МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЁТ

по лабораторной работе 1

по дисциплине

Защита информации

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Симанов В.С.

(подпись) (фамилия, и.,о.)

21-ИС\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2024

Цель: Реализовать классические криптосистемы (Шифры замены)

Задания:

Задание №1.2

Реализовать алгоритм шифрования данных «Биграммный шифр Плейфейра».

«Биграммный шифр Плейфейра»:

Шифр Плейфера использует матрицу 5х5 (для латинского алфавита, для кириллического алфавита необходимо увеличить размер матрицы до 4х8), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки матрицы буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо, либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом, составляет матрицу 5х5 и является ключом шифра[6][7].

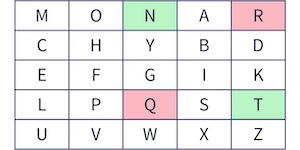
Для того чтобы зашифровать сообщение, необходимо разбить его на биграммы (группы из двух символов), например «Hello World» становится «HE LL OW OR LD», и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга. Затем, руководствуясь следующими 4 правилами[6], зашифровываем пары символов исходного текста:

Если два символа биграммы совпадают (или если остался один символ), добавляем после первого символа «Х», зашифровываем новую пару символов и продолжаем. В некоторых вариантах шифра Плейфера вместо «Х» используется «Q».

Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящиеся непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

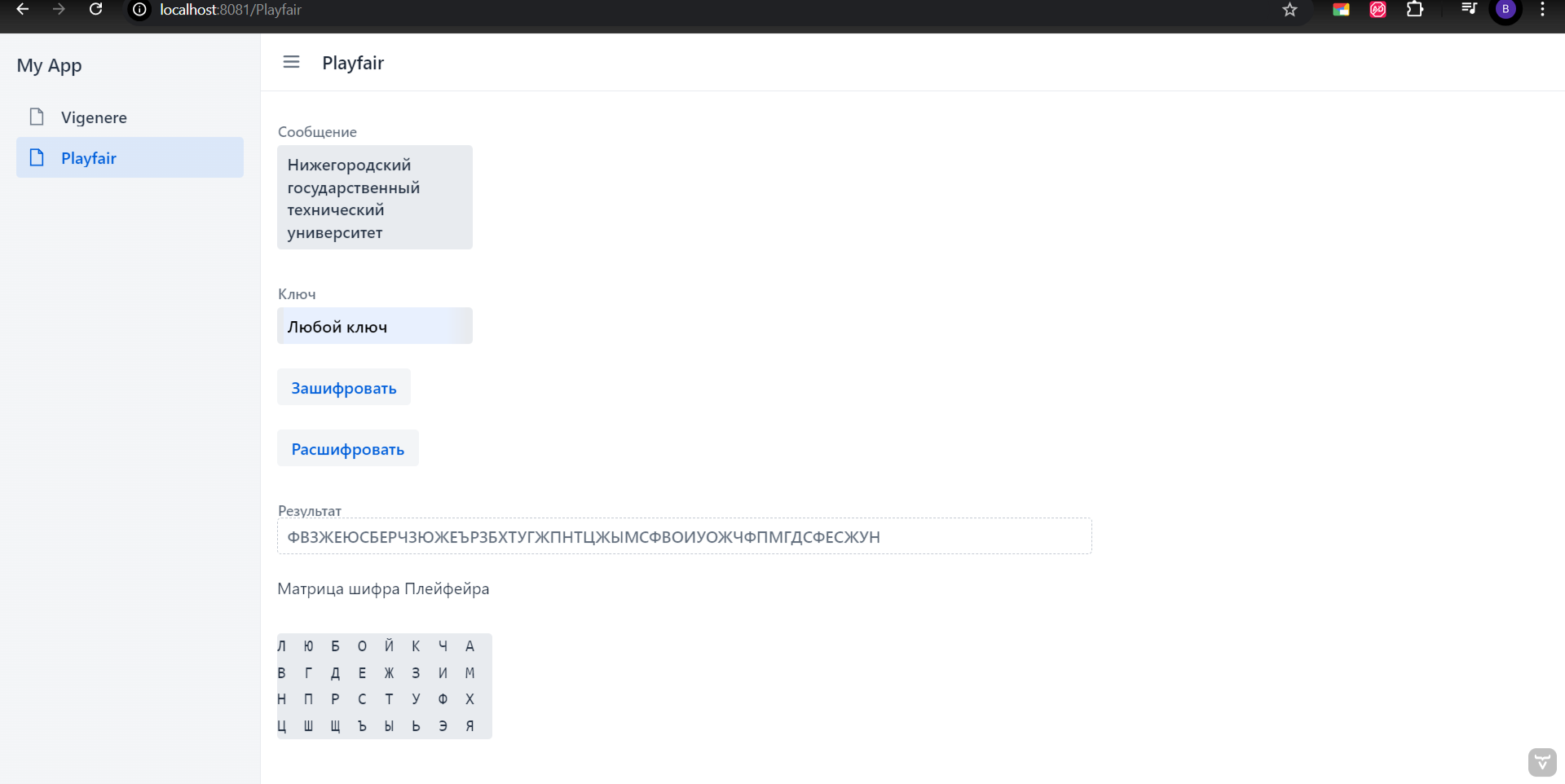
Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.



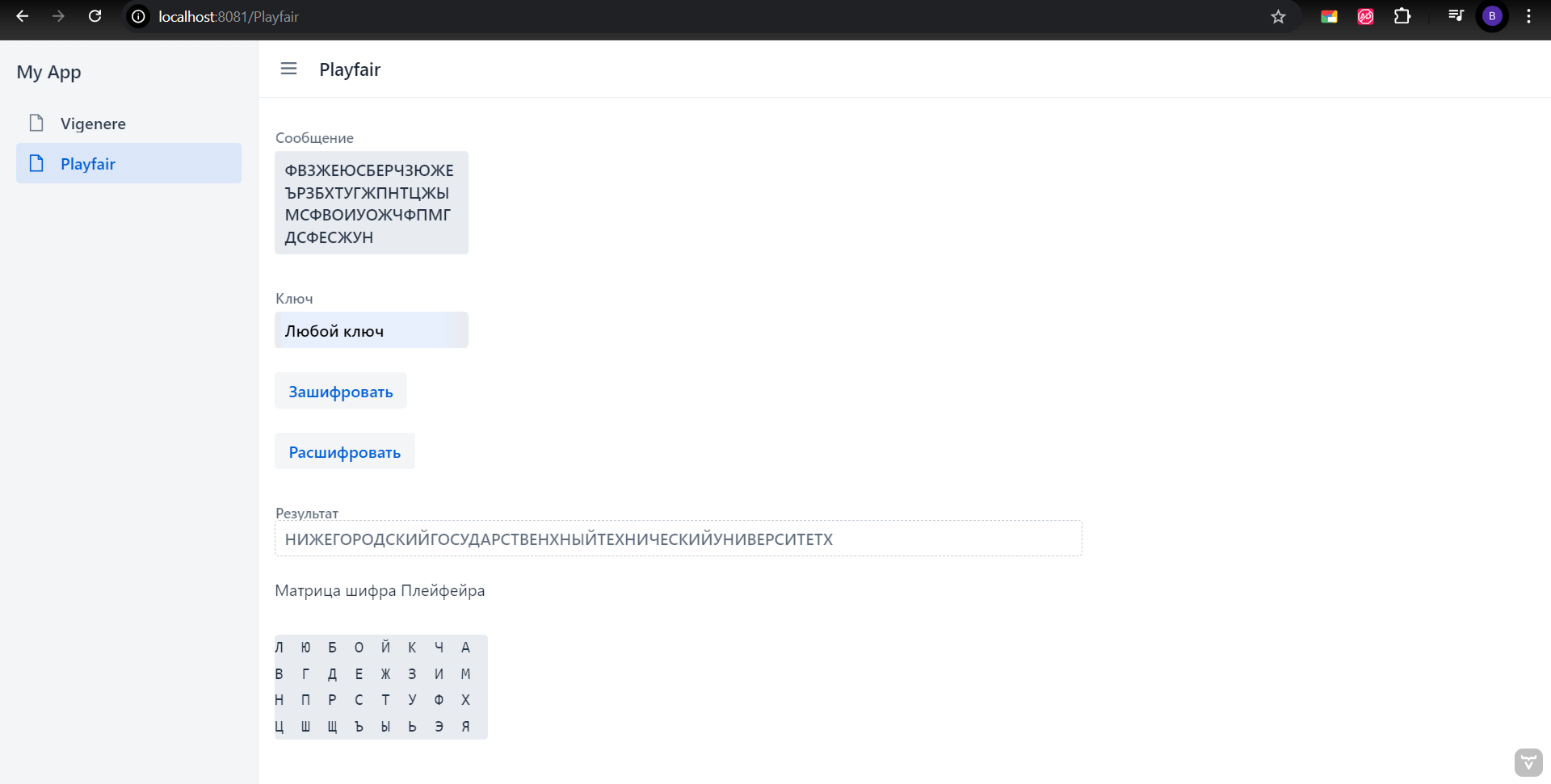
Примеры шифрования и расшифровывания:

Нижегородский государственный технический университет

Шифрование:

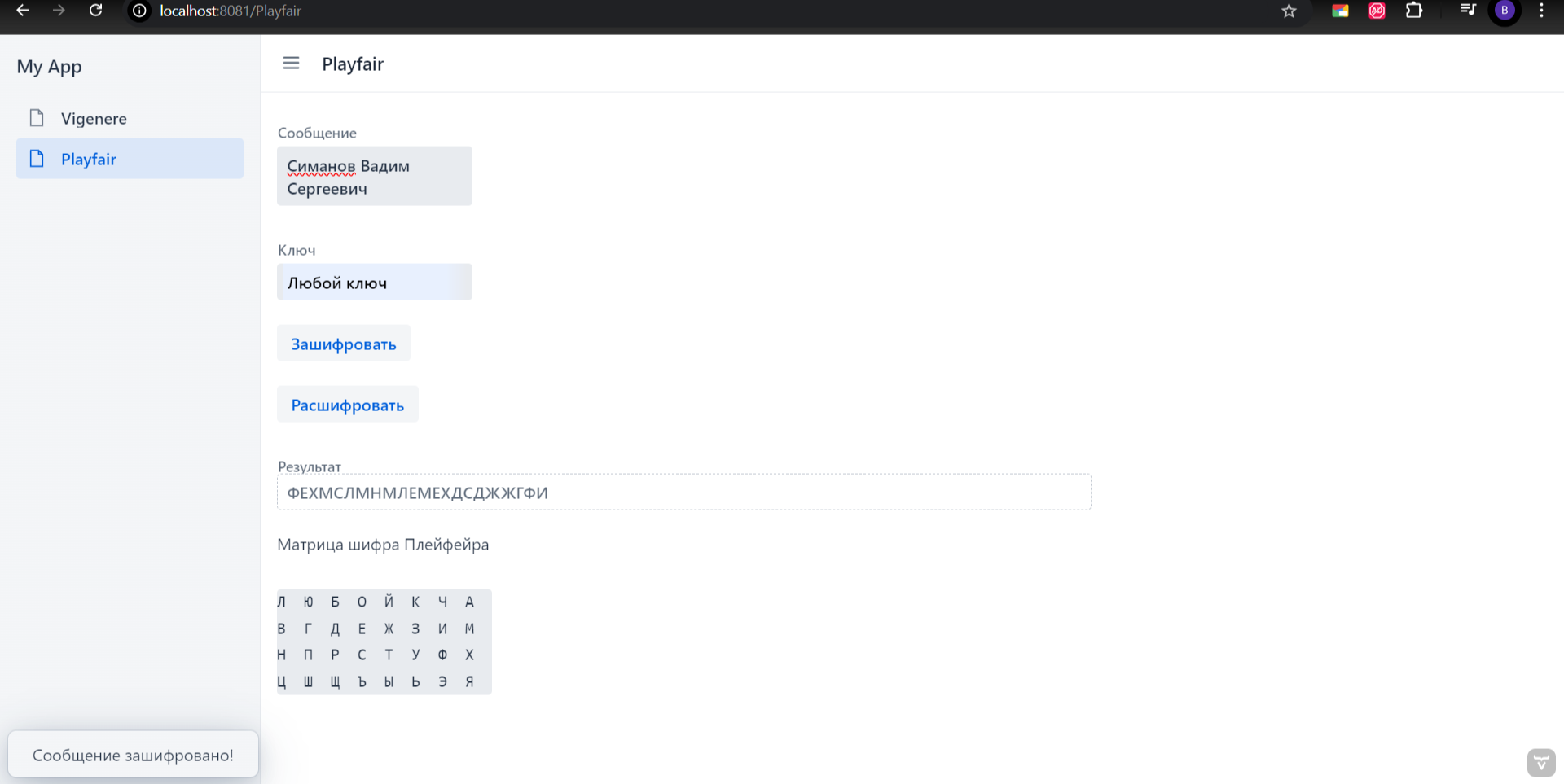


Нижегородский государственный технический университет:

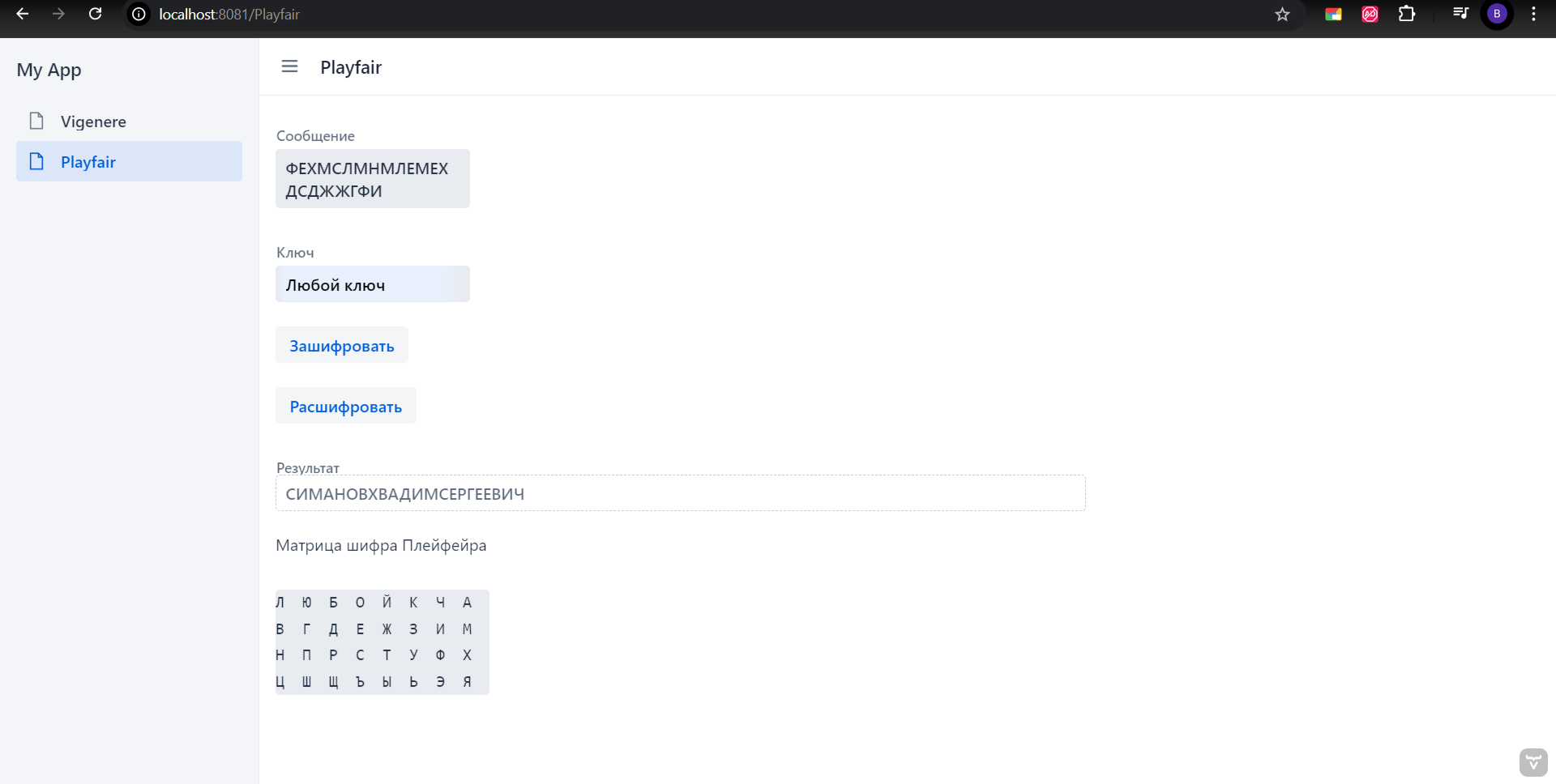


Симанов Вадим Сергеевич:

Шифрование:



Расшифрование:



Текст программы с комментариями:

PlayfairView – отвечает за отображении страницы

package com.example.application.views.lb;  
  
import com.vaadin.flow.component.html.Span;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextArea;  
import com.vaadin.flow.router.Menu;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.textfield.TextField;  
  
// Устанавливает заголовок страницы браузера  
@PageTitle("Playfair")  
// Добавляет элемент в меню с иконкой и указывает порядок отображения  
@Menu(icon = "line-awesome/svg/file.svg", order = 0)  
// Определяет маршрут для отображения этой страницы в браузере  
@Route(value = "Playfair")  
public class PlayfairView extends VerticalLayout {  
  
 // Конструктор класса, который инициализирует элементы пользовательского интерфейса  
 public PlayfairView() {  
 // Поле для ввода ключа шифрования  
 TextField keyText = new TextField("Ключ");  
  
 // Поле для ввода исходного сообщения  
 TextArea text = new TextArea("Сообщение");  
  
 // Поле для вывода зашифрованного или расшифрованного результата (только для чтения)  
 TextField result = new TextField("Результат");  
  
 // Метка для описания матрицы шифра Плейфейра  
 Span matrix = new Span("Матрица шифра Плейфейра");  
  
 // Устанавливаем ширину и высоту для поля вывода результата  
 result.setWidth("800px");  
 result.setHeight("50px");  
 result.setReadOnly(true); // Поле только для чтения, нельзя редактировать вручную  
  
 // Элемент для отображения матрицы шифра (будет заполняться позже)  
 Span matrixDisplay = new Span();  
  
 // Кнопка "Зашифровать" и обработчик события нажатия  
 Button encryptButton = new Button("Зашифровать", e -> {  
 // Получаем значения ключа и сообщения  
 String key = keyText.getValue();  
 String inputText = text.getValue();  
  
 // Проверяем, что оба поля не пустые  
 if (!key.isEmpty() && !inputText.isEmpty()) {  
 // Создаем экземпляр шифра Плейфейра с заданным ключом  
 cipherPlayfair cipher = new cipherPlayfair(key);  
  
 // Шифруем сообщение  
 String encryptedText = cipher.encrypt(inputText);  
  
 // Устанавливаем зашифрованный текст в поле результата  
 result.setValue(encryptedText);  
  
 // Показываем уведомление о завершении шифрования  
 Notification.*show*("Сообщение зашифровано!");  
  
 // Отображаем матрицу шифра в виде HTML  
 matrixDisplay.getElement().setProperty("innerHTML", cipher.getMatrixAsHtml());  
 } else {  
 // Показываем уведомление, если поля не заполнены  
 Notification.*show*("Пожалуйста, заполните все поля.");  
 }  
 });  
  
 // Кнопка "Расшифровать" и обработчик события нажатия  
 Button decryptButton = new Button("Расшифровать", e -> {  
 // Получаем значения ключа и зашифрованного сообщения  
 String key = keyText.getValue();  
 String inputText = text.getValue();  
  
 // Проверяем, что оба поля не пустые  
 if (!key.isEmpty() && !inputText.isEmpty()) {  
 // Создаем экземпляр шифра Плейфейра с заданным ключом  
 cipherPlayfair cipher = new cipherPlayfair(key);  
  
 // Расшифровываем сообщение  
 String decryptedText = cipher.decrypt(inputText);  
  
 // Устанавливаем расшифрованный текст в поле результата  
 result.setValue(decryptedText);  
  
 // Показываем уведомление о завершении расшифровки  
 Notification.*show*("Сообщение расшифровано!");  
  
 // Отображаем матрицу шифра в виде HTML  
 matrixDisplay.getElement().setProperty("innerHTML", cipher.getMatrixAsHtml());  
 } else {  
 // Показываем уведомление, если поля не заполнены  
 Notification.*show*("Пожалуйста, заполните все поля.");  
 }  
 });  
  
 // Добавляем элементы на страницу  
 add(text, keyText, encryptButton, decryptButton, result, matrix, matrixDisplay);  
 }  
}

cipherPlayfair – метод реализующий алгоритм шифрования данных «Биграммный шифр Плейфейра».

package com.example.application.views.lb;  
  
import java.util.\*;  
  
public class cipherPlayfair {  
  
 // Матрица для шифрования (4 строки на 8 столбцов)  
 private char[][] matrix;  
  
 // Ключ, использующийся для генерации матрицы  
 private String key;  
  
 // Русский алфавит без буквы "Ё"  
 private final String alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";  
  
 // Конструктор класса, инициализирует ключ и создает матрицу шифра  
 public cipherPlayfair(String key) {  
 this.key = prepareKey(key); // Подготовка ключа (удаление дубликатов)  
 matrix = new char[4][8]; // Создаем матрицу 4x8  
 createMatrix(); // Заполнение матрицы  
 }  
  
 // Подготовка ключа: убираем дубликаты символов и исключаем лишние символы  
 private String prepareKey(String key) {  
 key = key.toUpperCase().replaceAll("[^А-Я]", "").replace("Ё", "Е"); // Приведение к верхнему регистру и удаление лишних символов  
 StringBuilder sb = new StringBuilder(); // Строим строку без дубликатов  
 Set<Character> usedChars = new HashSet<>(); // Для хранения уже использованных символов  
 for (char c : key.toCharArray()) {  
 if (!usedChars.contains(c)) { // Добавляем только уникальные символы  
 usedChars.add(c);  
 sb.append(c);  
 }  
 }  
 return sb.toString(); // Возвращаем подготовленный ключ  
 }  
  
 // Создание матрицы шифра 4x8, используя ключ и оставшиеся буквы алфавита  
 private void createMatrix() {  
 Set<Character> usedChars = new HashSet<>(); // Для отслеживания использованных символов  
 int index = 0; // Индекс для заполнения матрицы  
  
 // Заполняем матрицу символами из ключа  
 for (char c : key.toCharArray()) {  
 if (!usedChars.contains(c)) {  
 usedChars.add(c);  
 matrix[index / 8][index % 8] = c; // Заполняем строку и столбец  
 index++;  
 }  
 }  
  
 // Заполняем оставшиеся ячейки матрицы символами алфавита  
 for (char c : alphabet.toCharArray()) {  
 if (!usedChars.contains(c)) {  
 matrix[index / 8][index % 8] = c;  
 index++;  
 }  
 }  
 }  
  
 // Подготовка текста: разбиваем на биграммы, заменяем недостающие символы на 'Х'  
 private List<String> prepareText(String text) {  
 text = text.toUpperCase().replaceAll("[^А-Я]", "").replace("Ё", "Е"); // Приводим текст к нужному формату  
 List<String> bigrams = new ArrayList<>();  
  
 // Формируем биграммы по две буквы  
 for (int i = 0; i < text.length(); i += 2) {  
 char first = text.charAt(i);  
 char second = (i + 1 < text.length()) ? text.charAt(i + 1) : 'Х'; // Если текст неполный, добавляем 'Х'  
  
 if (first == second) {  
 second = 'Х'; // Если два одинаковых символа подряд, второй заменяется на 'Х'  
 i--; // Возвращаем индекс для повторной обработки второго символа  
 }  
  
 bigrams.add("" + first + second); // Добавляем биграмму  
 }  
 return bigrams; // Возвращаем список биграмм  
 }  
  
 // Получение координат символа в матрице  
 private int[] getCharPos(char c) {  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 8; j++) {  
 if (matrix[i][j] == c) {  
 return new int[]{i, j}; // Возвращаем координаты символа  
 }  
 }  
 }  
 return null; // Символ не найден (но такого не должно быть, если текст правильно подготовлен)  
 }  
  
 // Метод для шифрования текста  
 public String encrypt(String text) {  
 List<String> bigrams = prepareText(text); // Подготовка текста к шифрованию (биграммы)  
 StringBuilder encryptedText = new StringBuilder();  
  
 // Шифруем каждую биграмму  
 for (String bigram : bigrams) {  
 encryptedText.append(encryptBigram(bigram));  
 }  
 return encryptedText.toString(); // Возвращаем зашифрованный текст  
 }  
  
 // Шифрование одной биграммы  
 private String encryptBigram(String bigram) {  
 char first = bigram.charAt(0);  
 char second = bigram.charAt(1);  
 int[] pos1 = getCharPos(first); // Получаем координаты первого символа  
 int[] pos2 = getCharPos(second); // Получаем координаты второго символа  
  
 if (pos1[0] == pos2[0]) {  
 // Если символы находятся в одной строке, сдвигаем их вправо  
 return "" + matrix[pos1[0]][(pos1[1] + 1) % 8] + matrix[pos2[0]][(pos2[1] + 1) % 8];  
 } else if (pos1[1] == pos2[1]) {  
 // Если символы находятся в одном столбце, сдвигаем их вниз  
 return "" + matrix[(pos1[0] + 1) % 4][pos1[1]] + matrix[(pos2[0] + 1) % 4][pos2[1]];  
 } else {  
 // Если символы образуют прямоугольник, заменяем их на противоположные углы  
 return "" + matrix[pos1[0]][pos2[1]] + matrix[pos2[0]][pos1[1]];  
 }  
 }  
  
 // Метод для расшифровки текста  
 public String decrypt(String text) {  
 List<String> bigrams = prepareText(text); // Подготовка текста (формирование биграмм)  
 StringBuilder decryptedText = new StringBuilder();  
  
 // Расшифровываем каждую биграмму  
 for (String bigram : bigrams) {  
 decryptedText.append(decryptBigram(bigram));  
 }  
 return decryptedText.toString(); // Возвращаем расшифрованный текст  
 }  
  
 // Расшифровка одной биграммы  
 private String decryptBigram(String bigram) {  
 char first = bigram.charAt(0);  
 char second = bigram.charAt(1);  
 int[] pos1 = getCharPos(first); // Получаем координаты первого символа  
 int[] pos2 = getCharPos(second); // Получаем координаты второго символа  
  
 if (pos1[0] == pos2[0]) {  
 // Если символы находятся в одной строке, сдвигаем их влево  
 return "" + matrix[pos1[0]][(pos1[1] + 7) % 8] + matrix[pos2[0]][(pos2[1] + 7) % 8];  
 } else if (pos1[1] == pos2[1]) {  
 // Если символы находятся в одном столбце, сдвигаем их вверх  
 return "" + matrix[(pos1[0] + 3) % 4][pos1[1]] + matrix[(pos2[0] + 3) % 4][pos2[1]];  
 } else {  
 // Если символы образуют прямоугольник, заменяем их на противоположные углы  
 return "" + matrix[pos1[0]][pos2[1]] + matrix[pos2[0]][pos1[1]];  
 }  
 }  
  
 // Метод для отображения матрицы в виде HTML  
 public String getMatrixAsHtml() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("<pre>"); // Используем тег <pre> для форматированного вывода  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 8; j++) {  
 sb.append(matrix[i][j]).append(" "); // Добавляем символы матрицы с пробелами  
 }  
 sb.append("\n"); // Переход на новую строку  
 }  
 sb.append("</pre>");  
 return sb.toString(); // Возвращаем HTML-код с матрицей  
 }  
}

Application – запуск программы

package com.example.application;  
  
import com.vaadin.flow.component.page.AppShellConfigurator;  
import com.vaadin.flow.theme.Theme;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
@Theme(value = "my-app")  
public class Application implements AppShellConfigurator {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}

Вывод: мною был реализован алгоритм шифрования данных «Биграммный шифр Плейфейра».