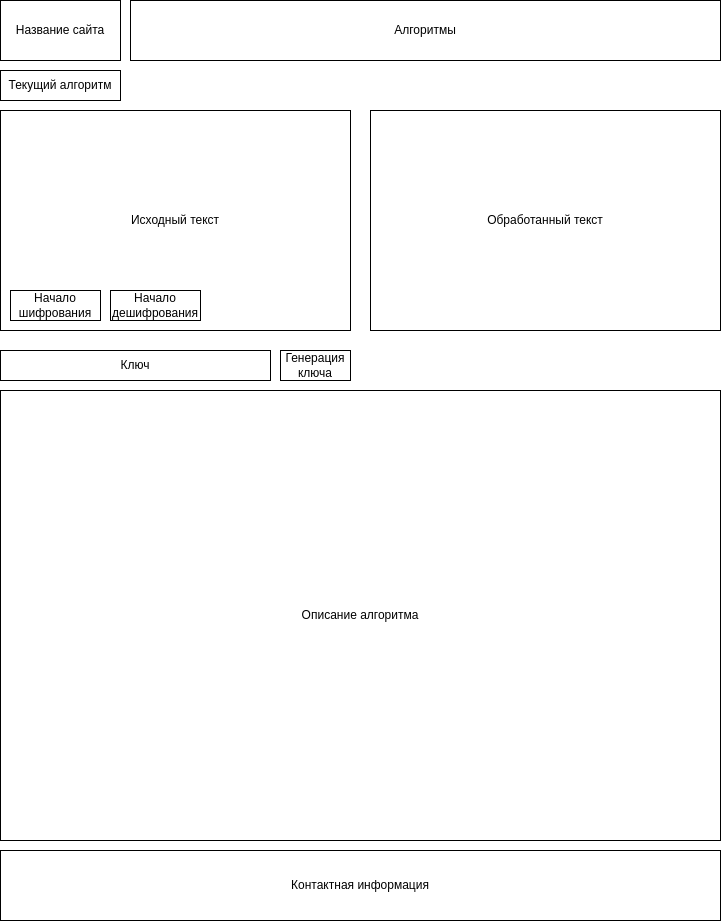


Рисунок 7 – Содержимое вкладки страниц с рецептами

# Разработка блок-схемы алгоритма решения задачи, которую решает информационная система.

На рисунке 8 представлена блок-схема алгоритма шифрования.

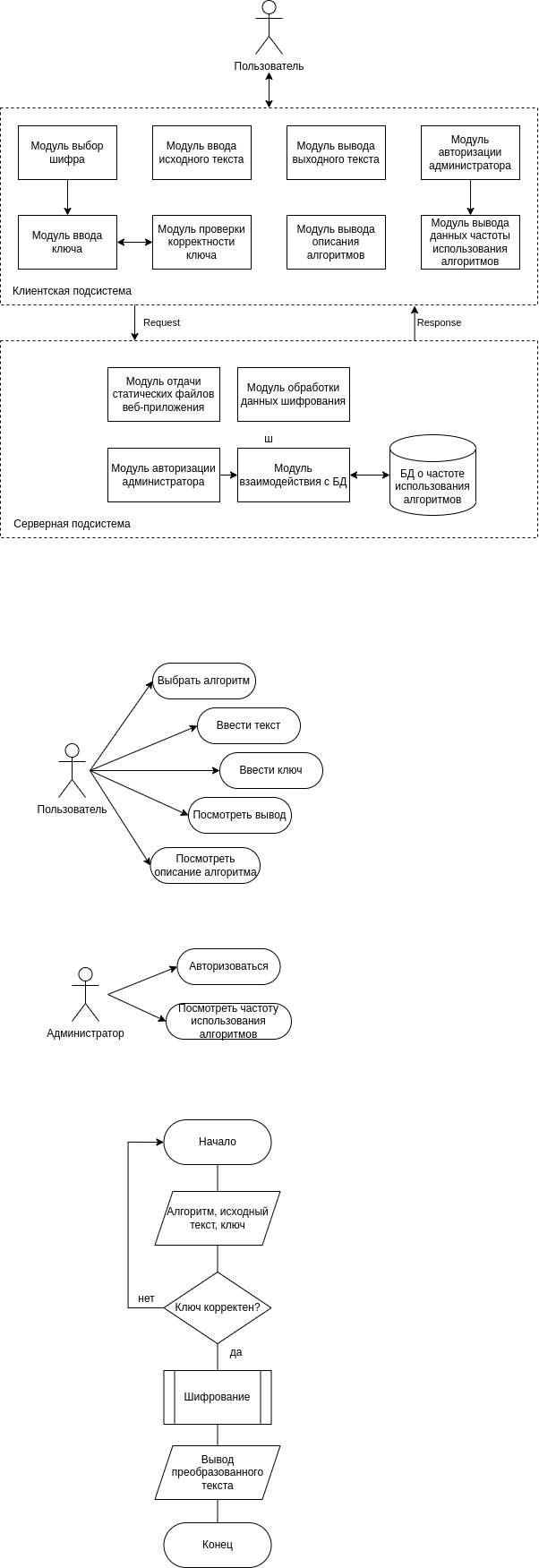


Рисунок 8 – Блок-схема алгоритма загрузки данных

# Проектирование серверного интерфейса взаимодействия с клиентом и СУБД.

В таблице 2 представлено описание API обращения клиента к серверу по основным запросам.

Таблица 2 – Описание API обращения клиента к серверу по основным запросам

|  |  |  |
| --- | --- | --- |
| Метод HTTP запроса | Относительный путь запроса | Назначение запроса |
| GET | / | Страница шифрования |
| POST | / | Шифрование текста |
| GET | /admin-login | Страница авторизация администратора |
| POST | /admin-login | Отображение таблицы частоты использования алгоритмов |

# Разработка базы данных информационной системы.

Структура БД представлена в таблице 3.

Таблица3 – Структура сущностей

|  |  |  |
| --- | --- | --- |
| Поле(я) | Тип данных | Значение |
| frequency\_algo(algo\_name, frequency) | text, int | Название алгоритма, частота использования |

# Разработка программного обеспечения информационной системы.

Далее представлен листинг программы.

[index.html]

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<title>Ciphers</title>

<link rel="stylesheet" href="styles.css">

<script src="script.js">script.js</script>

</head>

<body>

<header>

<nav>

<img id="logo" src="src/logo.jpg" alt="Шифр">

<ul id="topmenu">

<li>AES</li>

<li>BLOWFISH</li>

<li>CAST</li>

<li>DES</li>

</ul>

</nav>

</header>

<div id="page">

<h2 id="cipher\_name">AES</h2>

<div class="wrapper">

<div>Исходный текст</div>

<div>Преобразованный текст</div>

<div class="text\_box"><textarea id="in\_text"></textarea></div>

<div class="text\_box"><textarea id="out\_text" readonly></textarea></div>

<div id="ciph\_btns\_box">

<button id="ciph\_btn" type="button">Кодировать</button>

<button id="deciph\_btn" type="button">Декодировать</button>

</div>

<div></div>

<div>

<input type="text" size="40" id="key\_box" placeholder="Ключ" autocomplete="off" required>

<button type="button" id="generate\_key">Сгенерировать</button>

</div>

<div></div>

</div>

<article>

<h3>Алгоритм</h3>

<div id="AES-alghoritm" class="show">

<img src="src/aes-alghoritm.jpg" alt="Шифр AES">

<ol>

<li>Сообщение разбивается на блоки одинакового размера. При необходимости последний блок дополняется до стандартной длины 128 бит.</li>

<li>Шифрование очередного блока сообщения выполняется с использованием предыдущего зашифрованного блока.

Для первого блока предыдущего зашифрованного блока не существует, поэтому его шифруют с использованием «вектора инициализации» (<strong>initialization vector</strong>, <strong>IV</strong>).</li>

<li>Вектор инициализации — случайное число. Размер (длина) IV равна размеру блока.</li>

<li>Расшифровка выполняется с использованием тех же ключа k и вектора инициализации IV.</li>

</ol>

<div class="clear\_floating"></div>

</div>

<div id="BLOWFISH-alghoritm" class="hide">

<img src="src/blowfish-alghoritm.png" alt="Шифр BLOWFISH">

<ol>

<li>Исходные данные разбиваются на блоки фиксированной длины (как правило кратно степени двойки — 64 бит, 128 бит). В случае если длина блока исходных данных меньше длины разрядности шифра, то блок дополняется каким-либо заранее известным образом.</li>

<li>Блок делится на два равных подблока — «левый» L<sub>0</sub> и «правый» R<sub>0</sub>. В случае 64-битной разрядности — на два блока с длиной 32 бита каждый.</li>

<li>«Левый подблок» L<sub>0</sub> видоизменяется функцией итерации <i>F</i>(L<sub>0</sub>, P<sub>0</sub>) в зависимости от ключа P<sub>0</sub>, после чего он складывается по модулю 2 (XOR) с «правым подблоком» R<sub>0</sub>.</li>

<li>Результат сложения присваивается новому левому подблоку L<sub>1</sub>, который становится левой половиной входных данных для следующего раунда, а «левый подблок» L<sub>0</sub> присваивается без изменений новому правому подблоку R<sub>1</sub>, который становится правой половиной.</li>

<li>Эта операция повторяется n-1 раз, при этом при переходе от одного этапа к другому меняются раундовые ключи (P<sub>0</sub>, P<sub>1</sub>, P<sub>2</sub> и т.д.), где n — количество раундов для используемого алгоритма.</li>

</ol>

<div class="clear\_floating"></div>

</div>

<div id="CAST-alghoritm" class="hide">

<img src="src/cast-alghoritm.webp" alt="Шифр CAST">

<ol>

<li>(развертка ключа) составляет 16 пар подключей {Kmi, Kri} полученных из K (см. разделы Пары раундовых ключей и Неидентичные раунды).</li>

<li>(L0, R0) <- (m1. .. m64). (Разделяет текст на левую и правую 32-битные половины L0 = m1 ... m32 и R0 = m33 ... m64).</li>

<li>(16 раундов) for i from 1 to 16, вычислить Li и Ri следующим образом: Li = Ri-1; Ri = Li-1 ^ F(Ri-1,Kmi,Kri), где F определена в разделе «Пары раундовых ключей» (F имеет тип 1, тип 2, тип 3 или, в зависимости от i).</li>

<li>c1 ... c64 <- (R16, L16). (Меняем окончательные блоки местами L16, R16 и объединяем, чтобы сформировать зашифрованный текст.)</li>

</ol>

<div class="clear\_floating"></div>

</div>

<div id="DES-alghoritm" class="hide">

<img src="src/des-alghoritm.png" alt="Шифр DES">

<ol>

<li>довести исходное сообщение до такого размера (в битах), чтобы оно нацело делилось на размер блока (sizeOfBlock = 128 бит);</li>

<li>разделить исходное сообщение на блоки;</li>

<li>довести длину ключа до длины половины блока;</li>

<li>перевести ключ в бинарный формат (в нули и единицы);</li>

<li>провести над каждым блоком прямое преобразование сетью Фейстеля в течении 16-ти раундов. После каждого раунда необходимо выполнять циклический сдвиг ключа на заданное количество символов;</li>

<li>соединить все блоки вместе; таким образом получим сообщение, зашифрованное алгоритмом DES.</li>

</ol>

<div class="clear\_floating"></div>

</div>

</article>

</div>

<footer>

<p>Copyright © 2023, Азаров Даниил</p>

</footer>

</body>

</html>

[index.html]

[admin.html]

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<title>Ciphers</title>

<script src="admin-script.js">adminadminadmin-script.js</script>

<style type="text/css">

table, td

{

border: 1px solid #333;

width: 350px;

text-align: center;

}

#changeDateRange

{

display: inline;

}

</style>

</head>

<body>

<div align="center">

<table id="frTable">

<caption>Частота использования алгоритмов за

<select id="changeDateRange">

<option value="0">неделю</option>

<option value="1">месяц</option>

<option value="2">год</option>

</select>

</caption>

<tr>

<th>Алгоритм</th>

<th>Частота</th>

</tr>

</table>

</div>

</body>

</html>

[admin.html]

[admin-logging.html]

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<title>Ciphers</title>

<style type="text/css">

</style>

</head>

<body>

<div align="center">

<form method="post">

<h3>Вход</h3>

<input type="text" name="log" placeholder="Логин"><br>

<input type="password" name="pass" placeholder="Пароль"><br>

<button type="submit">Войти</button>

</form>

</div>

</body>

</html>

[admin-logging.html]

[styles.css]

body {

margin-left: 20%;

margin-right: 20%;

background-color: #111f24;

}

button {

/\* width:90px; \*/

text-align:center;

/\* display:block; \*/

font-family: arial;

text-decoration: none;

font-weight: 300;

font-size: 14px;

border: #0B1F24 1px solid;

color: #0B1F24;

padding: 3px;

padding-left: 5px;

padding-right: 5px;

/\* margin: 20px auto; \*/

transition: .5s;

border-radius: 0px;

margin: 2px;

}

button:hover {

top: 5px;

transition: .5s;

color: #049bf3;

border: #049bf3 1px solid;

border-radius: 10px;

}

button:active {

color: #0066ff;

border: #0066ff 1px solid;

transition: .07s;

background-color: #FFF;

}

/\* #main { \*/

/\* min-width: max-content; \*/

/\* } \*/

header > nav {

display: flex;

flex-direction: row;

align-items: center;

}

#logo {

display: inline-block;

width: 200px;

}

#topmenu li

{

margin: 10px;

display: inline-block;

text-decoration: none;

color: #fafdff;

font-size: 30px;

}

#page

{

background-color: white;

padding: 20px;

margin-top: 0px;

}

#ciph\_btns\_box

{

display: flex;

justify-content: flex-end;

margin-top: -35px;

}

.wrapper

{

display: grid;

grid-template-columns: 1fr 1fr;

grid-template-rows: 1fr 15fr;

align-items: center;

gap:5px 20px;

}

.text\_box

{

height: 100%;

width: 100%;

}

textarea

{

height: 100%;

width: 100%;

resize: none;

}

.ciph\_box

{

margin-top: -30px;

}

img

{

float: left;

margin: 20px;

}

.clear\_floating

{

clear: both;

}

footer

{

color: #fafdff;

}

.hide

{

display: none;

}

[styles.css]

[script.js]

window.onload = init

function init()

{

// Изменение шифра

var topMenuChilds = document.getElementById("topmenu").children

for (let el of topMenuChilds)

{

el.addEventListener("click", \_ => changeCipherType(el.textContent))

}

// Генерация ключа

let generateKeyBtn = document.getElementById("generate\_key")

generateKeyBtn.addEventListener("click", setGenerateKeyOnTexBox)

// let keyTextBox = document.getElementById("key\_box")

// keyTextBox.setCustomValidity("Что-то не правильно")

// Кодирование

let cipherBtn = document.getElementById("ciph\_btn")

cipherBtn.addEventListener("click", \_ => ciphText(false))

let decipherBtn = document.getElementById("deciph\_btn")

decipherBtn.addEventListener("click", \_ => ciphText(true))

}

function changeCipherType(name)

{

let header = document.getElementById("cipher\_name")

header.textContent = name

let shownAlghoritm = document.getElementsByClassName("show")[0]

shownAlghoritm.className = "hide"

let alghoritmManual = document.getElementById(name + "-alghoritm")

alghoritmManual.className = "show"

}

function setGenerateKeyOnTexBox()

{

let keyTextBox = document.getElementById("key\_box")

keyTextBox.value = generateKey()

}

function generateKey()

{

let shownAlghoritm = document.getElementById("cipher\_name").textContent

let count = getAlgoKeyLenght(shownAlghoritm)

let min = Math.pow(10, 16);

let max = Math.pow(10, 17)

let val = "";

for(let i = 0; i < 10; ++i)

{

val += Math.floor(Math.random() \* (max - min) + min).toString(36)

}

return val.slice(-count)

}

function getAlgoKeyLenght(name)

{

let count = 0;

switch (name)

{

case "AES":

count = 128 //128/192/256 бит

break

case "BLOWFISH":

count = 32 //от 32 до 448 бит

break

case "CAST":

count = 64 //40-128 бит

break

case "DES":

count = 64 //56 бит + 8 проверочных бит

break

}

let v = count / 8

return count/8

}

function ciphText(isDecipher)

{

let keyText = document.getElementById("key\_box")

let typeCiph = document.getElementById("cipher\_name").textContent

if (!checkKeyValidity(keyText.value, typeCiph))

{

keyText.setCustomValidity(`Ключ должен содержать ${getAlgoKeyLenght(typeCiph)} символов`);

keyText.reportValidity()

return

}

let inText = document.getElementById("in\_text").value

let outText = document.getElementById("out\_text")

let req =

{

isDeciph: isDecipher,

type: typeCiph,

text: inText,

key: keyText.value

}

fetch("/",

{

method: 'POST',

headers: { 'Content-Type': 'application/json;charset=utf-8' },

body: JSON.stringify(req)

})

.then(res => res.text())

.then(text => outText.textContent = text)

}

function checkKeyValidity(key, ciphType)

{

return key.length == getAlgoKeyLenght(ciphType)

}

[script.js]

[admin-script.js]

window.onload = init

var dates = {};

async function init()

{

//заолняем dates

await initDates();

//заполняем таблицу

initTable();

let changeDateRange = document.getElementById("changeDateRange")

changeDateRange.addEventListener("click", \_ => range(changeDateRange.value))

}

function initTable()

{

let table = document.getElementById("frTable");

let range = document.getElementById("changeDateRange");

for (let algo in dates)

{

let newRow = table.insertRow(-1);

let newCell = newRow.insertCell(0);

newCell.innerHTML=algo;

let newCell2 = newRow.insertCell(1);

newCell2.innerHTML=dates[algo][range.value];

}

}

function range(val)

{

let table = document.getElementById("frTable");

let cells = table.getElementsByTagName("td");

let i = 1;

for (let algo in dates)

{

cells[i].innerText = dates[algo][val];

i+=2;

}

}

async function initDates()

{

await fetch("/admin-dates",

{

method: 'POST',

})

.then(res => res.json())

.then(j => dates = j);

}

[admin-script.js]

[server.js]

const express = require("express");

const bodyParser = require('body-parser')

const crypto = require('crypto')

var pg = require("pg");

const e = require("express");

// const { log } = require("console");

const fs = require('fs').promises;

//export NODE\_OPTIONS=--openssl-legacy-provider//

//Сайт

const app = express();

app.use(express.static(\_\_dirname));

const urlencodedParser = express.urlencoded({extended: false});

var jParser = bodyParser.json();

//БД

var client = new pg.Client("postgres://postgres:1@localhost:5432/cipher");

client.connect();

// Settings

const port = 3000;

const adminLogin = "admin";

const adminPass = "admin";

let isAdminLogined = false;

const cryptos = [ "aes-128-cbc", "blowfish", "cast", "des" ]

const iv8 = Buffer.from([0x62, 0x75, 0x66, 0x66, 0x65, 0x72, 0xfa, 0x8a]);

const iv16 = Buffer.from([0x1e, 0xd5, 0x83, 0x2c, 0x1f, 0xf0, 0xdc, 0xb0, 0x95, 0xfa, 0xec, 0x7e, 0x8a, 0xfe, 0xc6, 0xb3]);

const tableName = "frequency\_algo";

const algoNameColumn = "algo";

const dateColumn = "date";

function logCipherCount(type)

{

var today = new Date();

var dd = String(today.getDate()).padStart(2, '0');

var mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!

var yyyy = today.getFullYear();

client.query(`INSERT INTO ${tableName} values('${type}', '${yyyy + '-' + mm + '-' + dd }')`);

}

app.post("/", jParser, function (req, res)

{

const isDeciph = req.body.isDeciph

const inText = req.body.text;

const key = req.body.key;

let type = "";

for (const val of cryptos)

{

if(val.includes(req.body.type.toLowerCase()))

{

type = val

logCipherCount(type)

break

}

}

let iv = iv8

if(type == "aes-128-cbc") iv = iv16

let outText;

if (isDeciph)

{

let cipher = crypto.createDecipheriv(type, key, iv);

outText = cipher.update(inText, 'hex', 'utf8') + cipher.final('utf8');

}

else

{

let cipher = crypto.createCipheriv(type, key, iv);

outText = cipher.update(inText, 'utf8', 'hex') + cipher.final('hex');

}

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end(outText);

});

app.get("/admin", function (req, res)

{

let file = "/admin-logging.html";

if (isAdminLogined)

{

isAdminLogined = false;

file = "/admin.html"

}

fs.readFile(\_\_dirname + file)

.then(contents => {

res.setHeader("Content-Type", "text/html");

res.writeHead(200);

res.end(contents);

})

});

app.post("/admin-dates", function (req, res)

{

dates(res)

});

async function dates(res)

{

let arr = {};

let i = 0;

for (const type of cryptos)

{

arr[type] = {};

arr[type][0] = await client.query(`SELECT COUNT(\*) FROM ${tableName}

WHERE ${algoNameColumn} = '${type}' AND (now() - interval '7 day') < ${dateColumn}`)

.then((result) =>

{

return result.rows[0].count;

});

arr[type][1] = await client.query(`SELECT COUNT(\*) FROM ${tableName}

WHERE ${algoNameColumn} = '${type}' AND (now() - interval '31 day') < ${dateColumn}`)

.then((result) =>

{

return result.rows[0].count;

});

arr[type][2] = await client.query(`SELECT COUNT(\*) FROM ${tableName}

WHERE ${algoNameColumn} = '${type}' AND (now() - interval '356 day') < ${dateColumn}`)

.then((result) =>

{

return result.rows[0].count;

});

}

res.setHeader("Content-Type", "application/json;charset=utf-8");

res.writeHead(200);

res.end(JSON.stringify(arr));

}

app.post("/admin", urlencodedParser, function (req, res)

{

isAdminLogined = req.body.log == adminLogin && req.body.pass == adminPass;

res.redirect("/admin");

});

app.listen(port, "localhost", () =>

{

console.log(`listening port ${port}`);

});

[server.js]

# Тестирование программного комплекса на примере конкретного объекта (информационного, математического, аппаратно-технологического оформления).

На рисунках 9-13 представлены результаты работы сайт

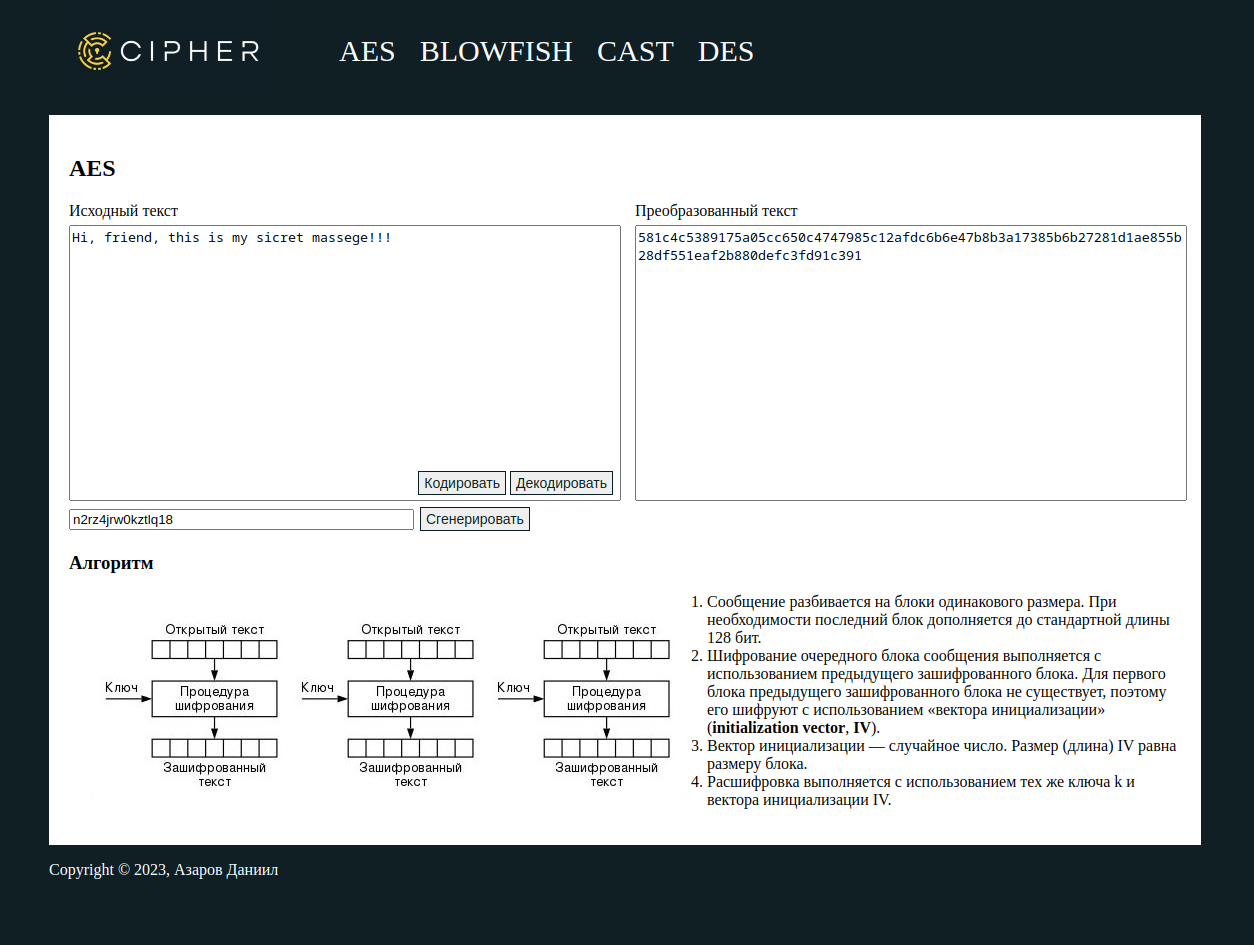


Рисунок 9 – Результат работы программы

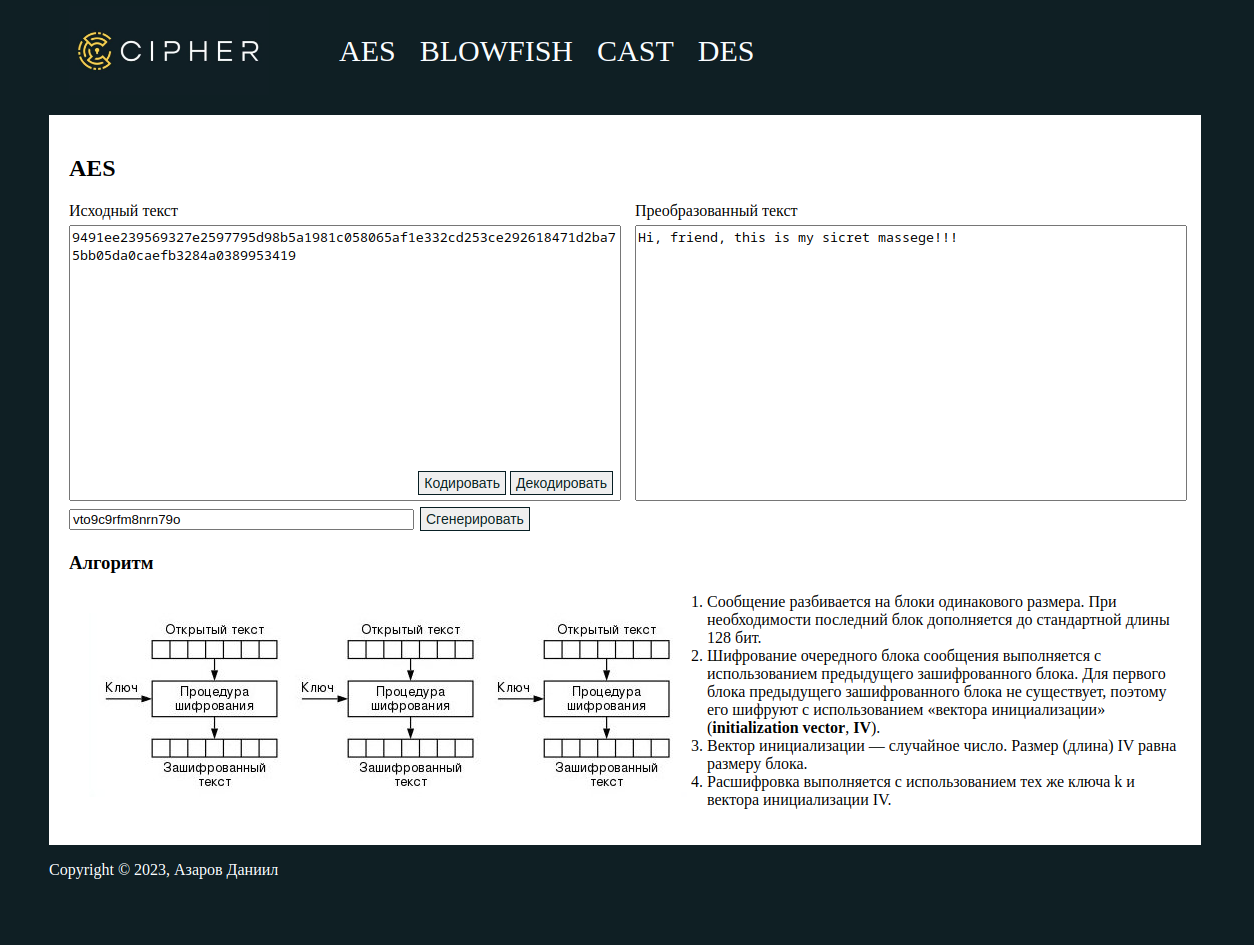


Рисунок 10 – Результат работы программы

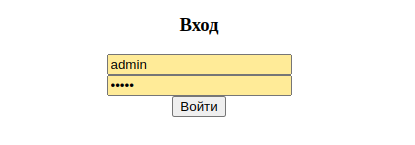


Рисунок 11 – Результат работы программы

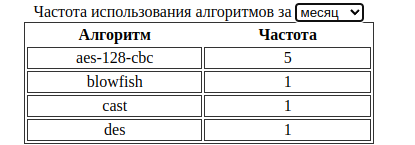


Рисунок 12 – Результат работы программы

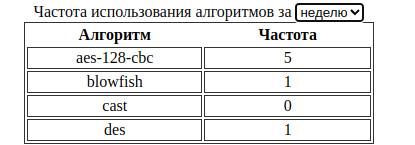


Рисунок 13 – Результат работы программы

# ВЫВОДЫ ПО РАБОТЕ

В ходе выполнения курсового проекта разработано веб-приложение, позволяющее шифровать тексты разными алгоритмами aes, blowfish, cast и des, а так же наблюдать частоту использования каждого алгоритма администратором:

– На основании аналитического обзора разработана функциональная структура веб-приложения, включающего базу пользователей, постов, комментариев к постам, директов, лайков, модуль авторизации, модуль отображения графических и текстовых данных;

– Сформулирована задачи разработки веб-приложения для обмена информацией;

– Разработаны структуры интерфейсов пользователя;

– Разработаны макеты графического интерфейса и структура меню веб-приложения;

– Описан алгоритм решения задачи проектирования;

– Спроектирован и описан серверный интерфейс взаимодействия с клиентской подсистемой и СУБД PostgreSQL;

– Разработана информационная модель, по которой создано информационное обеспечение веб-приложения, а именно база данных пользователей, постов, комментариев к постам, директов, лайков;

– Разработано программное обеспечение для решения задачи проектирования, тестирование которого было произведено с помощью тестовых данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. Э. Мэйволд. Безопасность сетей. — 2006. — 528 с. — [ISBN 978-5-9570-0046-9](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785957000469)
  2. LoftSchool: Методы верстки сайтов: сайт. – Москва, 2022 –.– URL: <https://loftschool.com/blog/posts/verstka-sajta-chto-eto#verstka4> (Дата обращения: 15.03.2023).
  3. 4 типа архитектуры программного обеспечения | by Daria Sidorova | NOP::Nuances of Programming | Medium : сайт. – Хельсинки, 2021 – . – URL: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724 (дата обращения: 27.02.22).
  4. Что такое SQL? Чем отличаются MySql и PostgreSQL?. – Москва, 2019 – . – URL: https://webhost1.ru/information/blog/2021-05-24-sql-osnovnye-sql-subd.html (дата обращения: 29.03.2023).
  5. Гвоздева, Т.В. Проектирование информационных систем. Стандартизация [Электронный ресурс] : учебное пособие / Т.В. Гвоздева, Б.А. Баллод. — Электрон. дан. — Санкт-Петербург : Лань, 2019. — 252 с. — Режим доступа: https://e.lanbook.com/book/115515. — Загл. с экрана.
  6. Архитектурные решения информационных систем [Электронный ресурс] : учебник / А.И. Водяхо [и др.]. — Электрон. дан. — Санкт-Петербург : Лань, 2017. — 356 с. — Режим доступа: https://e.lanbook.com/book/96850. — Загл. с экрана.
  7. Заяц, А.М. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и node.js [Электронный ресурс] : учебное пособие / А.М. Заяц, Н.П. Васильев. — Электрон. дан. — Санкт-Петербург : Лань, 2019. — 120 с.
  8. Старолетов, С.М. Основы тестирования и верификации программного обеспечения [Электронный ресурс] : учебное пособие / С.М. Старолетов. — Электрон. дан. — Санкт-Петербург : Лань, 2018. — 344 с.