Додаток А
Графічні матеріали

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

«ЗАТВЕРДЖУЮ»

Керівник кваліфікаційної роботи

_____ проф. Нечипоренко А.С.
(підпис)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ КОМПОНЕНТІВ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Графічні матеріали

АРКУШ ЗАТВЕРДЖЕННЯ

ГЮИК.506000.003 С10

РОЗРОБИЛА

ст. гр. ІТПм-20-1

Жукова А.В.

2021

ЗАТВЕРДЖЕНО
ГЮИК.506000.003 С10

РОЗРОБКА ТА ДОСЛІДЖЕННЯ КОМПОНЕНТІВ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Графічні матеріали

ГЮИК.506000.003

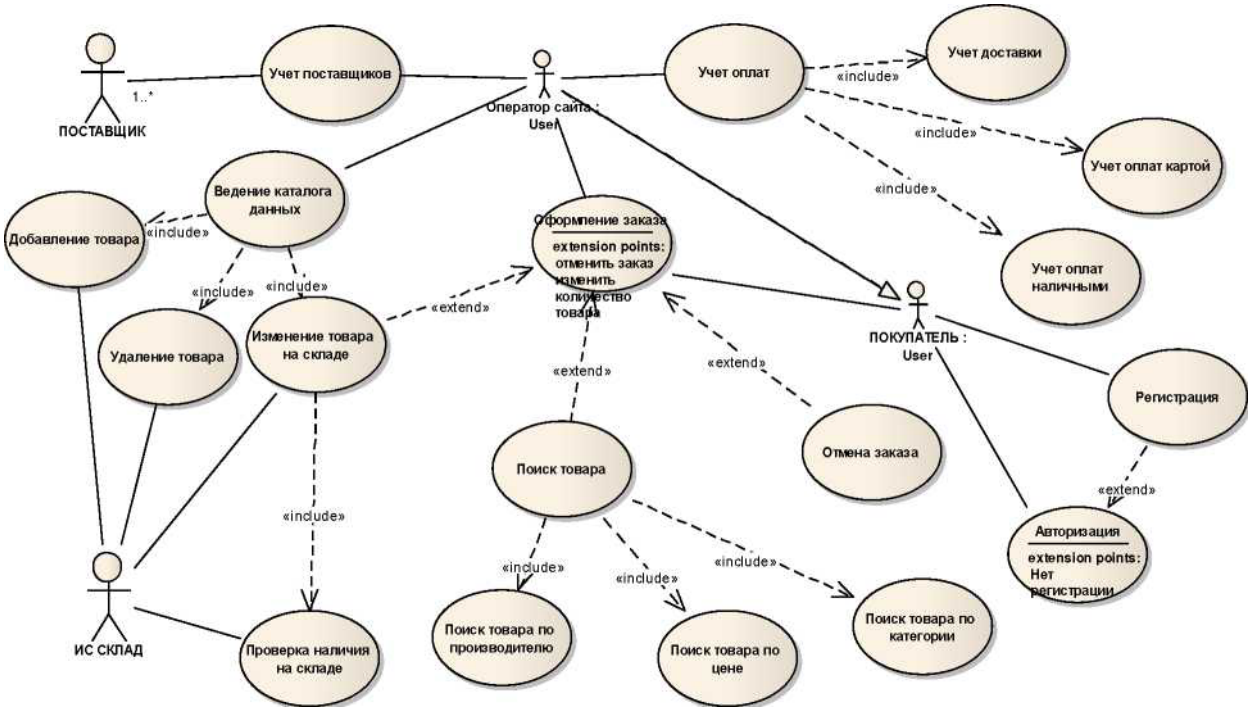
Аркушів 6

2021

ЗМІСТ

ГЮИК.506000.003 С10

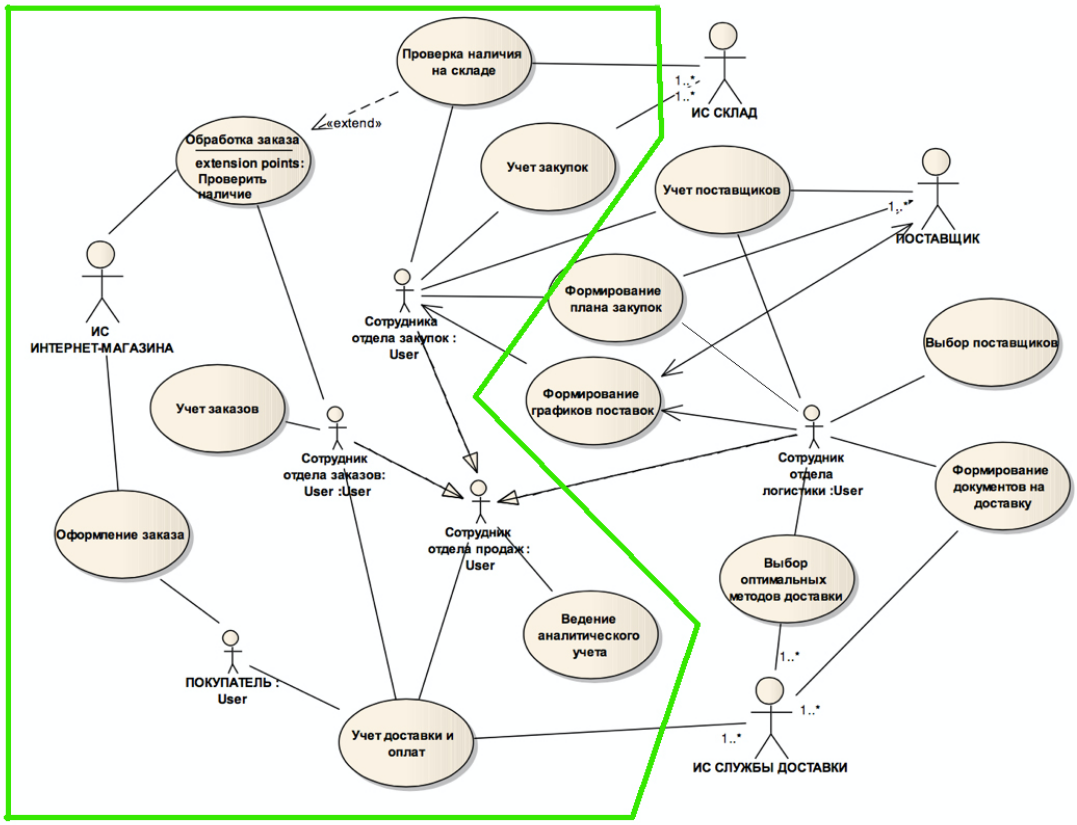
ФУНКЦІОНАЛЬНА СТРУКТУРА ІНТЕРНЕТ - МАГАЗИНУ БЕЗ ВИКОРИСТАННЯ СИСТЕМИ ЛОГІСТИКИ



					ГЮИК.506000.003 С10			
					Функціональна структура інтернет - магазину без використання системи логістики	Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Жукова А.В.						
Провер.		Нечипоренко А.С.						
Т. Контр.						Лист 1	Листов 1	
Реценз.						ХНУРЭ Кафедра СТ		

ГЮИК.506000.003 С10

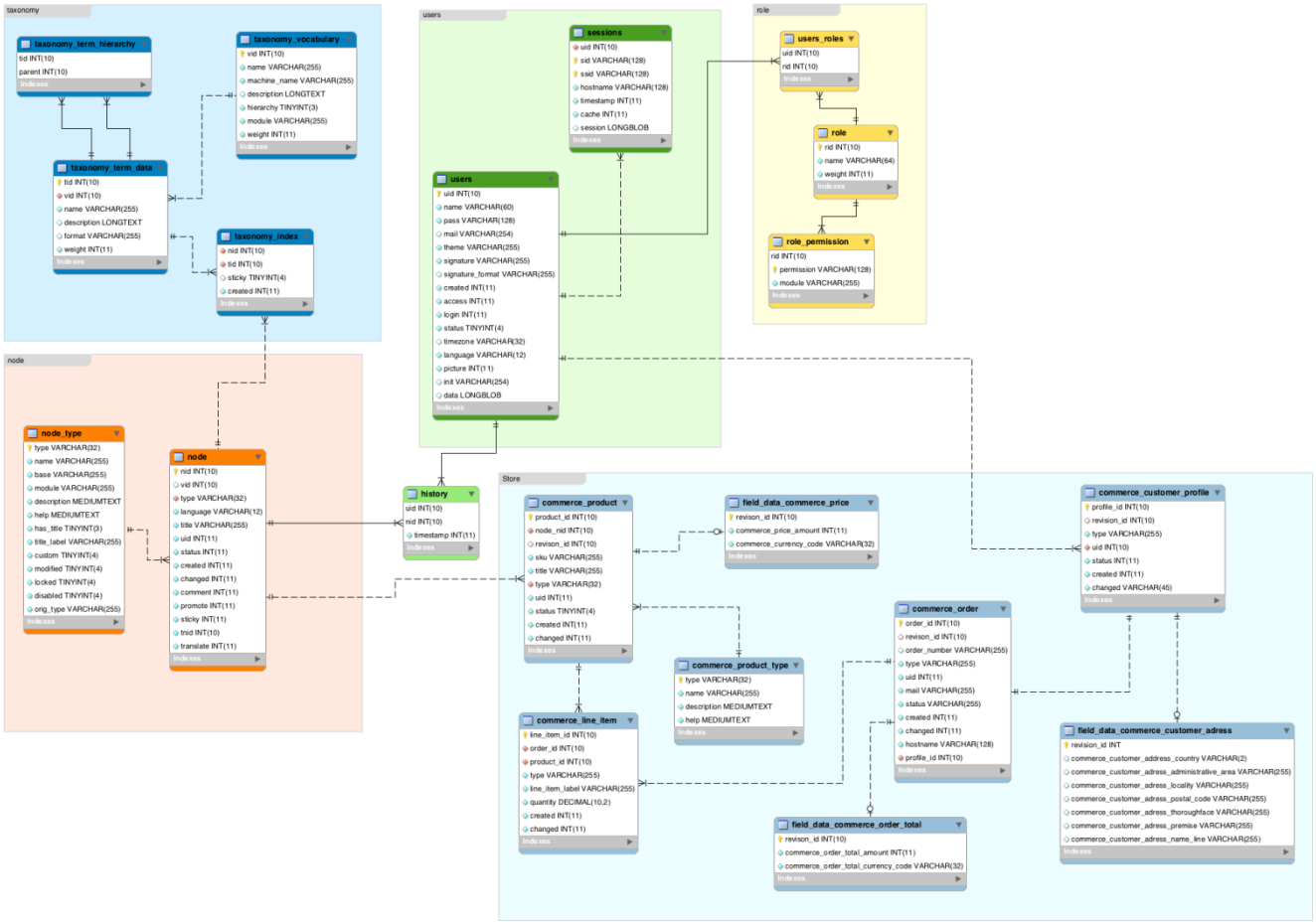
ФУНКЦІОНАЛЬНА ЛОГІСТИЧНА СИСТЕМА ІНТЕРНЕТ - МАГАЗИНУ У ВИГЛЯДІ USE-CASE ДІАГРАМИ



					ГЮИК.506000.003 С10	
Изм.	Лист	№ докум.	Подпись	Дата	Функціональна логістична система інтернет - магазину у вигляді use-case діаграми	Лит. / Масса / Масштаб
Разраб.		Жукова А.В.				
Провер.		Нечипоренко А.С.				Лист 1 / Листов 1
Т. Контр.						
Реценз.						ХНУРЭ Кафедра СТ

ГЮИК.506000.003 С10

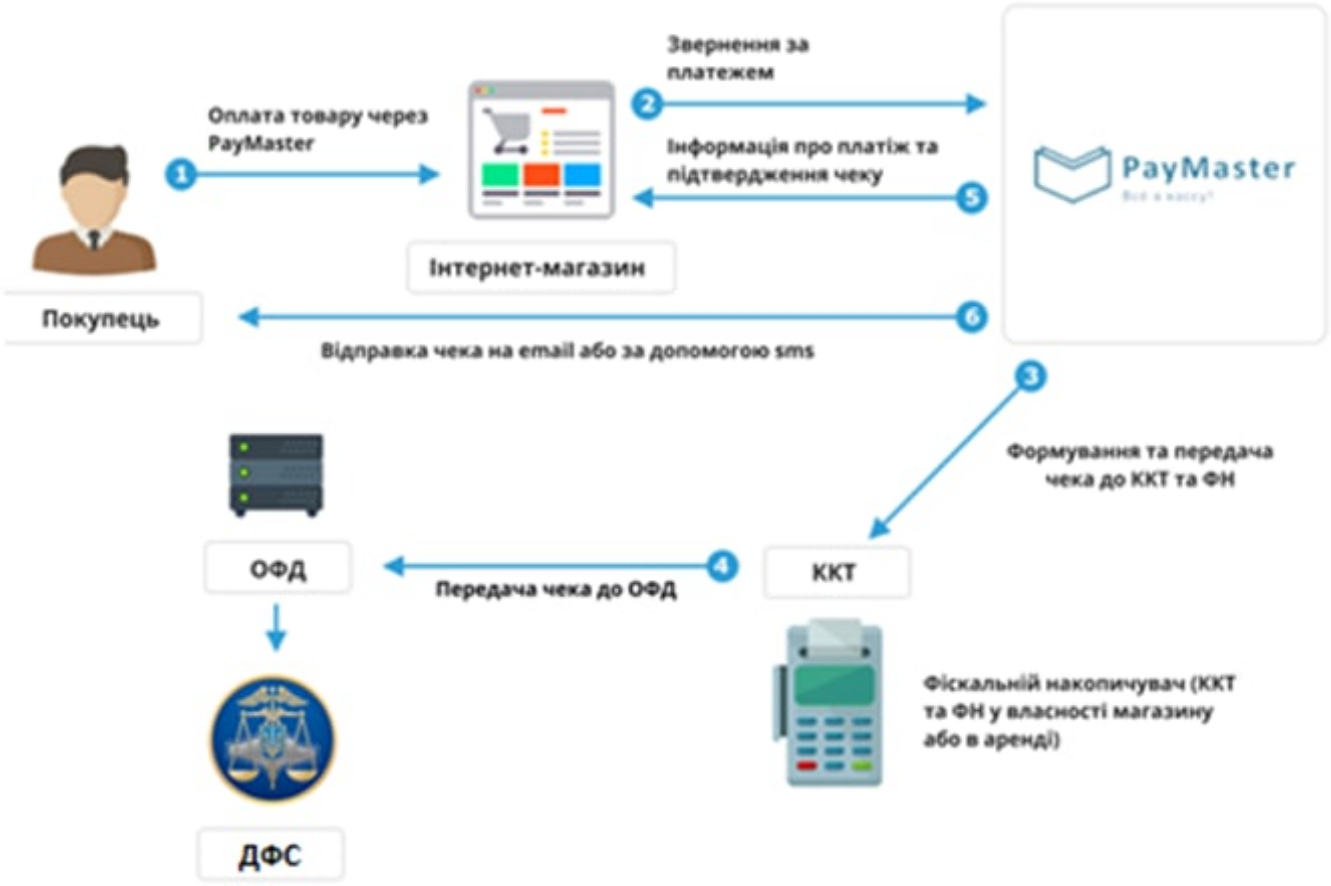
ФІЗИЧНА МОДЕЛЬ ДАНИХ СИСТЕМИ «ІНТЕРНЕТ - МАГАЗИН» З ОГЛЯДУ НА ОСОБЛИВОСТІ ІНСТРУМЕНТУ РОЗРОБКИ



					ГЮИК.506000.003 С10			
					Фізична модель даних системи «Інтернет - магазин» з огляду на особливості інструменту розробки	Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Жукова А.В.						
Провер.		Нечипоренко А.С.						
Т. Контр.						Лист 1		Листов 1
Реценз.						ХНУРЭ Кафедра СТ		

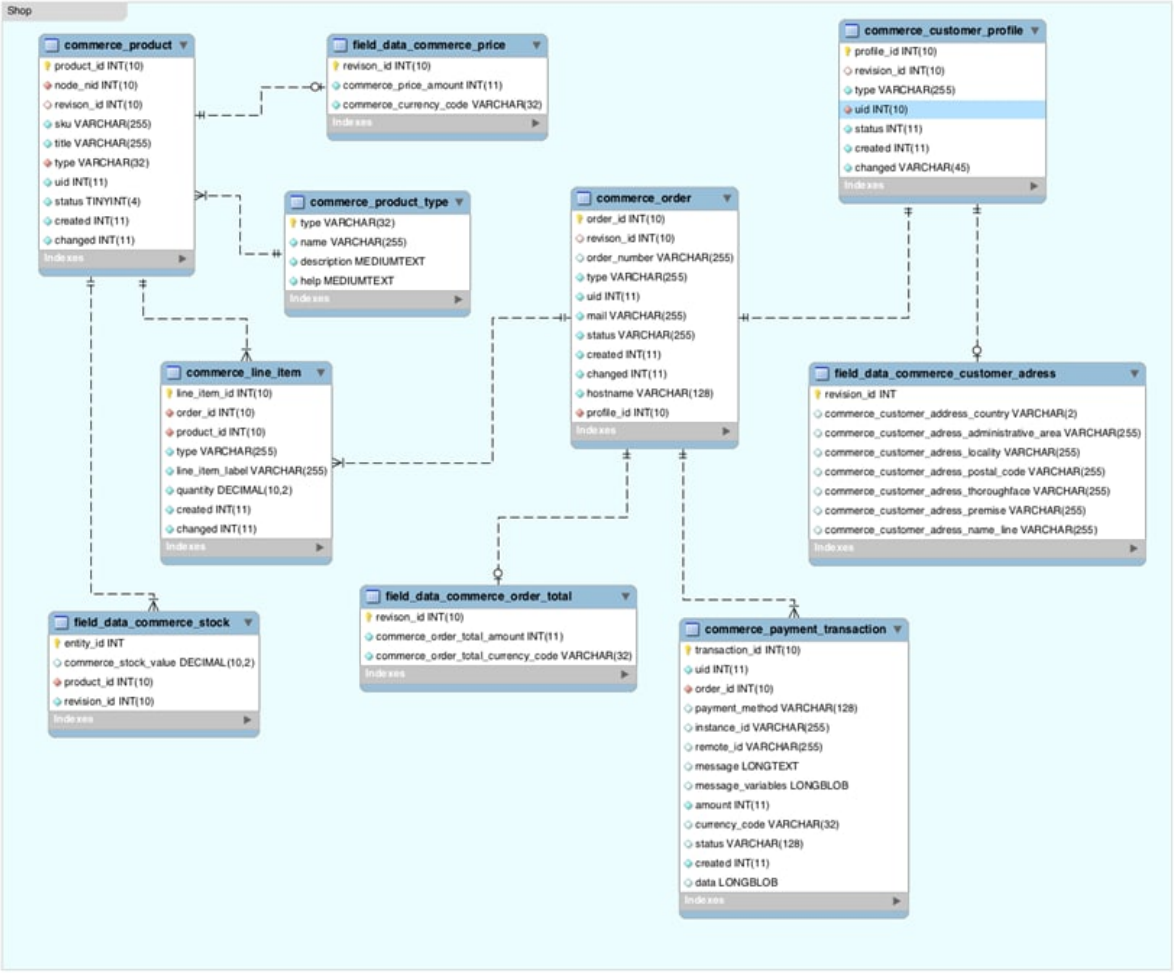
ГЮИК.506000.003 С10

СХЕМА РОБОТИ ОНЛАЙН-КАСИ ЧЕРЕЗ PAYMASTER



					ГЮИК.506000.003 С10			
					Схема роботи онлайн-каси через paymaster	Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Жукова А.В.						
Провер.		Нечипоренко А.С.				Лист 1	Листов 1	
Т. Контр.								
Реценз.						ХНУРЭ Кафедра СТ		

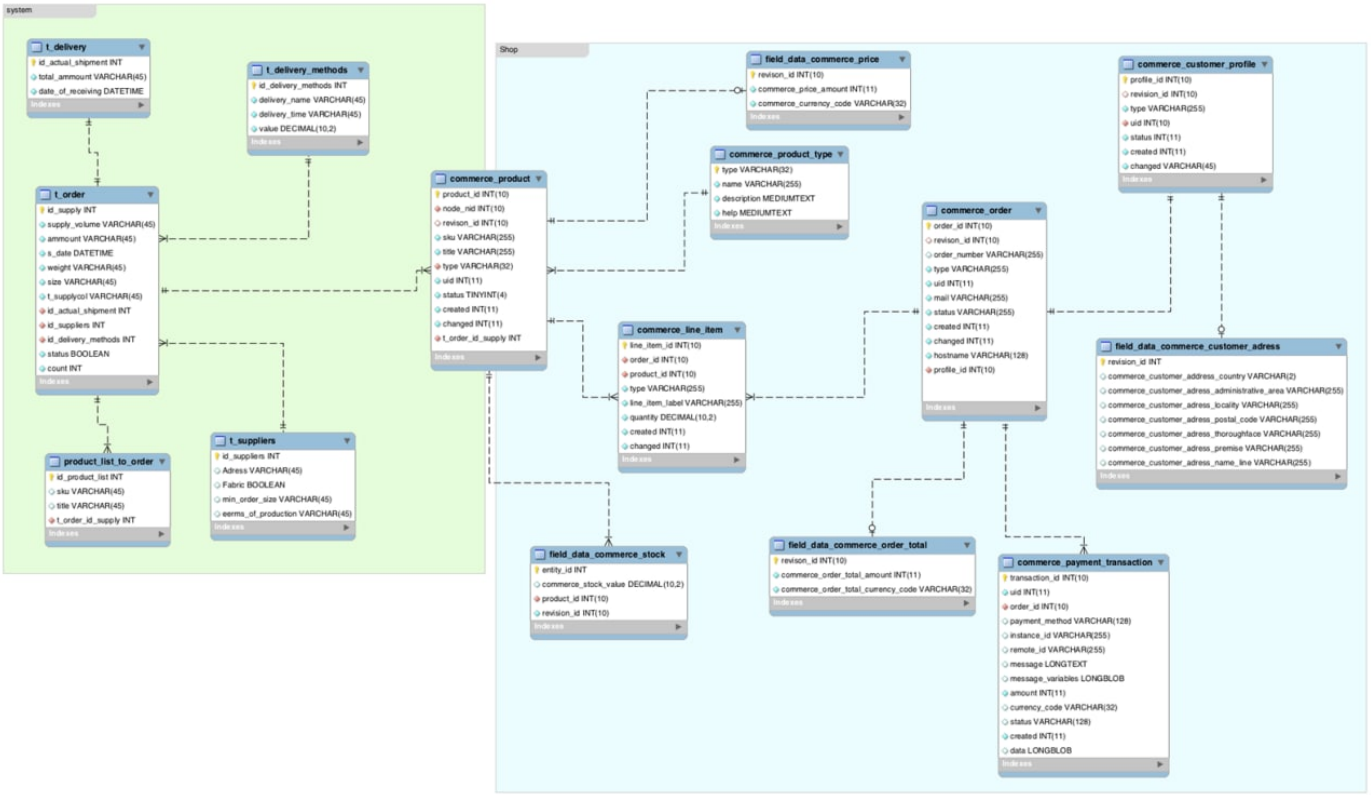
ФІЗИЧНА МОДЕЛЬ ДАНИХ ПІСЛЯ ВПРОВАДЖЕННЯ СТАТИСТКИ ОБЛІКУ ТА ОНЛАЙН ОПЛАТИ



					ГЮИК.506000.003 С10	
Изм.	Лист	№ докум.	Подпись	Дата	Фізична модель даних після впровадження статистки обліку та онлайн оплати	Лит. / Масса / Масштаб
Разраб.		Жукова А.В.				
Провер.		Нечипоренко А.С.				Лист 1 / Листов 1
Т. Контр.						
Реценз.						ХНУРЭ Кафедра СТ

ГЮИК.506000.003 С10

ФІЗИЧНА МОДЕЛЬ ДАНИХ СИСТЕМИ АНАЛІТИКИ



					ГЮИК.506000.003 С10	
					Фізична модель даних системи аналітики	Лит. / Масса / Масштаб
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Жукова А.В.				Лист 1 / Листов 1
Провер.		Нечипоренко А.С.				
Т. Контр.						ХНУРЭ Кафедра СТ
Реценз.						

Додаток Б

Текст програми

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

«ЗАТВЕРДЖУЮ»
Керівник кваліфікаційної роботи
_____ проф. Нечипоренко А.С.
(підпис)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ КОМПОНЕНТІВ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Текст програми

АРКУШ ЗАТВЕРДЖЕННЯ
ГЮИК.506000.003-01 12 01-ЛУ

РОЗРОБИЛА
ст. гр. ІТПм-20-1
Жукова А.В.

2021

ЗАТВЕРДЖЕНО
ГЮИК.506000.003-01 12 01-ЛУ

РОЗРОБКА ТА ДОСЛІДЖЕННЯ КОМПОНЕНТІВ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Текст програми

ГЮИК.506000.003-01 12 01

Аркушів 12

2021

```
namespace matrixGame
{
    public partial class Form1 : Form
    {
        private int[,] paymentMatrix;
        private int m, n;
        private int[,] simpleMatrix;
        private int coefPositive;
        private double[] p;
        private double[] q;
        private double V;
        //Средняя цена товара
        private int[] averagePrice;

        //Продажи товара в день
        private double sellsForDay = 200;

        public Form1()
        {
         averagePrice = new int[5];
         averagePrice[0] = 1000; averagePrice[1] = 5500; averagePrice[2] =
12500; averagePrice[3] = 25000;
averagePrice[4] = 55000;
            paymentMatrix = new int[6, 7];
            p = new double[5];
            q = new double[6];
            InitializeComponent();
            basicTable.Rows.Add("Д0", 5, 7, 1);
            basicTable.Rows.Add("Д1", 27, 14, 7);
            basicTable.Rows.Add("Д2", 62, 90, 14);
            basicTable.Rows.Add("Д3", 125, 90, 31);
            basicTable.Rows.Add("Д4", 275, 100, 62);
            tbSells.Text = sellsForDay.ToString();
        }

        private void btnCount_Click(object sender, EventArgs e)
        {
         for (int i = 0; i < 5; ++i)
            p[i] = 0;
        for (int i = 0; i < 6; ++i)
            q[i] = 0;
        int rowsCount = paymentMatrixTable.Rows.Count;
        for (int i = 0; i < rowsCount; i++)
        {
        paymentMatrixTable.Rows.Remove(paymentMatrixTable.Rows[0]);
        }
        Double.TryParse(tbSells.Text.ToString(), out sellsForDay);

           for (int i = 0; i < 5; ++i)
                basicTable.Rows[i].Cells[1].Value = ((int) (averagePrice[i] /
sellsForDay)).ToString();
```

```
            constructPayMatrix();
            simplifyPaymentMatrix();
            positive();
            solve();
            for (int i = 0; i < 5; ++i)
            {
            if (Double.IsNaN(p[i]) || Double.IsInfinity(p[i]))
               p[i] = 1;
                else p[i] = Math.Round(p[i], 2,
MidpointRounding.AwayFromZero);
            }
            for (int i = 0; i < 6; ++i)
            {
                if (Double.IsNaN(q[i]) || Double.IsInfinity(q[i]))
                    q[i] = 1;
                else q[i] = Math.Round(q[i], 2,
MidpointRounding.AwayFromZero);
            }
            rowsCount = pTable.Rows.Count;
            for (int i = 0; i < rowsCount; i++)
            {
                pTable.Rows.Remove(pTable.Rows[0]);
                qTable.Rows.Remove(qTable.Rows[0]);
            }
            pTable.Rows.Add(p[0],p[1],p[2],p[3],p[4]);
            qTable.Rows.Add(q[0], q[1], q[2], q[3], q[4], q[5]);
            V = Math.Round(V, 2, MidpointRounding.AwayFromZero);
            tbValue.Text = V.ToString();
            Vald();
            Savidge();
            Hurwitz();
        }

        private void constructPayMatrix()
        {
            int temp;
            //Текущий курс доллара
            int currentExchangeRates = 60;
            //Возможный рост курса доллара
            int prognosedExchangeRates = 65;
            //Количество необходимых закупок в год
            double[] cd = new double[5];
            //Потери при задержке поставки
            double[] loss = new double[5];
            //Потери при возрастании курса
            double[] exchangeRateLoss = new double[5];
            //Задержка в производстве товара
            int[] productionDelay = new int[5];
            for (int i = 0; i < 5; ++i)
            {
                Int32.TryParse(basicTable.Rows[i].Cells[1].Value.ToString(),
out temp);
```

```
                cd[i] = Math.Round(365.0 / temp, 2,
MidpointRounding.AwayFromZero);

                loss[i] = temp * 0.15;
                if (loss[i] < 7)
                    loss[i] = 0;

                Int32.TryParse(basicTable.Rows[i].Cells[3].Value.ToString(),
out temp);
                productionDelay[i] = temp;
                if (productionDelay[i] < 3)
                    productionDelay[i] = 0;
                else if (productionDelay[i] > 30)
                    productionDelay[i] = 30;

                exchangeRateLoss[i] = Math.Round((averagePrice[i] / 2.0 *
prognosedExchangeRates - averagePrice[i] / 2 * currentExchangeRates) /
currentExchangeRates, 2, MidpointRounding.AwayFromZero);
                if (i == 0)
                    exchangeRateLoss[i] = 0;
            }

            //Расходы на хранение товара по мнению экспертов - процент от
общей стоимости
            double[] keepingPrice = new double[5];
            keepingPrice[0] = 0.25; keepingPrice[1] = 0.3; keepingPrice[2] =
0.4; keepingPrice[3] = 0.45; keepingPrice[4] = 0.55;
            double[] deliveryCoef = new double[5];
            deliveryCoef[0] = 1; deliveryCoef[1] = 0.55; deliveryCoef[2] =
0.39; deliveryCoef[3] = deliveryCoef[4] = 0.3;
            //Процент накрутки стоимости товара при задержке на таможне
            double[] customsFee = new double[5];
            customsFee[0] = 0; customsFee[1] = 0.005; customsFee[2] = 0.05;
customsFee[3] = 0.01; customsFee[4] = 0.01;
            //Стоимость хранения товара
            double[] keeping = new double[5];
            //Стоимость доставки товара
            double[] deliveryPrice = new double[5];
            for (int i = 0; i < 5; ++i)
            {
                deliveryPrice[i] = (int)(averagePrice[i] * deliveryCoef[i]);
                keeping[i] = (averagePrice[i] + deliveryPrice[i]) *
keepingPrice[i];

                paymentMatrix[i, 0] = (int)(keeping[i] + (deliveryPrice[i] *
cd[i])) * -1;

                paymentMatrix[i, 1] = (int)(loss[i] * cd[i] * sellsForDay +
deliveryPrice[i] * cd[i] + keeping[i] * 0.85) * -1;
                paymentMatrix[i, 2] = (int)(keeping[i] * 1.15 +
(deliveryPrice[i] * cd[i])) * -1;
                paymentMatrix[i, 3] = (int)(keeping[i] + (deliveryPrice[i] *
```

```
cd[i]) + averagePrice[i] * customsFee[i]) * -1;
                paymentMatrix[i, 4] = (int)(keeping[i] + (deliveryPrice[i] *
cd[i]) + productionDelay[i] * sellsForDay) * -1;
                paymentMatrix[i, 5] = (int)(keeping[i] + (deliveryPrice[i] *
cd[i]) + exchangeRateLoss[i]) * -1;
                paymentMatrix[i, 6] = paymentMatrix[i, 0];

                for (int j = 1; j < 5; ++j)
                {
                    if (paymentMatrix[i, j] < paymentMatrix[i, 6])
                        paymentMatrix[i, 6] = paymentMatrix[i, j];
                }
                paymentMatrixTable.Rows.Add(paymentMatrix[i, 0],
paymentMatrix[i, 1], paymentMatrix[i, 2], paymentMatrix[i, 3],
paymentMatrix[i, 4], paymentMatrix[i, 5]);
            }
            for (int i = 0; i < 6; ++i)
            {
                paymentMatrix[5, i] = paymentMatrix[0, i];
                for (int j = 1; j < 5; ++j)
                {
                    if (paymentMatrix[5, i] < paymentMatrix[j, i])
                        paymentMatrix[5, i] = paymentMatrix[j, i];
                }
            }

        }
        private void simplifyPaymentMatrix()
        {
            n = 5; m = 6;
            int[,] temporaryMatrix = new int[n + 1, m + 1];
            for (int i = 0; i < n; ++i)
            {
                for (int j = 0; j < m; ++j)
                    temporaryMatrix[i, j] = paymentMatrix[i, j];
                temporaryMatrix[i, m] = i;
            }
            //Убираем доминируемые строки
            for (int i = 0; i < n; ++i)
            {
                bool dominated;
                if (temporaryMatrix[i, m] == -1)
                    continue;
                for (int j = 0; j < n; ++j)
                {
                    if (temporaryMatrix[j, m] != -1 && j != i)
                    {
                        dominated = true;
                        for (int k = 0; k < m && dominated; ++k)
                        {
                            if (temporaryMatrix[j, k] >= temporaryMatrix[i,
k])
```

```
                                dominated = false;
                        }
                        if (dominated)
                            temporaryMatrix[j, m] = -1;
                    }
                }
            }
            int count = 0;
            for (int i = 0; i < n; ++i)
            {
                if (temporaryMatrix[i, m] == -1)
                    ++count;
            }
            int l = 0; int h = count;
            while (h > 0)
            {
                if (temporaryMatrix[l, m] == -1)
                {
                    for (int j = l; j < n - 1; ++j)
                        for (int k = 0; k < m + 1; ++k)
                            temporaryMatrix[j, k] = temporaryMatrix[j + 1, k];
                    --h;
                }
                else ++l;
            }
            n -= count;
            for (int i = 0; i < m; ++i)
                temporaryMatrix[n, i] = i;
            //Убираем доминируемые столбцы
            for (int i = 0; i < m; ++i)
            {
                bool dominated;
                if (temporaryMatrix[n, i] == -1)
                    continue;
                for (int j = 0; j < m; ++j)
                {
                    if (temporaryMatrix[n, j] != -1 && j != i)
                    {
                        dominated = true;
                        for (int k = 0; k < n && dominated; ++k)
                        {
                            if (temporaryMatrix[k, i] >= temporaryMatrix[k,
j])
                                dominated = false;
                        }
                        if (dominated)
                            temporaryMatrix[n, j] = -1;
                    }
                }
            }
            count = 0;
            for (int i = 0; i < m; ++i)
```

```
            {
                if (temporaryMatrix[n, i] == -1)
                    ++count;
            }
            l = 0; h = count;
            while (h > 0)
            {
                if (temporaryMatrix[n, l] == -1)
                {
                    for (int j = l; j < m; ++j)
                        for (int k = 0; k < n + 1; ++k)
                            temporaryMatrix[k, j] = temporaryMatrix[k, j + 1];
                    --h;
                }
                else ++l;
            }
            m -= count;
            simpleMatrix = new int[n + 1, m + 1];
            for (int i = 0; i < n + 1; ++i)
            {
                for (int j = 0; j < m + 1; ++j)
                    simpleMatrix[i, j] = temporaryMatrix[i, j];
            }
            bool g;
            g = false;
        }
        private void positive()
        {
            coefPositive = simpleMatrix[0, 0];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
                    if (coefPositive > simpleMatrix[i, j])
                        coefPositive = simpleMatrix[i, j];
            if (coefPositive < 0)
            {
                for (int i = 0; i < n; ++i)
                    for (int j = 0; j < m; ++j)
                        simpleMatrix[i, j] += coefPositive * (-1);
            }
        }
        private void Vald()
        {
            int indexVald = 0;
            for (int i = 1; i < 5; ++i)
                if (paymentMatrix[indexVald, 6] < paymentMatrix[i, 6])
                    indexVald = i;
            tbValdStrategy.Text =
basicTable.Rows[indexVald].Cells[0].Value.ToString();
            tbValdValue.Text = paymentMatrix[indexVald, 6].ToString();

        }
        private void Savidge()
```

```
        {
            double[,] riskMatrix = new double[5, 6];
            for (int i = 0; i < 5; ++i)
                for (int j = 0; j < 6; ++j)
                    riskMatrix[i, j] = paymentMatrix[5, j] - paymentMatrix[i,
j];

            double[] savidgeRes = new double[5];
            for (int i = 0; i < 5; ++i)
            {
                savidgeRes[i] = paymentMatrix[i, 0];
                for (int j = 1; j < 6; ++j)
                    if (savidgeRes[i] < paymentMatrix[i, j])
                        savidgeRes[i] = paymentMatrix[i, j];
            }
            int savidgeIndex = 0;
            for (int i = 1; i < 5; ++i)
                if (savidgeRes[savidgeIndex] < savidgeRes[i])
                    savidgeIndex = i;
            tbSavidgeStrategy.Text =
basicTable.Rows[savidgeIndex].Cells[0].Value.ToString();
            tbSavidgeValue.Text = savidgeRes[savidgeIndex].ToString();
        }
        private void Hurwitz()
        {
            double k = 0.6;
            double[] hurwitzValue = new double[5];
            int hurwitzIndex = 0;
            for(int i = 0; i < 5; ++i)
            {
                hurwitzValue[i] = paymentMatrix[i, 0];
                for (int j = 1; j < 6; ++j)
                    if (paymentMatrix[i, j] > hurwitzValue[i])
                        hurwitzValue[i] = paymentMatrix[i, j];
            }
            for (int i = 0; i < 5; ++i)
            {
                hurwitzValue[i] = paymentMatrix[i, 6] * k + (1 - k) *
hurwitzValue[i];
                if (hurwitzValue[i] > hurwitzValue[hurwitzIndex] && i != 0)
                    hurwitzIndex = i;
            }
            tbHurwitzStrategy.Text =
basicTable.Rows[hurwitzIndex].Cells[0].Value.ToString();
            tbHurwitzValue.Text = hurwitzValue[hurwitzIndex].ToString();
        }
        private void solve()
        {
            //для игрока
            double[,] simplexTable = new double[n + 1, m + 1];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
```

```
                    simplexTable[i, j + 1] = simpleMatrix[i, j];
            for (int i = 0; i < n; ++i)
                simplexTable[i, 0] = 1;
            for (int j = 1; j < m + 1; ++j)
                simplexTable[n, j] = -1;
            simplexTable[n, 0] = 0;
            double[] resultPlayer = new double[m];
            Simplex solutionPlayer = new Simplex(simplexTable);
            simplexTable = solutionPlayer.calculate(resultPlayer);
            double playValue = 0;
            for (int i = 0; i < resultPlayer.Length; ++i)
                playValue += resultPlayer[i];
            //для природы
            double[] resultNature = new double[n];
            int str = simplexTable.GetLength(0) - 1;
            int col = simplexTable.GetLength(1) - resultNature.Length;
            for (int i = 0; i < resultNature.Length; ++i)
                resultNature[i] = simplexTable[str, col + i];
            //Получение результатов - p, q и V
            V = 1 / playValue;
            for (int i = 0; i < n; ++i)
            {
                p[simpleMatrix[i, m]] = resultNature[i] * V;
            }
            for (int i = 0; i < m; ++i)
            {
                q[simpleMatrix[n, i]] = resultPlayer[i] * V;
            }
            if (coefPositive < 0)
                V += coefPositive;
        }}}
```

ПРИЛОЖЕНИЕ В

Листинг функции Симплекс-метода

```
namespace matrixGame
{
    class Simplex
    {
        //source - симплекс таблица без базисных переменных
        double[,] table; //симплекс таблица
        int m, n;
        List<int> basis; //список базисных переменных
        public Simplex(double[,] source)
        {
            m = source.GetLength(0);
            n = source.GetLength(1);
            table = new double[m, n + m - 1];
            basis = new List<int>();
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < table.GetLength(1); j++)
                {
                    if (j < n)
```

```
                        table[i, j] = source[i, j];
                    else
                        table[i, j] = 0;
                }
                //выставляем коэффициент 1 перед базисной переменной в строке
                if ((n + i) < table.GetLength(1))
                {
                    table[i, n + i] = 1;
                    basis.Add(n + i);
                }
            }
            n = table.GetLength(1);
        }
        //result - в этот массив будут записаны полученные значения X
        public double[,] calculate(double[] result)
        {
            int mainCol, mainRow; //ведущие столбец и строка
            while (!IsItEnd())
            {
                mainCol = findMainCol();
                mainRow = findMainRow(mainCol);
                basis[mainRow] = mainCol;
                double[,] new_table = new double[m, n];
                for (int j = 0; j < n; j++)
                    new_table[mainRow, j] = table[mainRow, j] / table[mainRow,
mainCol];
                for (int i = 0; i < m; i++)
                {
                    if (i == mainRow)
                        continue;
                    for (int j = 0; j < n; j++)
                        new_table[i, j] = table[i, j] - table[i, mainCol] *
new_table[mainRow, j];
                }
                table = new_table;
            }
            //заносим в result найденные значения X
            for (int i = 0; i < result.Length; i++)
            {
                int k = basis.IndexOf(i + 1);
                if (k != -1)
                    result[i] = table[k, 0];
                else
                    result[i] = 0;
            }
            return table;
        }
        private bool IsItEnd()
        {
            bool flag = true;
            for (int j = 1; j < n; j++)
            {
```

```
                if (table[m - 1, j] < 0)
                {
                    flag = false;
                    break;
                }
            }
            return flag;
        }
        private int findMainCol()
        {
            int mainCol = 1;
            for (int j = 2; j < n; j++)
                if (table[m - 1, j] < table[m - 1, mainCol])
                    mainCol = j;
            return mainCol;
        }

        private int findMainRow(int mainCol)
        {
            int mainRow = 0;
            for (int i = 0; i < m - 1; i++)
                if (table[i, mainCol] > 0)
                {
                    mainRow = i;
                    break;
                }
            for (int i = mainRow + 1; i < m - 1; i++)
                if ((table[i, mainCol] > 0) && ((table[i, 0] / table[i,
mainCol]) < (table[mainRow, 0] / table[mainRow, mainCol])))
                    mainRow = i;

            return mainRow;
        }
    }
}
```