Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б. Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

Разработка кроссплатформенного web-приложения расчета процесса получения смеси химических веществ минимальной стоимости

домашняя работа

**по дисциплине**

**«Моделирование процессов и объектов в АСУ ТП»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

|  |  |
| --- | --- |
| Студент  группы НМТ-413901 | Е. А. Куц |
| Преподаватели  профессора, д.т.н. | Н. А. Спирин  В. В. Лавров |

Екатеринбург

2024

Содержание

[Домашняя работа «Разработка web-приложения расчета получения смеси химических веществ минимальной стоимости» 3](#_Toc197001977)

[1.1 Цель работы 3](#_Toc197001978)

[1.2 Постановка задачи 3](#_Toc197001979)

[1.3 Ход проведения работы 3](#_Toc197001980)

[1.3.1 Создание модели расчета в таблицах MS Excel 3](#_Toc197001981)

[1.3.2 Разработка WEB-приложения с помощью ASP.NET Core MVC 5](#_Toc197001982)

[1.4 Выводы 10](#_Toc197001983)

[Приложение А. Листинг кода HomeController.cs 11](#_Toc197001984)

[Приложение Б. Листинг кода MixtureSolver.cs 13](#_Toc197001985)

[Приложение В. Листинг кода Index.cshtml 15](#_Toc197001986)

[Приложение Г. Листинг кода Result.cshtml 17](#_Toc197001987)

Домашняя работа «Разработка web-приложения расчета получения смеси химических веществ минимальной стоимости»

1.1 Цель работы

Цель данной работы — изучить и освоить практические аспекты разработки кроссплатформенного web-приложения расчета процесса получения смеси химических веществ минимальной стоимости.

1.2 Постановка задачи

Необходимо составить смесь, в состав которой должно входить не менее определенного количества химических веществ видов *В1, В2,…, Вn*. Количество единиц химических веществ, содержащихся в 1 кг сырьевых материалов *A1, A2,…, Am,* пределы содержания химических элементов в смеси и цены сырья каждого вида заданы в таблице 4.9.1.

Составить наиболее дешевую смесь, содержащую не менее заданного количества химических веществ каждого вида при соблюдении ограничения на общий расход смеси, кг.

Исходные данные для вариантапредставлены в табл. 4.9.1.

Таблица 4.9.1 – Исходные данные для расчета (вариант 1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вещество | Нижний предел содержания вещества в смеси, ед. | Количество единиц вещества, содержащегося в 1 кг сырья соответствующего вида, ед. | | | |
| *A*1 | *A*2 | *A*3 | *A*4 |
| *В1* | 15 | 12 | 20 | 12 | 20 |
| *В2* | 15 | 12 | 18 | 18 | 14 |
| *В3* | 70 | 76 | 62 | 70 | 66 |
| Стоимость 1 кг сырья соответствующего вида, у.е. | | 3,5 | 5,2 | 4,0 | 4,6 |

Всего необходимо приготовить 1 кг смеси

1.3 Ход проведения работы

1.3.1 Создание модели расчета в таблицах MS Excel

Для начального построения модели задачи по расчету состава химической смеси с минимальной стоимостью была использована электронная таблица MS Excel (рисунок 1.3.1). Это позволило наглядно представить структуру задачи, определить набор входных параметров, а также опробовать расчет вручную для нескольких тестовых случаев.

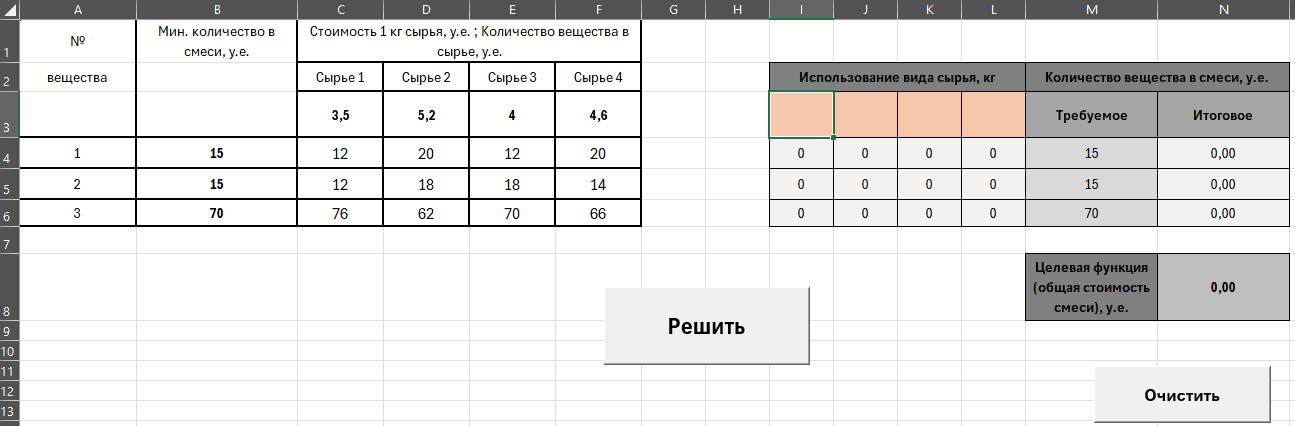


Рисунок 1.3.1 – Пользовательский интерфейс в Excel

Модель базируется на типичной задаче линейного программирования: необходимо определить, в каком количестве использовать различные типы сырья, чтобы покрыть минимальные требования по содержанию каждого вещества, при этом суммарная стоимость сырья была минимальной.

Исходные данные модели включают:

* требуемые количества каждого вещества в смеси (у.е.);
* стоимости единицы массы каждого типа сырья (у.е./кг);
* содержание каждого вещества в одном килограмме соответствующего сырья (у.е.).

Для представления задачи в виде таблицы были использованы следующие элементы:

* строки, соответствующие веществам;
* столбцы, соответствующие типам сырья;
* дополнительный столбец — требуемые количества веществ;
* дополнительная строка — стоимости сырья.

В центральной части таблицы исходных данных (слева на рисунке 1.3.1) расположена матрица содержания веществ в сырье. В ячейках этой матрицы указано, какое количество вещества содержится в одном килограмме соответствующего сырья. Это позволяет выразить каждое ограничение задачи как неравенство, в котором сумма вклада каждого сырья по данному веществу должна быть не меньше требуемого значения.

Для оценки стоимости смеси в ячейку N8 была добавлена формула, рассчитывающая сумму произведений объема каждого сырья на его цену. Эта сумма и является целевой функцией задачи — она должна быть минимизирована.

На основе этой таблицы удобно выполнять отладку модели: можно подставлять различные значения объемов сырья вручную и отслеживать, выполняются ли ограничения и какая получается итоговая стоимость.

Для решения пользователю нужно ввести исходные данные задачи в таблицу слева, такие как минимальные содержания каждого вещества в смеси, стоимость 1 кг каждого сырья и количество каждого вещества в каждом сырье. После этого при нажатии на кнопку «Решить» в таблице справа алгоритм Поиска решения подберет оптимальные потребности в каждом сырье, при котором общая стоимость производства смеси сводиться в минимуму (рисунок 1.3.2).

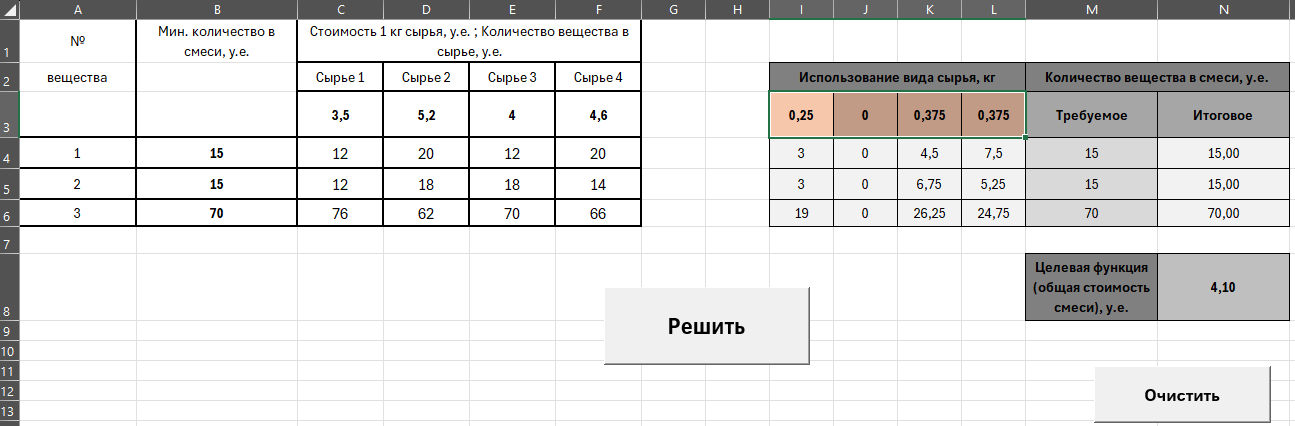


Рисунок 1.3.2 – Результат расчета в Excel

1.3.2 Разработка WEB-приложения с помощью ASP.NET Core MVC

На втором этапе проектирования задача расчета стоимости смеси была перенесена из среды MS Excel в полноценное кроссплатформенное веб-приложение, реализованное на платформе ASP.NET Core. Ядром приложения стала служба MixtureSolver, реализующая алгоритм линейного программирования с использованием библиотеки Google OR-Tools.

Класс MixtureSolver представляет собой модуль, отвечающий за построение и решение математической модели задачи (рисунок 1.3.3). Основная логика сосредоточена в его методе Solve, который принимает три параметра:

* список требуемых количеств веществ;
* список цен на единицу массы каждого сырья;
* матрицу содержания веществ в единице сырья.

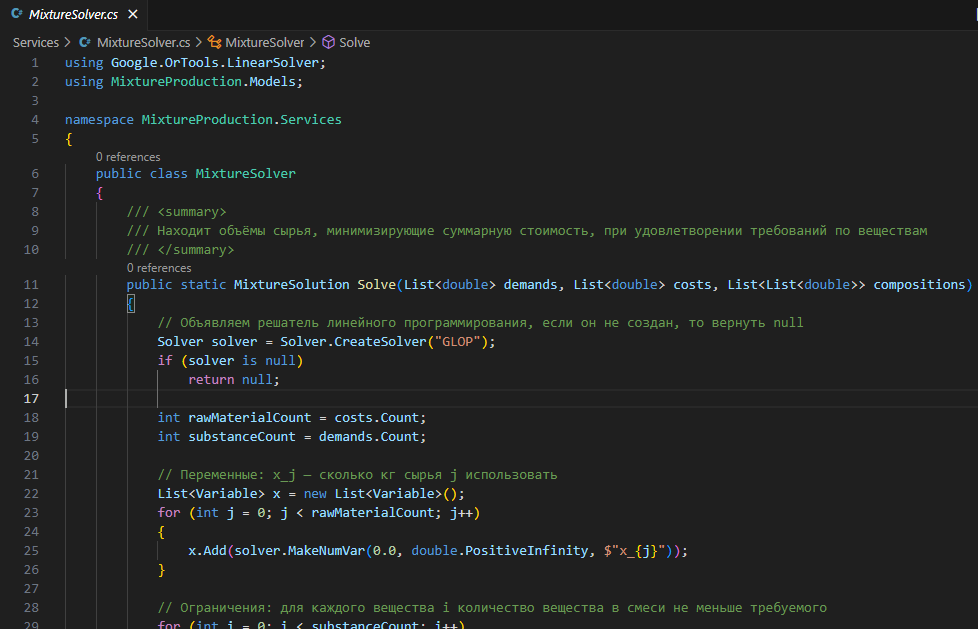


Рисунок 1.3.3 – Фрагмент кода MixtureSolver

На первом этапе внутри метода объявляются переменные, отражающие объём каждого типа сырья. Эти переменные непрерывны и ограничены снизу нулём, что соответствует реальным условиям задачи (сырьё не может использоваться в отрицательном объёме).

Далее формируются ограничения по каждому веществу: суммарное содержание вещества во всех выбранных типах сырья должно быть не менее заданного значения. Эти ограничения создаются циклически и добавляются в решатель при помощи метода MakeConstraint.

Затем задаётся целевая функция — суммарная стоимость смеси. Она выражается как сумма произведений объёма каждого сырья на его цену. Для задания функции используется метод SetCoefficient над объектом Objective, с последующим вызовом SetMinimization().

После определения всех переменных, ограничений и целевой функции вызывается метод solver.Solve(), возвращающий оптимальное решение задачи.

Итоговое решение извлекается из модели: значения переменных x\_j, соответствующие объёмам сырья, а также минимальная стоимость смеси. Эти данные возвращаются в виде объекта MixtureSolution (рисунок 1.3.4).

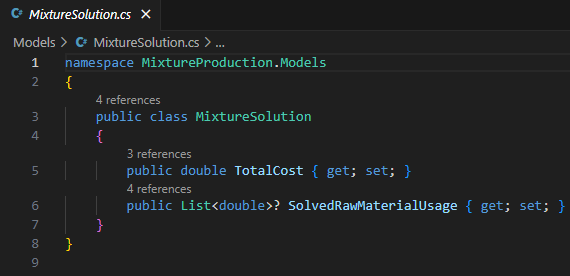


Рисунок 1.3.4 – Модель MixtureSolution для возврата результата

Для организации ввода исходных данных пользователем в веб-приложении был разработан адаптивный интерфейс на Razor-странице Index.cshtml. Он позволяет в интерактивной форме задать:

* требуемые количества веществ;
* цены на каждый тип сырья;
* состав сырья по каждому веществу (матрица содержаний).

Форма реализована с использованием HTML-таблицы (рисунок 1.3.5), которую пользователь может динамически изменять: добавлять и удалять строки (вещества) и столбцы (типы сырья) при помощи соответствующих кнопок.

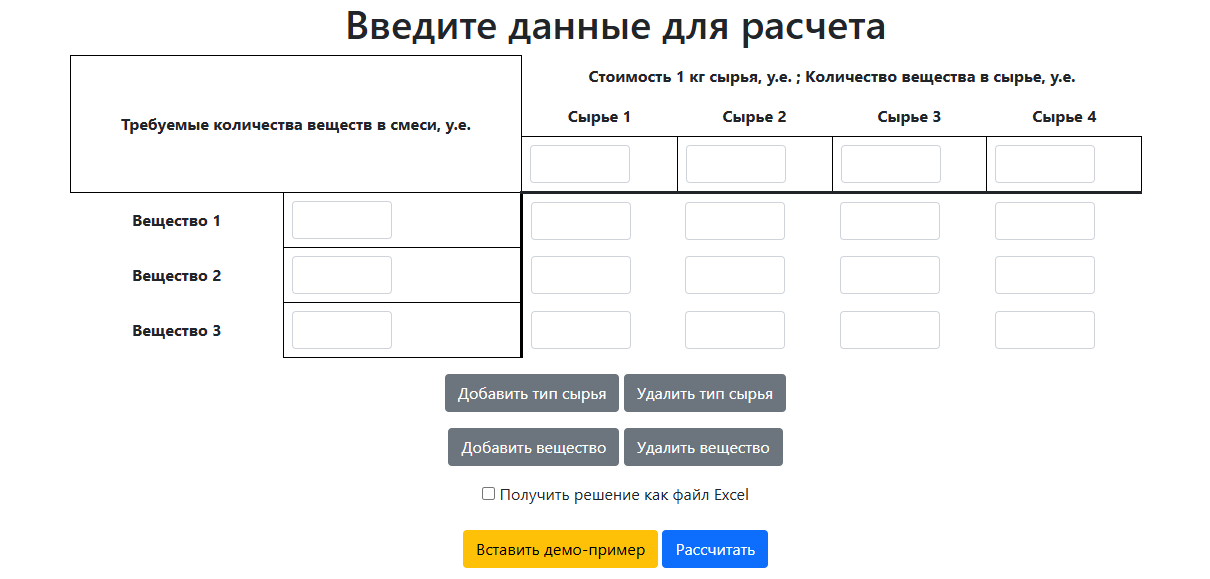


Рисунок 1.3.5 – Форма ввода исходных данных для расчета

Форма инициализируется объектом InputModel, содержащим два параметра: SubstanceTypeCount и RawMaterialCount, задающие начальные размеры таблицы. В первой колонке пользователь указывает минимально необходимое количество каждого вещества (у.е.); в первой строке над матрицей указываются цены за 1 кг каждого сырья; основная часть таблицы содержит матрицу содержания: сколько вещества содержится в одном килограмме того или иного сырья.

В нижней части интерфейса расположены кнопки управления таблицей:

* «Добавить тип сырья» / «Удалить тип сырья» — изменяют количество столбцов;
* «Добавить вещество» / «Удалить вещество» — изменяют количество строк.

Чекбокс «Получить решение как файл Excel» позволяет указать необходимость формирования файла с результатом.

Для демонстрации возможностей, что может быть полезно при первом знакомстве пользователя с приложением, предусмотрена кнопка «Вставить демо-пример», автоматически заполняющая поля формы данными из примера.

После заполнения данных пользователь нажимает кнопку «Рассчитать», и форма отправляется в контроллер HomeController, метод SolveMixture. Серверная часть получает данные в виде списков и матриц, обрабатывает их в модуле MixtureSolver и возвращает результат.

В ответ пользователю будет выведена страница с введенными им данными и решение, подобранное программой (рисунок 1.3.6).

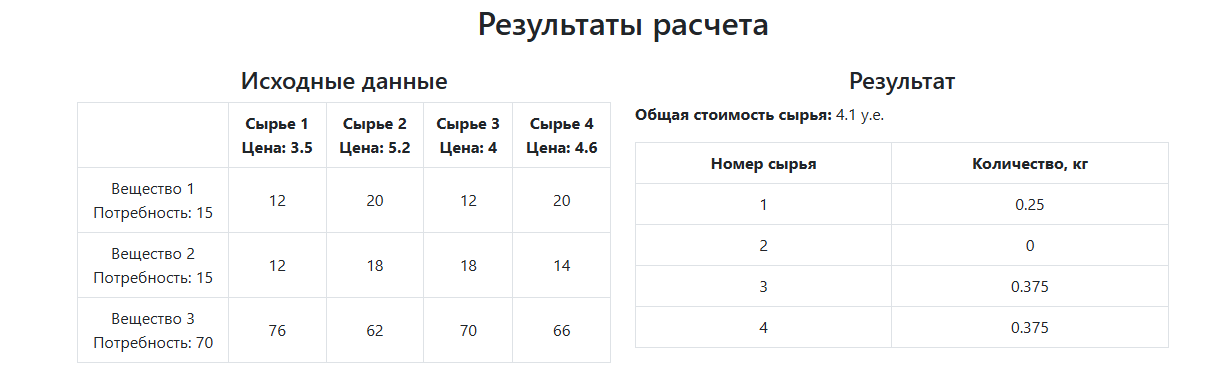


Рисунок 1.3.6 – Вывод результата расчета в Web-приложении

Чтобы сохранить результат, приложение может записать решение в таблицу Excel и вернуть ее пользователю. Для этого следует оставить галочку под формой ввода данных. На рисунке 1.3.7 видно, как размещаются данные в таблице.

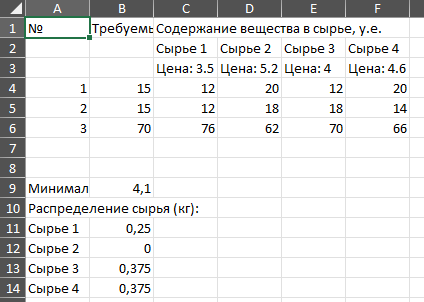


Рисунок 1.3.7 – Файл Excel для вывода результат расчета

1.4 Выводы

В ходе выполнения работы были изучены и освоены практические аспекты разработки кроссплатформенного web-приложения расчета процесса получения смеси химических веществ минимальной стоимости.

Приложение А. Листинг кода HomeController.cs

using Microsoft.AspNetCore.Mvc;

using MixtureProduction.Models;

using MixtureProduction.Services;

using System.Diagnostics;

using System.Text.Json;

namespace ChemicalMixture.Controllers

{

    public class HomeController : Controller

    {

        private readonly ILogger<HomeController> \_logger;

        public HomeController(ILogger<HomeController> logger)

        {

            \_logger = logger;

        }

        public IActionResult Index()

        {

            return View(new InputModel { SubstanceTypeCount = 3, RawMaterialCount = 4 });

        }

        [HttpPost]

        public IActionResult SolveMixture(List<double> demands, List<double> costs, List<List<double>> compositions, bool createExcelFile)

        {

            var solution = MixtureSolver.Solve(demands, costs, compositions);

            if (createExcelFile)

            {

                // ������������ ������� ��������, ��� compositions � solution, � ������ JSON ��� ��������

                string compositionsJson = JsonSerializer.Serialize(compositions);

                string solutionJson = JsonSerializer.Serialize(solution);

                return RedirectToAction("ExportToExcel", new

                {

                    Demands = demands,

                    Costs = costs,

                    compositionsJson = compositionsJson,

                    solutionJson = solutionJson

                });

            }

            var resultModel = new ResultModel

            {

                Demands = demands,

                Costs = costs,

                Compositions = compositions,

                TotalCost = solution.TotalCost,

                SolvedRawMaterialUsage = solution.SolvedRawMaterialUsage

            };

            return View("Result", resultModel);

        }

        public IActionResult ExportToExcel(List<double> demands, List<double> costs, string compositionsJson, string solutionJson)

        {

            var compositions = JsonSerializer.Deserialize<List<List<double>>>(compositionsJson);

            var solution = JsonSerializer.Deserialize<MixtureSolution>(solutionJson);

            var stream = ExcelFileService.CreateExcelFile(demands, costs, compositions, solution);

            string excelName = $"Mixture\_Solution.xlsx";

            return File(stream, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", excelName);

        }

        public IActionResult Privacy()

        {

            return View();

        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

        public IActionResult Error()

        {

            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

        }

    }

}

Приложение Б. Листинг кода MixtureSolver.cs

using Google.OrTools.LinearSolver;

using MixtureProduction.Models;

namespace MixtureProduction.Services

{

    public class MixtureSolver

    {

        /// <summary>

        /// Находит объёмы сырья, минимизирующие суммарную стоимость, при удовлетворении требований по веществам

        /// </summary>

        public static MixtureSolution Solve(List<double> demands, List<double> costs, List<List<double>> compositions)

        {

            // Объявляем решатель линейного программирования, если он не создан, то вернуть null

            Solver solver = Solver.CreateSolver("GLOP");

            if (solver is null)

                return null;

            int rawMaterialCount = costs.Count;

            int substanceCount = demands.Count;

            // Переменные: x\_j — сколько кг сырья j использовать

            List<Variable> x = new List<Variable>();

            for (int j = 0; j < rawMaterialCount; j++)

            {

                x.Add(solver.MakeNumVar(0.0, double.PositiveInfinity, $"x\_{j}"));

            }

            // Ограничения: для каждого вещества i количество вещества в смеси не меньше требуемого

            for (int i = 0; i < substanceCount; i++)

            {

                Constraint c = solver.MakeConstraint(demands[i], double.PositiveInfinity, $"Demand\_{i}");

                for (int j = 0; j < rawMaterialCount; j++)

                {

                    c.SetCoefficient(x[j], compositions[i][j]);

                }

            }

            // Целевая функция: минимизировать общую стоимость смеси

            Objective objective = solver.Objective();

            for (int j = 0; j < rawMaterialCount; j++)

            {

                objective.SetCoefficient(x[j], costs[j]);

            }

            objective.SetMinimization();

            // Решение

            var resultStatus = solver.Solve();

            if (resultStatus != Solver.ResultStatus.OPTIMAL)

                return null;

            // Результат

            return new MixtureSolution

            {

                TotalCost = solver.Objective().Value(),

                SolvedRawMaterialUsage = x.Select(v => Math.Round(v.SolutionValue(), 4)).ToList()

            };

        }

    }

}

Приложение В. Листинг кода Index.cshtml

@model InputModel

@{

    ViewData["Title"] = "Расчет минимальной стоимости смеси химических веществ";

}

<div class="container mt-4">

    <h1 class="text-center">Введите данные для расчета</h1>

    <form asp-action="SolveMixture" method="post">

        <div class="table-responsive">

            <table class="table table-borderless" id="table">

                <thead>

                    <tr>

                        <th class="bordered centered" colspan="2" rowspan="3">

                            Требуемые количества веществ в смеси, у.е.

                        </th>

                        <th class="centered" colspan="4">

                            Стоимость 1 кг сырья, у.е. ; Количество вещества в сырье, у.е.

                        </th>

                        <th></th>

                    </tr>

                    <tr>

                        @for (int i = 0; i < Model.RawMaterialCount; i++)

                        {

                            <th class="centered">

                                Сырье @(i + 1)

                            </th>

                        }

                    </tr>

                    <tr>

                        @for (int i = 0; i < Model.RawMaterialCount; i++)

                        {

                            <th class="bordered border-bottom-bold centered">

                                <input type="number" step="any" name="Costs[@i]" class="form-control text-center fw-bold" min="0" required />

                            </th>

                        }

                    </tr>

                </thead>

                <tbody>

                    @for (int i = 0; i < Model.SubstanceTypeCount; i++)

                    {

                        <tr>

                            <th class="centered">

                                Вещество @(i + 1)

                            </th>

                            <th class="bordered border-right-bold">

                                <input type="number" step="0.1" name="Demands[@i]" class="form-control text-center fw-bold" min="0" />

                            </th>

                            @for (int j = 0; j < Model.RawMaterialCount; j++)

                            {

                                <td>

                                    <input type="number" step="0.1" name="Compositions[@i][@j]" class="form-control text-center" required>

                                </td>

                            }

                        </tr>

                    }

                </tbody>

            </table>

        </div>

        <div class="text-center">

            <div>

                <button type="button" id="add-column-btn" class="btn btn-secondary mb-3">Добавить тип сырья</button>

                <button type="button" id="drop-column-btn" class="btn btn-secondary mb-3">Удалить тип сырья</button>

            </div>

            <div>

                <button type="button" id="add-row-btn" class="btn btn-secondary mb-3">Добавить вещество</button>

                <button type="button" id="drop-row-btn" class="btn btn-secondary mb-3">Удалить вещество</button>

            </div>

            <input type="checkbox" name="createExcelFile" id="createExcelFile" value="true" />

            <label for="createExcelFile">Получить решение как файл Excel</label>

        </div>

        <div class="text-center mt-4">

            <button type="button" id="insert-demo-btn" class="btn btn-warning">Вставить демо-пример</button>

            <button type="submit" class="btn btn-primary">Рассчитать</button>

        </div>

    </form>

</div>

<style>

    .form-control {

        max-width: 100px;

    }

</style>

Приложение Г. Листинг кода Result.cshtml

@model ResultModel

@{

    ViewData["Title"] = "Результаты расчета";

}

<div class="container mt-5">

    <h2 class="text-center mb-4">Результаты расчета</h2>

    <div class="row">

        <div class="col-md-6">

            <h4 class="text-center">Исходные данные</h4>

            <table class="table table-bordered text-center">

                <thead>

                    <tr>

                        <th></th>

                        @for (int i = 0; i < Model.Costs.Count; i++)

                        {

                                <th>Сырье @(i + 1)<br />Цена: @Model.Costs[i]</th>

                        }

                    </tr>

                </thead>

                <tbody>

                    @for (int i = 0; i < Model.Demands.Count; i++)

                    {

                            <tr>

                                <td>Вещество @(i + 1)<br />Потребность: @Model.Demands[i]</td>

                            @for (int j = 0; j < Model.Costs.Count; j++)

                            {

                                        <td>@Model.Compositions[i][j]</td>

                            }

                            </tr>

                    }

                </tbody>

            </table>

        </div>

        <div class="col-md-6">

            <h4 class="text-center">Результат</h4>

            <p><strong>Общая стоимость сырья:</strong> @Model.TotalCost у.е.</p>

            @if (Model.SolvedRawMaterialUsage != null)

            {

                    <table class="table table-bordered text-center">

                        <thead>

                            <tr>

                                <th>Номер сырья</th>

                                <th>Количество, кг</th>

                            </tr>

                        </thead>

                        <tbody>

                        @for (int j = 0; j < Model.SolvedRawMaterialUsage.Count; j++)

                        {

                                    <tr>

                                        <td>@(j + 1)</td>

                                        <td>@Model.SolvedRawMaterialUsage[j]</td>

                                    </tr>

                        }

                        </tbody>

                    </table>

            }

            else

            {

                    <p>Решение не найдено.</p>

            }

        </div>

    </div>

</div>