eISSN 2311-2468
Том 11, № 16. 2023
Vol. 11, no. 16. 2023

электронное периодическое издание
для студентов и аспирантов

Огарёв-онлайн
Ogarev-online

https://journal.mrsu.ru

ШАМАНАЕВ П. А., КАТИН Д. А., ДЕСЯЕВ Е. В.

РЕАЛИЗАЦИЯ АЛГОРИТМА НАХОЖДЕНИЯ РЕЗОЛЬВЕНТЫ МАТРИЦЫ С ИСПОЛЬЗОВАНИЕМ ПРИСОЕДИНЕННОЙ МАТРИЦЫ И ХАРАКТЕРИСТИЧЕСКОГО МНОГОЧЛЕНА

Аннотация. В настоящей работе излагается реализация алгоритма вычисления резольвенты матрицы с использованием присоединенной матрицы и характеристического многочлена матрицы на языке Python. На графиках приведены зависимости скорости работы алгоритма при различных размерностях матрицы.

Ключевые слова: резольвента матрицы, присоединенная матрица, характеристический многочлен, наибольший общий делитель, Python.

SHAMANAEV P. A., KATIN D. A., DESYAEV E. V.

IMPLEMENTATION OF AN ALGORITHM FOR FINDING THE RESOLVENT OF A MATRIX USING THE ADJECT MATRIX AND CHARACTERISTIC POLYNOMIAL

Abstract. The article describes the implementation of an algorithm for calculating the resolvent of a matrix using the adjoint matrix and the characteristic polynomial of the matrix in Python. The graphs show the speed of the algorithm for various matrix dimensions.

Keywords: matrix resolvent, adjoint matrix, characteristic polynomial, greatest common divisor, Python.

Введение. При решении многих задач требуется вычислять резольвенту матрицы [1], не зная ее собственных значений. В частности, такая задача возникает при решении систем линейных алгебраических уравнений с малым параметром методом Ляпунова-Шмидта [2; 3].

В работе [1] приведен подход вычисления резольвенты матрицы с использованием присоединенной матрицы и характеристического многочлена матрицы. Для вычисления последних Д. К. Фаддеевым предложен метод одновременного вычисления коэффициентов характеристического многочлена и присоединенной матрицы [1].

Изложим алгоритм вычисления резольвенты матрицы на основе подхода, изложенного в [1] и метода Д. К. Фаддеева одновременного вычисления коэффициентов характеристического многочлена и присоединенной матрицы.

Алгоритм вычисления резольвенты матрицы. Для вычисления резольвенты постоянной $(m \times m)$ −матрицы A

$$R\ (\lambda) = [\lambda E - A]^{-1}$$

воспользуемся формулой [1, с. 112]

$$R\ (\lambda) = \frac{1}{\chi_{min}(\lambda)} C(\lambda),$$

где $C(\lambda)$ – приведенная присоединенная матрица для матрицы $[\lambda E - A]$ (см. [1, с. 99]); $\chi_{min}(\lambda)$ – минимальный характеристический многочлен матрицы A.

Приведенную присоединенную матрицу $C(\lambda)$ и минимальный характеристический многочлен матрицы A будем вычислять по формулам [1, с. 99-100]

$$C(\lambda) = \frac{1}{d(\lambda)}\ B_A(\lambda)$$

$$\chi_{min}(\lambda) = \frac{\Delta(\lambda)}{d(\lambda)}$$

где $B_A(\lambda)$ – присоединенная матрица для матрицы A [1, с. 92], $d(\lambda)$ – наибольший общий делитель всех элементов матрицы $B_A(\lambda)$,

$$\Delta(\lambda) = \lambda^n - p_1\lambda^{n-1} - p_2\lambda^{n-2} - \cdots - p_{n-1}\lambda - p_n,$$

– характеристический многочлен матрицы A.

Для нахождения присоединенной матрицы $B_A(\lambda)$ воспользуемся формулой [1, с. 94]

$$B_A(\lambda) = B_0\lambda^{n-1} + B_1\lambda^{n-2} + B_2\lambda^{n-3} + \cdots + B_{n-1},$$

где B_0 – единичная $(m \times m)$ −матрица, а $(m \times m)$ −матрицы B_k , $k = 1, \dots, n-1$, находятся методом Д. К. Фаддеева по формулам [1, с. 97]

$$A_k = A\ B_{k-1},$$

$$p_k = \frac{1}{k}\ Sp(A_k),$$

$$B_k = A_k - p_k E,$$

$$k = 1, \dots, m.$$

Здесь E_0 – единичная $(m \times m)$ −матрица.

Программная реализация алгоритм вычисления резольвенты матрицы. Приведем фрагменты алгоритма вычисления резольвенты матрицы A на языке Python с использованием пакета SymPy.

Фрагмент кода инициализации матрицы A с целыми случайными элементами a_{ij}, $i,j = 1,\dots,m$ из отрезка $[-10,10]$:

```
max_aij = 10
A = np.random.randint(-max_aij, max_aij + 1, (m, m))
```

Фрагмент кода реализации метода Д. К. Фаддеева одновременного вычисления характеристического многочлена $\Delta(\lambda)$ и присоединенной матрицы $B_A(\lambda)$:

```
def fadeev_method(A):
    m = len(A)
    E = sympy.eye(m)
    B = [0 for i in range(m + 1)]
    B[0] = sympy.eye(m)
    tmpA = [0 for i in range(m + 1)]
    p = [0 for i in range(m + 1)]
    for k in range(1, m + 1):
        tmpA[k] = A * B[k - 1]
        p[k] = 1 / k * matrix_trace(tmpA[k])
        B[k] = tmpA[k] - p[k] * E
    deltaLambda = lyamda ** m
    for i in range(1, m + 1):
        deltaLambda -= p[i] * lyamda ** (m - i)
    B_A = sympy.zeros(m)
    for k in range(m):
        B_A += B[k] * lyamda ** (m - 1 - k)
    return B_A, deltaLambda
```

Фрагмент кода вычисления наибольшего общего делителя всех элементов присоединенной матрицы $B_A(\lambda)$:

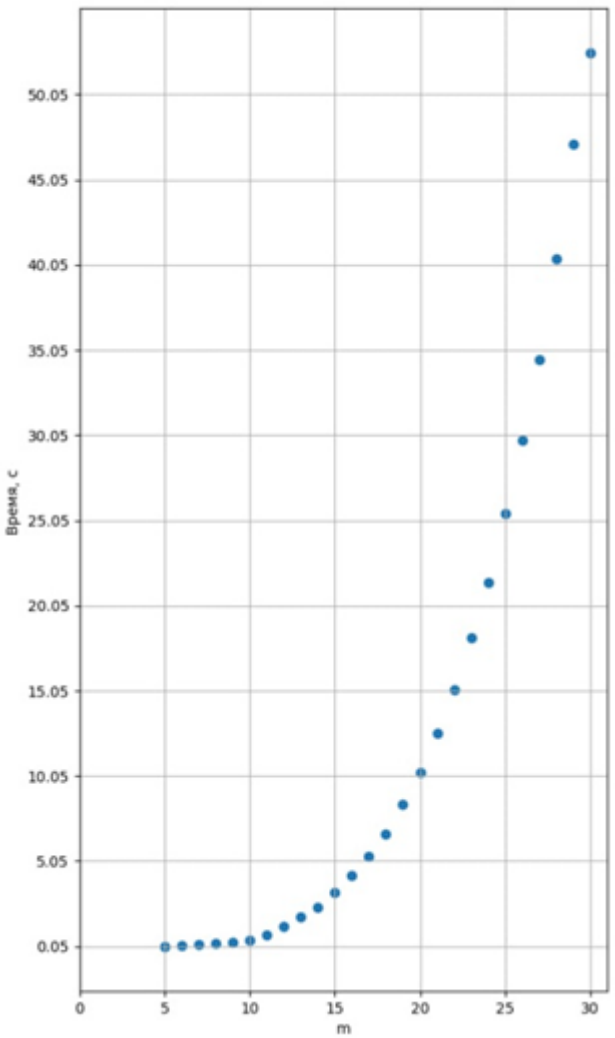
```
d = B_A[0, 0]
for i in range(m):
    for j in range(m):
        d = sympy.gcd(d, B_A[i, j])
```

Фрагмент кода вычисления приведенной присоединенной матрицы $C(\lambda)$, минимального характеристического многочлена $\chi_{min}(\lambda)$ и резольвенты $R\ (\lambda)$ матрицы A:

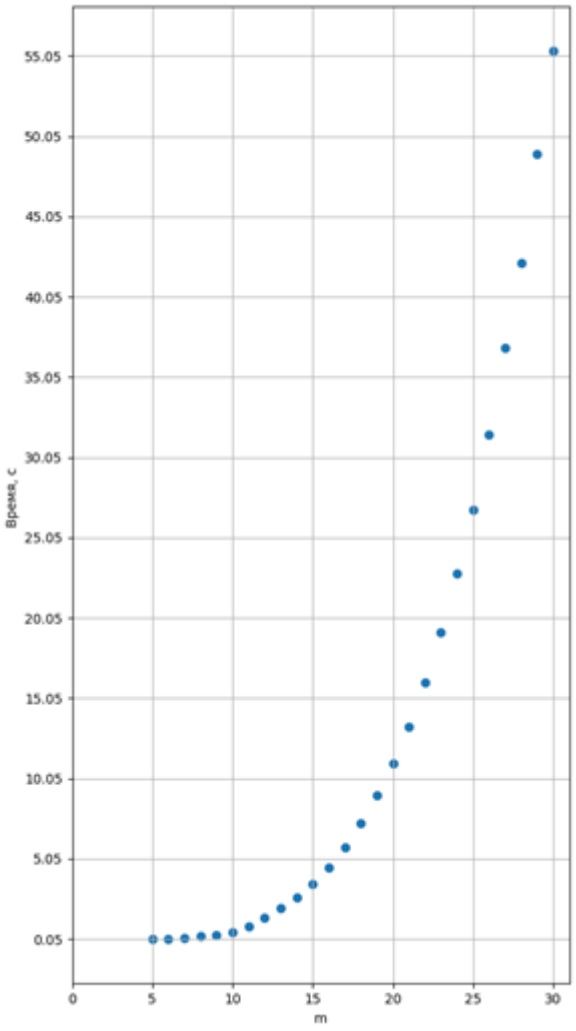
```
C = 1 / d * B_A
x_min = deltaLambda.as_expr() / d
R = 1 / x_min * C
```

Вычислительный эксперимент. Вычисления проводились на ноутбуке с процессором 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz и операционной системой Windows 11 Pro.

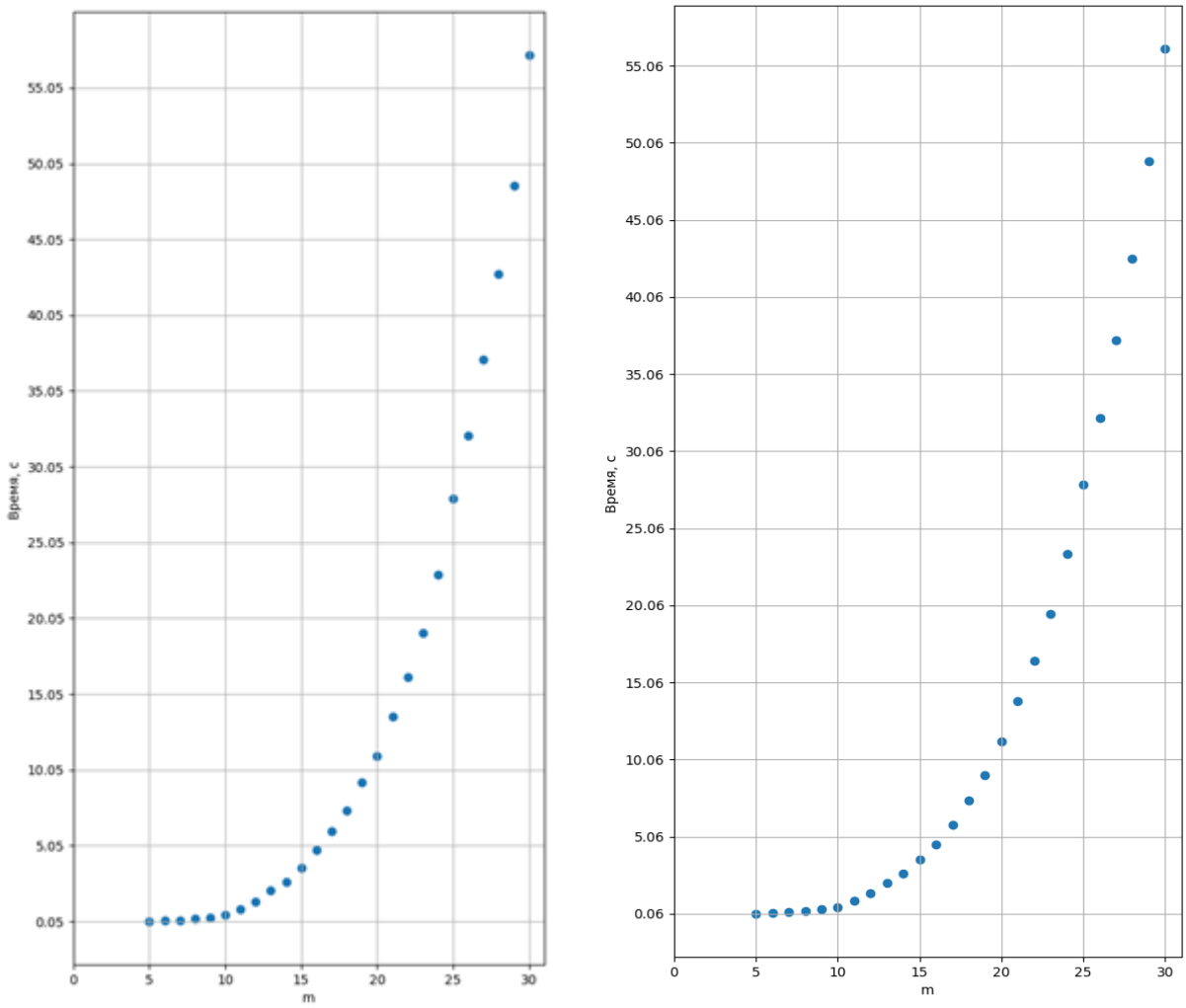
На графиках представлены зависимости скорости работы алгоритма вычисления резольвенты матрицы A с целыми случайными элементами при различных размерностях матрицы.



a)



b)



c) *d*)

Рисунок. Графики зависимостей скорости работы алгоритма от размерности матрицы A при условиях: *a*) $|a_{ij}| \leq 10$, *b*) $|a_{ij}| \leq 10^5$, *c*) $|a_{ij}| \leq 10^6$, *d*) $|a_{ij}| \leq 10^7$, $i, j = 1, \dots, m$

Из графиков видно, что при увеличении значений коэффициентов матрицы A по абсолютной величине скорость работы алгоритма увеличивается незначительно.

Для сравнительного анализа использовалась функция пакета Sympy нахождения обратной матрицы, зависящей от параметра. Вычисления показали, что при $m = 30$ и $|a_{ij}| \leq 10$ эта функция уже затрачивает порядка 10 мин., что значительно превышает скорость работы разработанного алгоритма вычисления резольвенты матрицы с использованием приведенной присоединенной матрицы и минимального характеристического многочлена.

СПИСОК ЛИТЕРАТУРЫ

1. Гантмахер Ф. Р. Теория матриц. – М.: Наука, 1966. – 576 с.
2. Вайнберг М. М., Треногин В. А. Теория ветвления решений нелинейных уравнений. – М.: Наука, 1964. – 524 с.
3. Шаманаев П. А., Прохоров С. А. Алгоритм решения систем линейных алгебраических уравнений с малым параметром методом Ляпунова-Шмидта в регулярном случае [Электронный ресурс] // Математическое моделирование, численные методы и комплексы программ имени Е. В. Воскресенского: IX Международная научная молодежная школа-семинар (Саранск, 8-11 октября 2020 г.). – С. 129–131. – Режим доступа: https://conf.svmo.ru/files/2020/papers/paper40.pdf (дата обращения: 23.11.2023).

ГУЩИНА О. А., КОРЖОВ А. С.

ПРИМЕНЕНИЕ АЛГОРИТМА СЛУЧАЙНОГО ЛЕСА ДЛЯ АВТОМАТИЗАЦИИ КЛАССИФИКАЦИИ КАТЕГОРИЙ ГРУНТОВ

Аннотация. В статье рассматривается создание модели машинного обучения для решения задачи классификации грунтов с использованием ансамбля случайных деревьев решений (случайного леса) для автоматизации определения с максимальной точностью категорий грунтов на основе имеющихся о них данных, включающих такие характеристики как плотность, влажность, фракционный состав и прочие. Также представлен пользовательский интерфейс разработанной программно-информационной системы для проведения предсказательной аналитики с помощью полученной модели.

Ключевые слова: дерево решений, алгоритм случайного леса, машинное обучение, предсказательная аналитика.

GUSHCHINA O. A., KORZHOV A. S.

APPLYING RANDOM FOREST ALGORITHM FOR AUTOMATING CLASSIFICATION OF SOIL CATEGORIES

Abstract. The article discusses the creation of a machine learning model to solve the problem of soil classification using an ensemble of random decision trees (random forest) to automate the determination of soil categories with maximum accuracy based on the data available about them, including such characteristics as soil density, humidity, fractional composition and others. The user interface of the developed software and information system for conducting predictive analytics using the resulting model is also presented.

Keywords: decision tree, random forest algorithm, machine learning, predictive analytics.

Постановка задачи. Пусть дана таблица из результатов проб грунта (более 500 000 экземпляров) с 32 характеристиками различных грунтов. На основе данных взятых проб грунта в Москве и Московской области необходимо определить категорию грунта и сделать заключение о пригодности почвы к застройке определенных типов зданий. Это делается потому, что тип грунта ключевым фактором, влияющим на выбор типа фундамента здания (так как некоторые типы грунта более подходят для строительства строго определенных типов зданий) [1].

Процесс решения задачи. Грунты, представляющие собой комплекс природных материалов, различаются по своему происхождению, составу (супесчаные, глинистые, песчанисто-глинистые, глинисто-песчанистые и пр.), химическому составу, физическим

свойствам (плотность, влажность, текстура, консистенция) и другим характеристикам (палеоповерхность, уровень углерода, минерализованные зоны) [2].

Для решения поставленной задачи целесообразно использовать возможности языка программирования Python, который специально используется в машинном обучении для выполнения научных и коммерческих проектов [3; 4]. Так, библиотека scikit-learn предназначена для непосредственного построения и работы с деревьями решений.

При разработке программно-информационной системы (ПИС) был использован CSV-файл с результатами анализа грунта. Структура CSV-файла представлена в таблице 1. CSV-файл расположен на Google Drive и, после загрузки, данные сохраняются в переменной 'df' типа DataFrame.

Таблица 1

Структура CSV-файла

1	index	Класс грунта
2	input_index	Входное значение класса по результатам первичной обработки экспертами (может совпадать или не совпадать с выходными значениями)
3	index_kod	Дубль поля input_index, но записан как id (число). Можно воспринимать как ранговый показатель, т.к. порядок закреплен – большие значения id лежат глубже меньших
4	prev_index	Предыдущий индекс – проинтерпретированный класс предшествующего (меньшего по глубине) слоя в данной скважине. Важный параметр, поскольку в большинстве случаев породы залегают последовательно и более древние породы не могут залегать выше более молодых
5	bottom	Абсолютные отметки кровли и подошвы слоя (метры над уровнем моря)
6	top	
7	depth_to_paleosurf	Разница между отметкой кровли слоя и поверхностью палеорельефа. Палеорельеф – результат работы по картированию ненарушенной топографии городской территории (до начала хозяйственного использования)
8	depth_to_carbon	Разница между кровлей слоя и поверхностями дочетвертичных отложений
9	depth_to_mz	Разница между кровлей слоя и поверхностями каменноугольных отложений
10	top_MZ_map	top + depth_to_mz
11	top_C_map	top + depth_to_carbon
12	paleosurf	top + depth_to_paleosurf
13	x	Координаты скважины в некоторой системе координат, в нашем случае – в Московской системе координат
14	y	
15	okrug	Округ Москвы. Введен для упрощения учета территориального фактора
16	PreQ_map	Числовая информация с разных геологических карт. Для разных геологических индексов важна разная информация
17	Carb_map	
18	Q_map	
19	Geomorf	

Продолжение таблицы 1

20	genesis	Генезис происхождение грунтов – например речные, морские, болотные, ледниковые отложения на основе входного класса. За каждым классом жестко закреплен генезис. Чаще всего генезис не меняется в процессе интерпретации. Генезис можно рассматривать как кластер более высокого уровня, один генезис всегда включает несколько индексов. Грунты одного индекса не могут быть разного генезиса
21	litol	Литологическое описание грунта Дисперсные грунты: – Глинистые грунты – Глины, Суглинки, Супеси. Относятся к одной группе глинистых грунтов, но это не значит, что это одно и то же. В некоторых слоях (древние каменноугольные отложения – генезис carbon) могут быть только глины, причём только твердые и полутвердые. – Песчаные грунты – пески различной крупности и плотности сложения. – Крупнообломочные грунты (щебень, гравий, галька и т.д.). – Скальные грунты: известняк, доломит, мергель, песчаник. Бывают только в древних отложениях (генезис – carbon, Creatcious, Jurassic). – Специфические органогенные грунты – торф, ил, сапропель. Редкие и опасные для строительства. – Техногенные грунты – отходы человеческой деятельности, строительные материалы и т.д. (генезис – technogen). – Карстовые полости – большие полости в известняках, часто заполненные дисперсными грунтами
22	podoshva	Глубина подошвы слоя
23	prochn	Прочность грунта (только для скальных пород)
24	vkluch	Наличие или отсутствие включений (случайные органические или минеральные тела или предметы, генетически не связанные с почвенными процессами)
25	vlaga	Влажность, водонасыщенность грунта
26	color_lit	Цвет грунта
27	konsist	Консистенция грунта: текучесть / пластичность / твердость и т.д. (только для глинистых пород)
28	krupnost	Крупность (только для песчаных грунтов)
29	kavern	Кавернозность – наличие полостей и пор в скальных грунтах
30	plotn	Плотность сложения песчаных грунтов
31	sohran	Степень разрушения скального грунта
32	cons	konsist в числовом формате

Для эффективной обработки и анализа результатов анализа проб грунта, необходимо реализовать процесс импорта данных из базы данных для их последующей обработки с использованием алгоритмов машинного обучения, а также процесс экспорта полученных результатов. Для этого были проведены следующие манипуляции с данными.

1. Выбор таблицы и данных из корпоративной базы данных с помощью SQL-запросов.

2. Преобразование полученной таблицы в файл CSV и его экспорт.

3. Импорт CSV-файла в среду Jupyter Lab с помощью Pandas.

4. Обработка данных с помощью моделей машинного обучения является основным этап и включает следующие подэтапы обработки информации.

4.1. Преобразование типов данных.

4.2. Обработка пропущенных значений.

4.3. Нормализация данных.

4.4. Построение моделей машинного обучения.

4.5.Преобразование в исходный формат.

5. Подготовка полученных данных к экспорту в формат CSV.

6. Загрузка файла CSV в базу данных с использованием SQL.

На рисунке 1 представлена диаграмма вариантов использования ПИС.

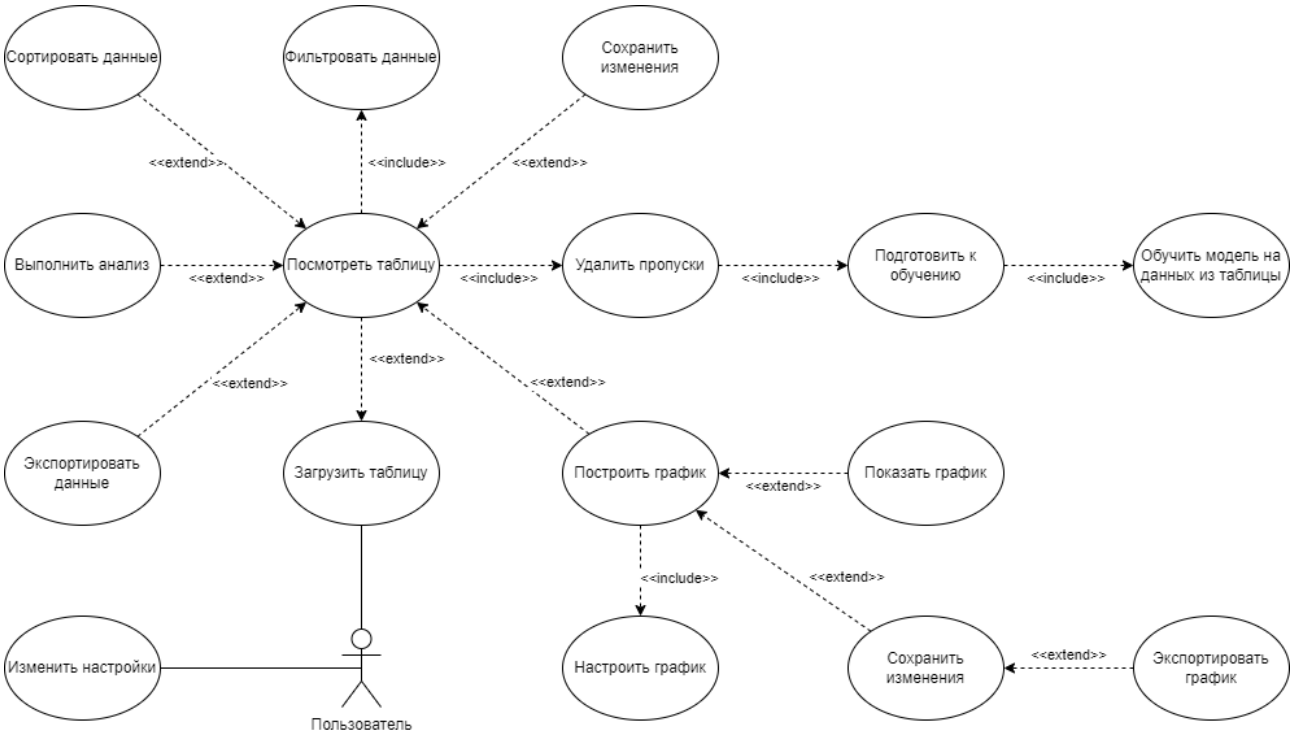


Рис. 1. Диаграмма вариантов использования программно-информационной системы для определения категории грунта.

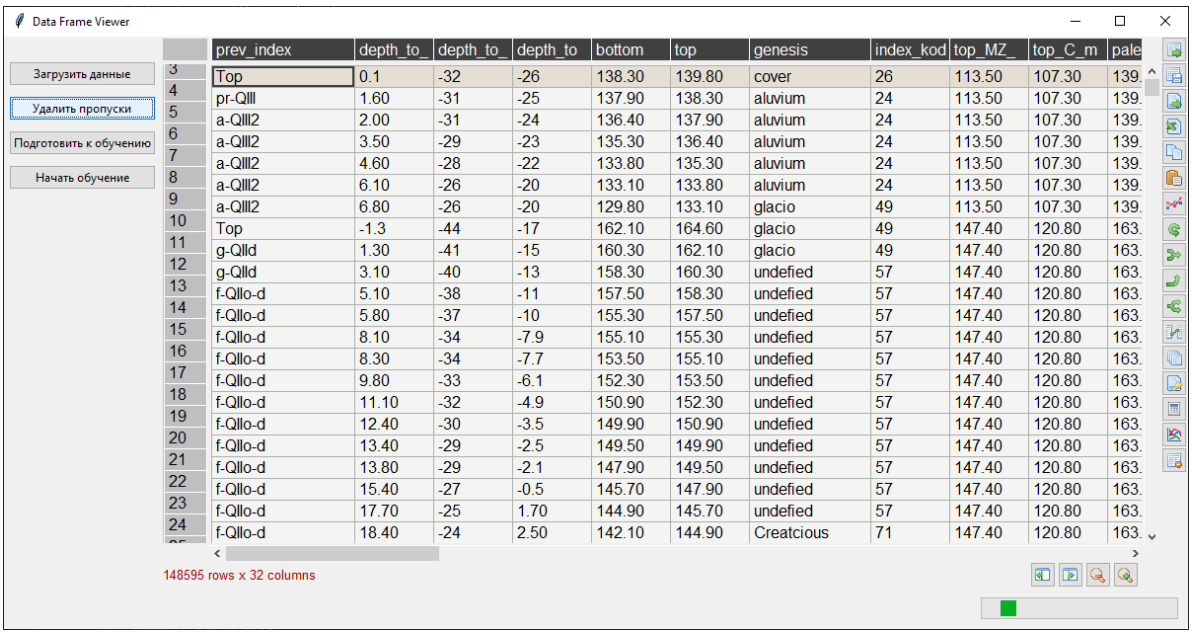
ПИС разработана на языке программирования Python с его мощными библиотеками для научных вычислений (pandas, numpy, sklearn, seaborn, matplotlib, tkinter, pandastable и time). Она выполняет перечисленную выше последовательность преобразований и визуализаций данных, а затем импортирует их в модель машинного обучения Random Forest для непосредственного определения типа грунта на основе предоставленных данных. Random Forest работает с ансамблями деревьев решений и используется для задач

классификации и регрессии. То есть он представляет собой ансамблевый метод, который комбинирует прогнозы нескольких деревьев решений для улучшения общей производительности модели.

Обучение модели является ключевым шагом, позволяющим модели извлекать закономерности и зависимости из имеющихся данных и использовать их для предсказания типов грунта. В процессе обучения модели Random Forest, каждое дерево строится на основе случайной подвыборки данных из обучающей выборки. Для каждого дерева также случайно выбирается подмножество признаков. Это помогает снизить переобучение модели и повысить ее обобщающую способность. Деревья строятся путем разделения данных на основе различных признаков и значений. Признаки и их значения используются для создания условий, по которым данные разделяются на более чистые подгруппы, соответствующие различным типам грунта. Процесс разделения продолжается до достижения определенного критерия остановки, такого как достижение определенной глубины дерева или минимального числа образцов в листьях. Каждое дерево в Random Forest производит свое собственное предсказание типа грунта для каждого образца данных. Затем, для получения окончательного предсказания, модель применяет голосование или усреднение предсказаний от всех деревьев в ансамбле. Это позволяет учесть мнение нескольких деревьев и получить более надежный результат.

Обучение модели Random Forest включает построение ансамбля деревьев и настройку их параметров. Указываем количество деревьев в ансамбле, функцию измерения качества разделения, минимальное число образцов в листьях и бутстрап. Настройка параметров может влиять на производительность и обобщающую способность модели. После завершения обучения модели Random Forest на обучающей выборке, можно провести оценку производительности модели на тестовой выборке для оценивания качества классификации и способности модели обобщать на новые данные.

На рисунке 2 приведен пример графического интерфейса пользователя ПИС при выполнении действия (о чем свидетельствует прогресс-бар в правой нижней части экрана). Он позволяет выполнить: импорт, обработку, визуализацию и экспорт данных; выбрать средства для редактирования и изменения отображения графиков; выбрать средства для обучения моделей машинного обучения.



	prev_index	depth_to_	depth_to_	depth_to	bottom	top	genesis	index_kod	top_MZ_	top_C_m	pale
3	Top	0.1	-32	-26	138.30	139.80	cover	26	113.50	107.30	139.
4	pr-QIII	1.60	-31	-25	137.90	138.30	aluvium	24	113.50	107.30	139.
5	a-QIII2	2.00	-31	-24	136.40	137.90	aluvium	24	113.50	107.30	139.
6	a-QIII2	3.50	-29	-23	135.30	136.40	aluvium	24	113.50	107.30	139.
7	a-QIII2	4.60	-28	-22	133.80	135.30	aluvium	24	113.50	107.30	139.
8	a-QIII2	6.10	-26	-20	133.10	133.80	aluvium	24	113.50	107.30	139.
9	a-QIII2	6.80	-26	-20	129.80	133.10	glacio	49	113.50	107.30	139.
10	Top	-1.3	-44	-17	162.10	164.60	glacio	49	147.40	120.80	163.
11	g-QIId	1.30	-41	-15	160.30	162.10	glacio	49	147.40	120.80	163.
12	g-QIId	3.10	-40	-13	158.30	160.30	undefied	57	147.40	120.80	163.
13	f-QIIo-d	5.10	-38	-11	157.50	158.30	undefied	57	147.40	120.80	163.
14	f-QIIo-d	5.80	-37	-10	155.30	157.50	undefied	57	147.40	120.80	163.
15	f-QIIo-d	8.10	-34	-7.9	155.10	155.30	undefied	57	147.40	120.80	163.
16	f-QIIo-d	8.30	-34	-7.7	153.50	155.10	undefied	57	147.40	120.80	163.
17	f-QIIo-d	9.80	-33	-6.1	152.30	153.50	undefied	57	147.40	120.80	163.
18	f-QIIo-d	11.10	-32	-4.9	150.90	152.30	undefied	57	147.40	120.80	163.
19	f-QIIo-d	12.40	-30	-3.5	149.90	150.90	undefied	57	147.40	120.80	163.
20	f-QIIo-d	13.40	-29	-2.5	149.50	149.90	undefied	57	147.40	120.80	163.
21	f-QIIo-d	13.80	-29	-2.1	147.90	149.50	undefied	57	147.40	120.80	163.
22	f-QIIo-d	15.40	-27	-0.5	145.70	147.90	undefied	57	147.40	120.80	163.
23	f-QIIo-d	17.70	-25	1.70	144.90	145.70	undefied	57	147.40	120.80	163.
24	f-QIIo-d	18.40	-24	2.50	142.10	144.90	Creatcious	71	147.40	120.80	163.

Рис. 2. Отображение графического интерфейса пользователя ПИС при выполнении действия.

При работе пользователи активно использует методы визуализации выбранных из таблицы данных в виде графиков прямо в интерфейсе (рис. 3). Существует возможность настройки различных атрибутов графика: написания заголовка, подписи осей, стиля линий.

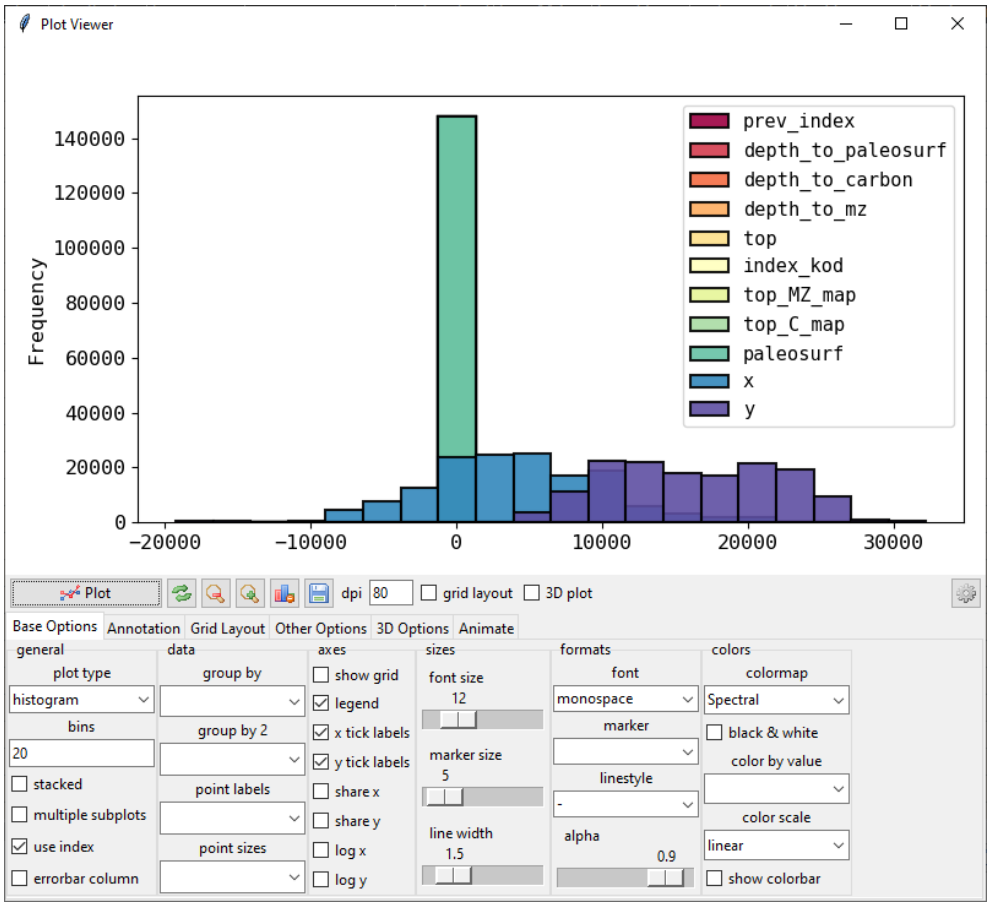


Рис. 3. Пример отображения данных и настройки его параметров.

После загрузки данных из .csv файла проводится предварительный анализ данных. Структура данных состоит из 32 столбцов (то есть 32 признаков для классификации типов грунта) и 148595 строк (то есть данных для 148595 различных экземпляров проб грунта).

Далее было произведено изучение распределения значений в каждом признаке, используя статистические метрики (минимум и максимум, среднее значение, стандартное отклонение). Также были построены гистограммы распределения значений для визуального оценивания форм распределения и выявления возможных выбросы или аномалии.

При анализе данных были выявлены пропущенных значений. Далее, используя библиотеку pandas, были идентифицированы эти пропущенные значения и к каждому была применена соответствующая стратегия обработки (в некоторых случаях пропущенные значения были заполнены неизвестными или полностью удалены некорректные данные).

Затем был выполнен анализ, включающий вычисление статистических метрик, визуализацию данных и проверку корреляции между признаками и целевой переменной. В результате были выбраны наиболее значимые признаки для классификации грунтов, то есть сформировано подмножество признаков, которые будут использоваться для обучения модели. Визуализации boxplot полезна для сравнения распределений разных признаков и выявления потенциальных выбросов или необычных значений, что важно при принятии решений о предобработке данных или выборе соответствующих статистических методов (рис. 4).

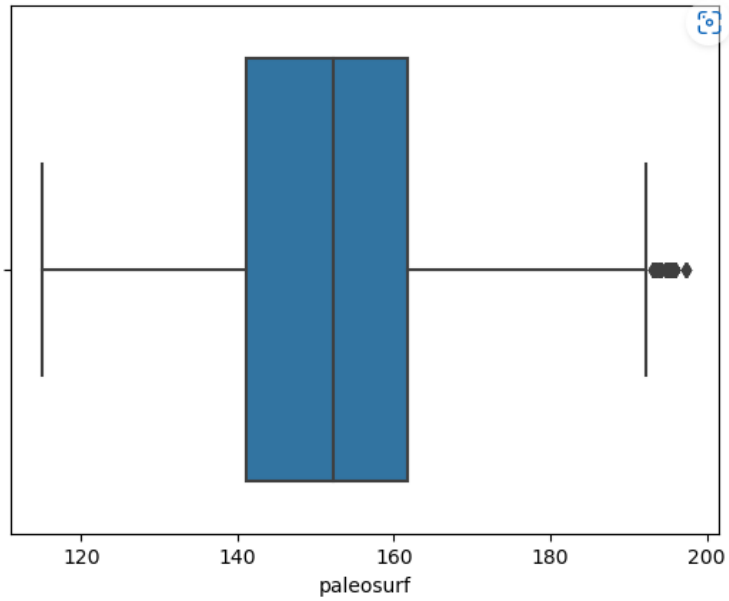


Рис. 4. Пример применения boxplot на один из признаков.

Гистограммы распределения данных использовались для визуализации частоты появления значений в различных интервалах. Она позволяет наглядно представить форму распределения данных и выявить ее особенности (моды, наличие выбросов и прочие). Это

помогает принять решения о преобразовании данных, удалении выбросов или выборе соответствующих статистических методов для дальнейшего анализа и моделирования.

Для анализа данных и подготовки их к обучению модели была использована корреляционная матрица (рис. 5). Она позволяет определить степень линейной связи между признаками и выявить сильно коррелирующие признаки. Это важно для исключения дублирования информации и снижения влияния мультиколлинеарности на процесс обучения модели. После анализа матрицы корреляции можно принимать решение о выборе наиболее значимых признаков и использовать их для обучения модели.

	depth_to_paleosurf	depth_to_carbon	depth_to_mz	top	top_MZ_map	top_C_map	paleosurf	x	y
depth_to_paleosurf	1.000000	0.679863	0.847739	-0.734667	-0.122145	-0.011433	-0.095446	-0.084972	-0.179697
depth_to_carbon	0.679863	1.000000	0.757827	-0.881950	-0.452814	0.374055	-0.626097	0.046105	-0.412643
depth_to_mz	0.847739	0.757827	1.000000	-0.755472	0.025480	0.112045	-0.275554	-0.072698	-0.382344
top	-0.734667	-0.881950	-0.755472	1.000000	0.635716	0.107225	0.745448	-0.084348	0.461933
top_MZ_map	-0.122145	-0.452814	0.025480	0.635716	1.000000	0.295630	0.812772	-0.214329	0.254355
top_C_map	-0.011433	0.374055	0.112045	0.107225	0.295630	1.000000	0.146069	-0.068706	0.038490
paleosurf	-0.095446	-0.626097	-0.275554	0.745448	0.812772	0.146069	1.000000	-0.207281	0.501189
x	-0.084972	0.046105	-0.072698	-0.084348	-0.214329	-0.068706	-0.207281	1.000000	-0.029307
y	-0.179697	-0.412643	-0.382344	0.461933	0.254355	0.038490	0.501189	-0.029307	1.000000

Рис. 5. Матрица корреляции.

Далее в четыре этапа происходит подготовка данных к обучению.

На первом этапе с помощью метода get_dummies было проведено преобразование категориальных переменных в числовые значения для подготовки данных к классификации с использованием алгоритма Random Forest (так как при наличии категориальными данными не получится их использовать в моделях машинного обучения (линейной регрессии или дереве принятия решений)).

На втором этапе было выполнено масштабирование признаков, то есть приведение значений признаков к одному и тому же диапазону или единой шкале (иначе признаки с большими значениями в своих диапазонах будут иметь большее влияние на модель, чем признаки с меньшими значениями и/или с меньшими диапазонами). Для масштабирования признаков использовались методы стандартизации (z-масштабирование). То есть при стандартизации признаков вычисляется среднее значение и стандартное отклонение каждого признака. Затем каждое значение признака вычитается из среднего значения и делится на стандартное отклонение. Это приводит к тому, что каждый признак имеет среднее значение равное нулю и стандартное отклонение равное единице.

На третьем этапе с помощью функции concat колонки объединяются и подготавливаются датафреймы к обучению.

На четвертом этапе проводится разделение данных на обучающую и тестовую выборки, необходимо для оценивания производительности модели машинного обучения и проверки ее способности к обобщению на новых данных. Пусть обучающая выборка составляет 70% от исходного набора данных, а тестовая – 30%.

После всех проведенных подготовительных работ переходим к обучению модели на обучающей выборке, что позволит ей извлекать закономерности и зависимости из имеющихся данных и использовать их для предсказания типов грунта.

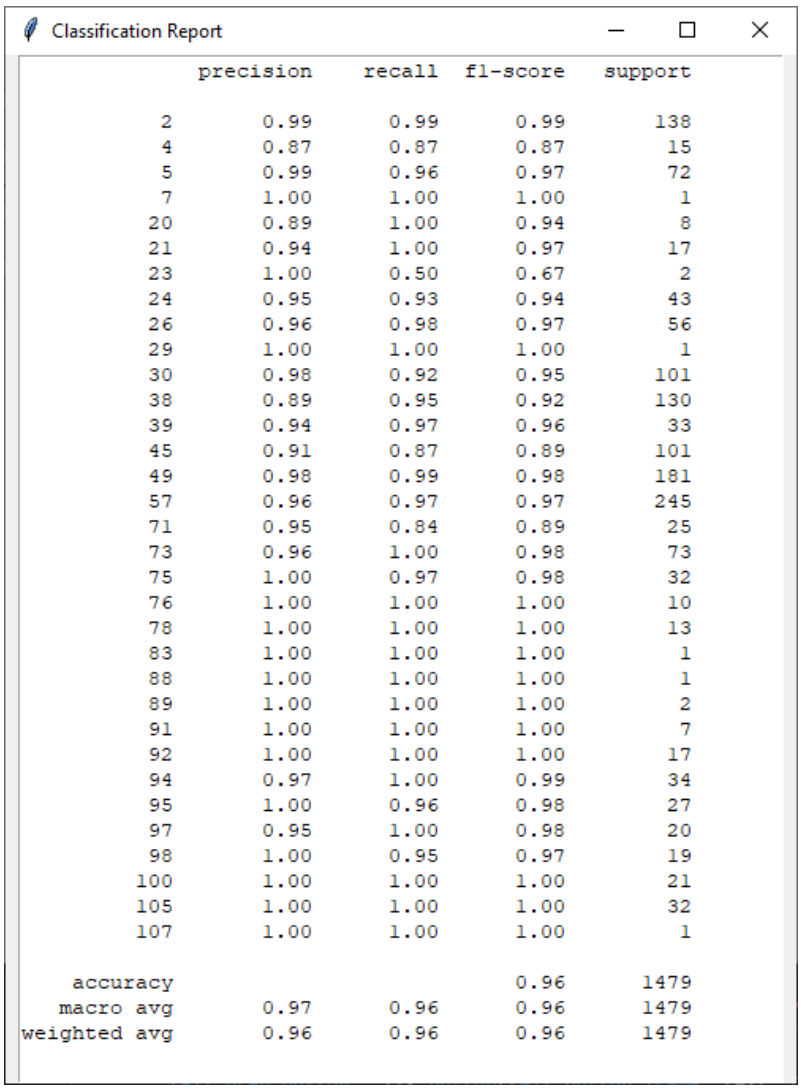
В процессе обучения модели Random Forest, каждое дерево строится на основе случайной подвыборки данных из обучающей выборки. Для каждого дерева также случайно выбирается подмножество признаков (это снижает переобучение модели и повышает ее обобщающую способность). Деревья строятся путем разделения данных на основе различных признаков и значений. Признаки и их значения используются для создания условий, по которым данные разделяются на более чистые подгруппы, соответствующие различным типам грунта. Процесс разделения продолжается до достижения определенного критерия остановки, такого как достижение определенной глубины дерева или минимального числа образцов в листьях.

Каждое дерево в Random Forest производит свое собственное предсказание типа грунта для каждого образца данных. Затем, для получения окончательного предсказания, модель применяет голосование или усреднение предсказаний от всех деревьев в ансамбле. Это позволяет учесть мнение нескольких деревьев и получить более надежный результат.

Обучение модели Random Forest включает построение ансамбля деревьев и настройку их параметров. Изначально указывается количество деревьев в ансамбле, функции измерения качества разделения, минимальное число образцов в листьях и бутстрап. Настройка параметров может влиять на производительность и обобщающую способность модели. После завершения обучения модели на обучающей выборке, можно провести оценку производительности модели на тестовой выборке.

После завершения обучения модели выполняется анализ результатов обучения, включающий оценку точности модели на тестовых данных, генерацию отчета о классификации и другие метрики оценивания модели (рис. 6).

Реализация интерфейса ПИС осуществлялась с использованием библиотеки Tkinter для создания интуитивно понятного графического пользовательского интерфейса (GUI) и PandasTable для отображения и взаимодействия с табличными данными.



	precision	recall	f1-score	support
2	0.99	0.99	0.99	138
4	0.87	0.87	0.87	15
5	0.99	0.96	0.97	72
7	1.00	1.00	1.00	1
20	0.89	1.00	0.94	8
21	0.94	1.00	0.97	17
23	1.00	0.50	0.67	2
24	0.95	0.93	0.94	43
26	0.96	0.98	0.97	56
29	1.00	1.00	1.00	1
30	0.98	0.92	0.95	101
38	0.89	0.95	0.92	130
39	0.94	0.97	0.96	33
45	0.91	0.87	0.89	101
49	0.98	0.99	0.98	181
57	0.96	0.97	0.97	245
71	0.95	0.84	0.89	25
73	0.96	1.00	0.98	73
75	1.00	0.97	0.98	32
76	1.00	1.00	1.00	10
78	1.00	1.00	1.00	13
83	1.00	1.00	1.00	1
88	1.00	1.00	1.00	1
89	1.00	1.00	1.00	2
91	1.00	1.00	1.00	7
92	1.00	1.00	1.00	17
94	0.97	1.00	0.99	34
95	1.00	0.96	0.98	27
97	0.95	1.00	0.98	20
98	1.00	0.95	0.97	19
100	1.00	1.00	1.00	21
105	1.00	1.00	1.00	32
107	1.00	1.00	1.00	1
accuracy			0.96	1479
macro avg	0.97	0.96	0.96	1479
weighted avg	0.96	0.96	0.96	1479

Рис. 6. Пример квалификационного отчета.

Заключение. В результате проведенного исследования разработано программное обеспечение (ПИС) для анализа и классификации данных проб грунтов, а также практического определению категории грунта на основе имеющихся о них данных.

СПИСОК ЛИТЕРАТУРЫ

1. Специалисты Газпром нефти научили программу исследовать образцы керна по фото [Электронный ресурс]. – Режим доступа: https://neftegaz.ru/news/standarts/637475-spetsialisty-gazprom-nefti-nauchili-programmu-issledovat-obraztsy-kerna-po-foto/ (дата обращения: 03.09.2023).
2. Комплексный анализ и определение фильтрационно-емкостных свойств геологического образца [Электронный ресурс]. – Режим доступа: https://xn--b1aghfftcbpg0bw.xn--p1ai/ (дата обращения: 11.10.2023).

3. Документация Python 3.12.0 [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/ (дата обращения: 11.10.2023).
4. Груздев А. В. Прогнозное моделирование в IBM SPSS Statistics, R и Python: метод деревьев решений и случайный лес. – М.: ДМК Пресс, 2018. – 642 с.

РАССАДИН А. Э.

ОБ ОДНОМ НЕЛИНЕЙНОМ ИНТЕГРО-ДИФФЕРЕНЦИАЛЬНОМ УРАВНЕНИИ

Аннотация. В работе дана иллюстрация взаимного влияния эффектов нелинейности и нелокальности, а именно, найдено точное решение задачи Коши для нелинейного интегро-дифференциального уравнения, обладающего следующим свойством: изменение знака у ограниченного начального условия за конечное время приводит к неограниченному возрастанию по модулю соответствующего ему решения. Получено общее решение задачи Коши для рассматриваемого уравнения, а также продемонстрирован метод расширения таблиц преобразования Лапласа по двум переменным с помощью частных решений этого уравнения.

Ключевые слова: оригинал, изображение, свёртка, функция Бесселя первого рода, модифицированная функция Бесселя, комплексная плоскость.

RASSADIN A. E.

ON A NONLINEAR INTEGRO-DIFFERENTIAL EQUATION

Abstract. The paper illustrates the mutual influence of the effects of nonlinearity and nonlocality, namely, an exact solution of the Cauchy problem for a nonlinear integro-differential equation is found, which has the following property: a change in the sign of a bounded initial condition in finite time leads to an unbounded increase in the modulus of the corresponding solution. A general solution of the Cauchy problem for the equation under consideration is obtained, and a method for expanding the Laplace transform tables by two variables using partial solutions of this equation is demonstrated.

Keywords: pre-image, transform table, convolution, Bessel functions of the first kind, modified Bessel functions, complex plane.

Одним из способов повышения точности описания процессов различной природы является учёт при протекании этих последних нелокальности взаимодействий [1; 2]. Однако комбинация нелинейности математической модели с простейшей нелокальностью в виде пространственного сдвига уже приводит к резкому возрастанию математических трудностей (см., например, [3]), поэтому поиск точно решаемых модельных примеров, обладающих как нелинейностью, так и нелокальностью, приобретает важное значение.

Рассмотрим следующее интегро-дифференциальное уравнение:

$$\frac{\partial u(x,y,t)}{\partial t} + \int_0^x \int_0^y u(x-\xi, y-\eta, t) \cdot u(\xi,\eta,t) \cdot d\xi d\eta = 0, \qquad (1)$$

в котором неизвестная функция $u(x,y,t)$ определена в I-м квадранте: $x \ge 0$, $y \ge 0$.

Снабдим уравнение (1) начальным условием:

$$u(x,y,0) = u_0(x,y), \qquad x \ge 0,\ y \ge 0, \tag{2}$$

тогда для решения задачи Коши (1)-(2) справедлива следующая теорема.

Теорема. *Пусть существуют такие константы* $M_0 > 0$, $h_0 > 0$ *и* $k_0 > 0$, *что в I-м квадранте функция (2) удовлетворяет неравенству:*

$$|u_0(x,y)| \le M_0 \cdot \exp(h_0 x + k_0 y), \tag{3}$$

тогда точное решение задачи Коши (1)-(2) имеет вид:

$$u(x,y,t) = \int_{a-i\infty}^{a+i\infty}\int_{b-i\infty}^{b+i\infty} \frac{U_0(p,q)}{1+t\cdot U_0(p,q)} \cdot \exp(p\cdot x + q\cdot y) \cdot \frac{dpdq}{(2\cdot\pi\cdot i)^2}, \tag{4}$$

где

$$U_0(p,q) = \int_0^{+\infty}\int_0^{+\infty} u_0(x,y)\cdot\exp(-p\cdot x - q\cdot y)\cdot dxdy, \tag{5}$$

а в формуле (4) прямые $\mathrm{Re}\, p = a$ *и* $\mathrm{Re}\, q = b$ *выбраны так, чтобы все особенности функции* $U_0(p,q)/(1+t\cdot U_0(p,q))$ *в комплексных плоскостях* p *и* q *соответственно оставались слева от них.*

Доказательство. Нелинейный член в уравнении (1) представляет собой двойную свёртку лапласовского типа по обеим пространственным переменным, поэтому будем искать его решение с помощью двойного преобразования Лапласа — и по x, и по y.

В рамках этого формализма переход от оригинала $u(x,y,t)$ к изображению $U(p,q,t)$ имеет вид [4]:

$$U(p,q,t) = \int_0^{+\infty}\int_0^{+\infty} u(x,y,t)\cdot\exp(-p\cdot x - q\cdot y)\cdot dxdy, \tag{6}$$

а исходное интегро-дифференциальное уравнение (1) для оригинала трансформируется в обыкновенное дифференциальное уравнение для изображения (6):

$$\frac{\partial U(p,q,t)}{\partial t} + U^2(p,q,t) = 0, \qquad U(p,q,0) = U_0(p,q). \tag{7}$$

Начальным условием для уравнения (7) является функция (5), причём условие (3) является достаточным условием её существования в области $\mathrm{Re}\, p > h_0$ и $\mathrm{Re}\, q > k_0$ [4].

Задача Коши (7) имеет следующее точное решение:

$$U(p,q,t) = \frac{U_0(p,q)}{1+t\cdot U_0(p,q)}, \tag{8}$$

а формула (4) представляет собой обращение двумерного преобразования Лапласа (6).

Однако формула (4), выражающая общее решение задачи Коши (1)-(2), неудобна для практического применения, поскольку для её использования надо проводить интегрирование в комплексном пространстве C^2. Гораздо проще находить точные решения задачи Коши (1)-(2) в явном виде следующим образом: вычислить по формуле (5) изображение начального условия (2), а затем проверить по таблице двумерных преобразований Лапласа, приведённой в книге [4], соответствует ли изображение, найденное по формуле (8), какой-либо функции из этой таблицы.

Продемонстрируем применение этого приёма на примере.

Пусть начальное условие для уравнения (1) имеет вид:

$$u_0(x,y) = A_0 \cdot J_0(2\sqrt{a_0 xy}), \qquad x \ge 0,\ y \ge 0, \quad a_0 > 0, \tag{9}$$

где $J_0(z)$ — это функция Бесселя, тогда интеграл (5) от функции (9) легко вычисляется с помощью известного разложения функции $J_0(z)$ в степенной ряд:

$$U_0(p,q) = \frac{A_0}{pq + a_0}. \tag{10}$$

Подставляя выражение (10) в формулу (8), получим для изображения решения:

$$U(p,q,t) = \frac{A_0}{pq + a_0 + A_0 t}. \tag{11}$$

Функция (11) получается из функции (10) заменой $a_0 \to a_0 + A_0 t$, поэтому вследствие единственности обратного преобразования Лапласа по переменным p и q [4] оригинал для изображения (11) получается из функции (9) той же заменой:

$$u(x,y,t) = A_0 \cdot J_0(2\sqrt{(a_0 + A_0 t)xy}). \tag{12}$$

При $A_0 > 0$ функция (12) обладает колебательным характером. На рисунке 1 приведён график функции (9) при $A_0 = 1$ и $a_0 = 1$. С течением времени характерная «длина волны» уменьшается — для этого достаточно сравнить рис. 1 с рис. 2, на котором представлен график функции (12) при тех же параметрах $A_0 = 1$ и $a_0 = 1$, но в момент времени $t = 3$. Если же $A_0 < 0$, то при $t_c = a_0 / |A_0|$ в решении (12) происходит переход от колебательного режима к режиму с неограниченно возрастающей по модулю функцией $u(x,y,t)$. С формальной точки зрения этот переход выражается в замене функции Бесселя $J_0(z)$ на модифицированную функцию Бесселя $I_0(z)$ при $t > t_c$. Рисунки 3 и 4 иллюстрируют этот эффект. В этом случае при $t \to t_c - 0$ характерная «длина волны» увеличивается, что

приводит к уплощению графика функции (12) до тех пор, пока при $t = t_c$ он не становится плоскостью $u(x, y, t_c) = -|A_0|$.

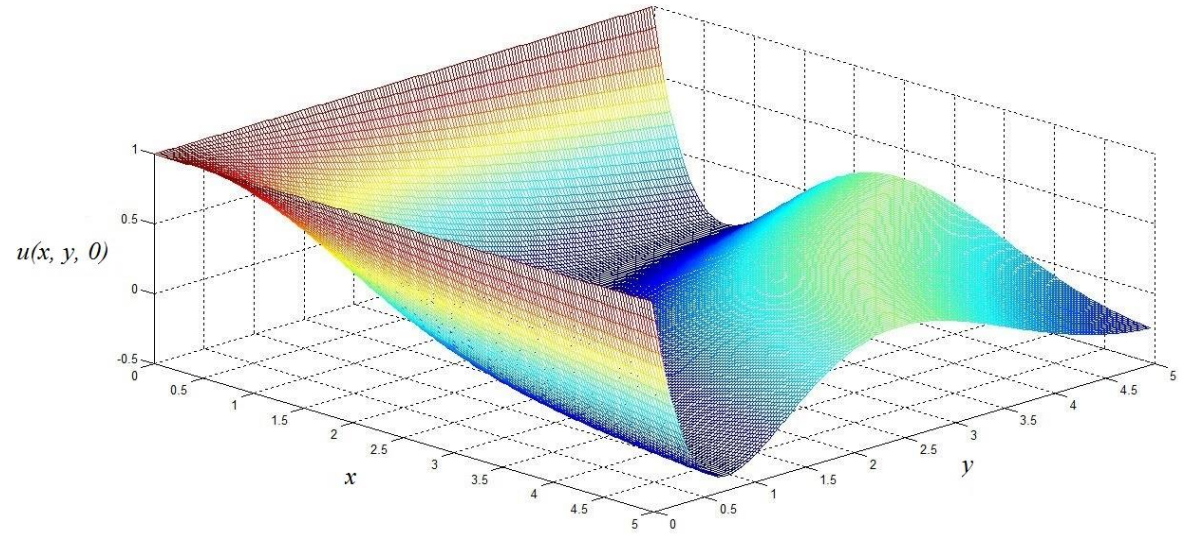


Рис. 1. График начального условия $u(x, y, 0)$ уравнения (1) при $A_0 = 1$ и $a_0 = 1$.

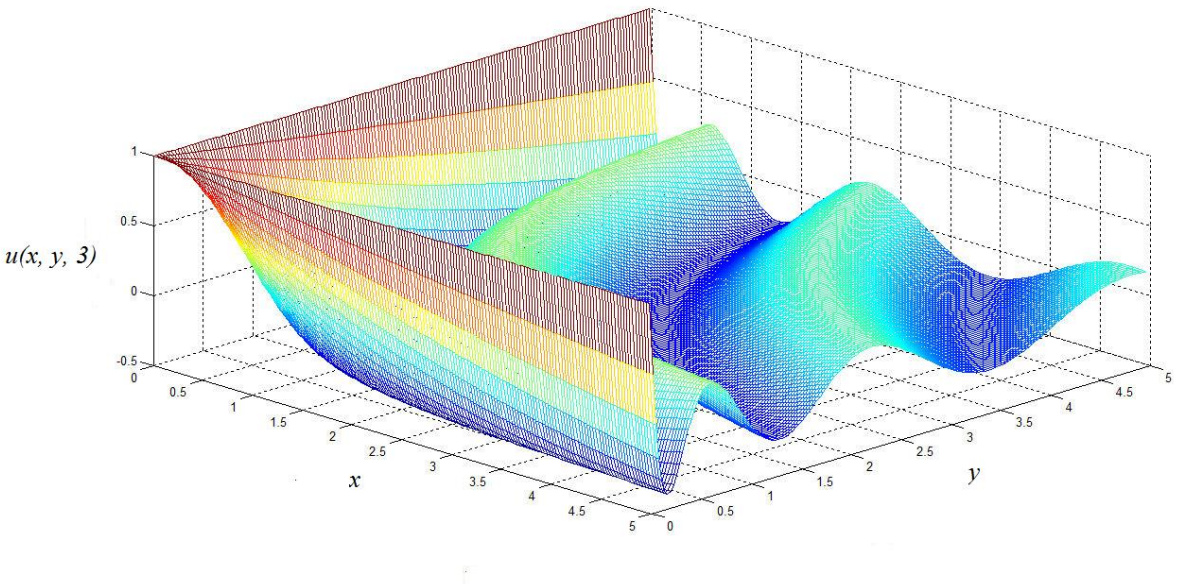


Рис. 2. График решения задачи Коши (1)-(2) при $t = 3$, $A_0 = 1$ и $a_0 = 1$.

Таким образом, при переходе в решении (12) параметром A_0 нулевого значения из ограниченного начального условия (9) за конечное время развивается неограниченное решение, то есть система (1) в своём поведении демонстрирует некоторое сходство с фазовым переходом второго рода [5] или с бифуркацией Андронова-Хопфа [6].

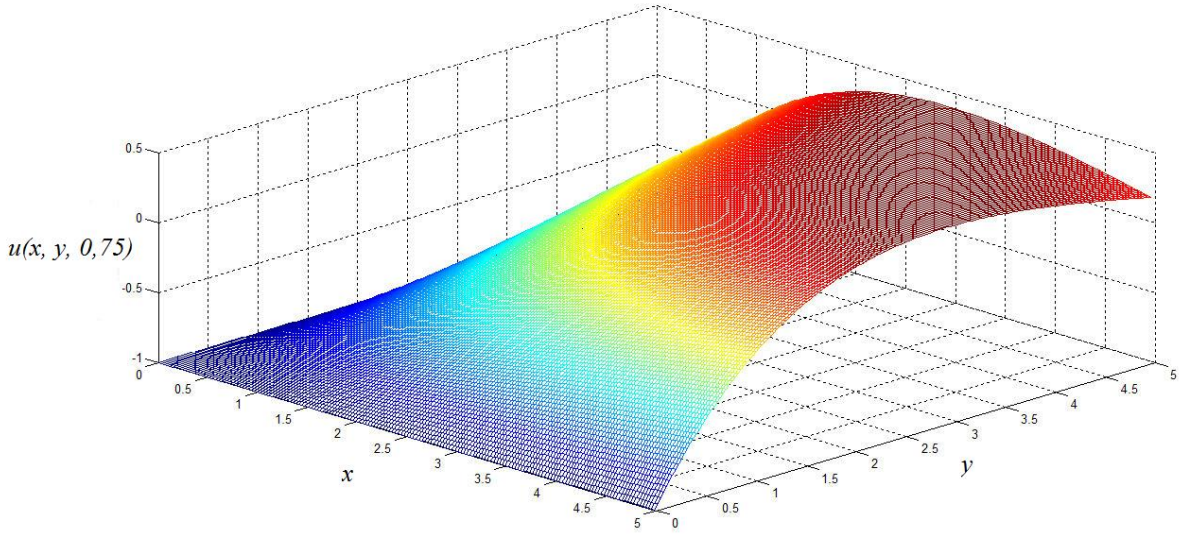


Рис. 3. График решения задачи Коши (1)-(2) при $t = 0{,}75$ и $A_0 = -1$ и $a_0 = 1$.

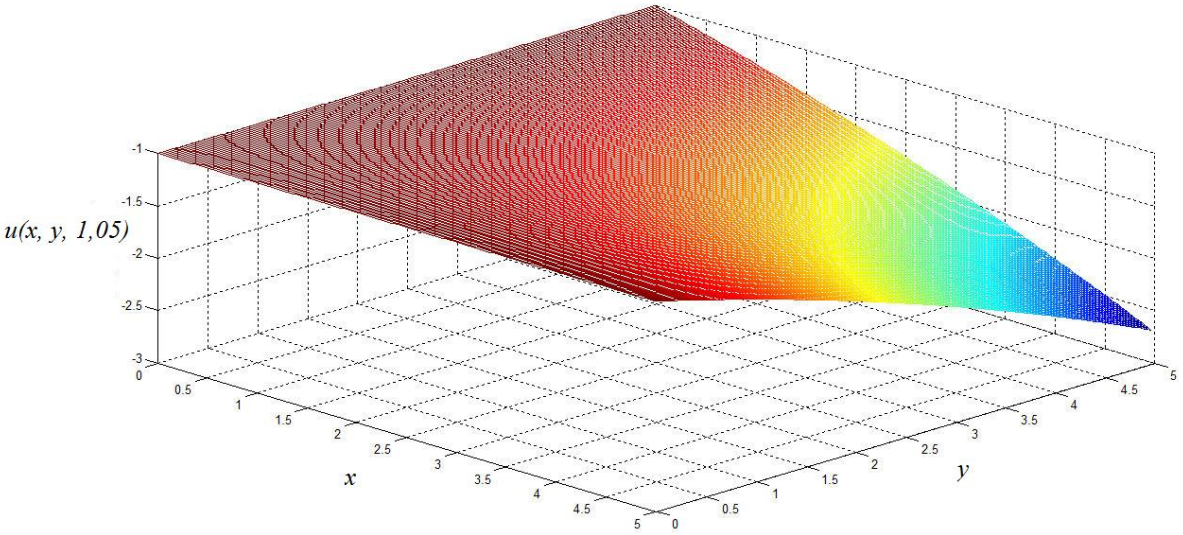


Рис. 4. График решения задачи Коши (1)-(2) при $t = 1{,}05$ и $A_0 = -1$, $a_0 = 1$.

В заключение необходимо отметить, что если точное решение $u(x,y,t)$ задачи Коши (1)-(2) из каких-либо соображений известно, то двойная свёртка лапласовского типа начального условия (2) самого с собой может быть вычислена по формуле:

$$\int_0^x \int_0^y u_0(x-\xi, y-\eta)\, u_0(\xi,\eta) \cdot d\xi d\eta = -\frac{\partial u(x,y,t)}{\partial t}\bigg|_{t=0}, \qquad (13)$$

а при известной функции (5) применение двойного преобразования Лапласа по переменным x и y к левой части формулы (13) может привести к пополнению таблиц преобразований Лапласа по двум переменным.

В частности, подставляя в соотношение (13) функции (9) и (12), найдём:

$$\int_0^x \int_0^y J_0(2\sqrt{a_0(x-\xi)(y-\eta)}) \cdot J_0(2\sqrt{a_0\xi\eta}) \cdot d\xi d\eta = \sqrt{\frac{xy}{a_0}} J_1(2\sqrt{a_0 xy}) . \qquad (14)$$

Наконец, с помощью выражения (10), взятого при $A_0 = 1$, из формулы (14) без вычислений можно получить, что:

$$\int_0^{+\infty} \int_0^{+\infty} \sqrt{\frac{xy}{a_0}} \cdot J_1(2\sqrt{a_0 xy}) \cdot \exp(-p \cdot x - q \cdot y) \cdot dxdy = \frac{1}{(pq + a_0)^2} .$$

СПИСОК ЛИТЕРАТУРЫ

1. Ефимов Г. В. Нелокальные взаимодействия квантованных полей. – М.: Наука, 1977. – 367 с.
2. Учайкин В. В. Метод дробных производных. – Ульяновск: Артишок, 2008. – 512 с.
3. Алёшин С. В., Глызин С. Д., Кащенко С. А. Особенности динамики уравнения Колмогорова-Петровского-Пискунова с отклонением по пространственной переменной // Моделирование и анализ информационных систем. – 2015. – Т. 22, № 5. – С. 609–628.
4. Диткин В. А., Прудников А. П. Операционное исчисление по двум переменным и его приложения. – М.: ГИФМЛ, 1958. – 179 с.
5. Ландау Л. Д., Лифшиц Е. М. Статистическая физика. Часть 1. – М.: Наука, 1976. – 584 с.
6. Шильников Л. П., Шильников А. Л., Тураев Д. В., Чуа Л. Методы качественной теории в нелинейной динамике. Часть 2 / пер. с англ. В. А. Осотовой. — М.; Ижевск: НИЦ «РХД», ИКИ, 2009. – 548 с.

ГРИГОРЬЕВ А. О., ФИРСОВА С. А.

РАЗРАБОТКА МОДУЛЯ ПРОГРАММНО-ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ИССЛЕДОВАНИЯ ЗАВИСИМОСТЕЙ МЕЖДУ ПОКАЗАТЕЛЯМИ ВАРИАБЕЛЬНОСТИ СЕРДЕЧНОГО РИТМА СПОРТСМЕНОВ МЕТОДОМ ФАКТОРНОГО АНАЛИЗА

Аннотация. В статье приводится описание разработанного авторами модуля программно-информационной системы, предназначенного для исследования зависимостей между показателями вариабельности сердечного ритма спортсменов методом факторного анализа. Для проведения факторного анализа используется метод главных компонент, а также метод ортогонального вращения в совокупности с тестом сферичности Бартлетта и методом Кайзера. Результатами анализа являются матрицы собственных значений факторов, матрицы факторных нагрузок и кольцевые диаграммы значимых факторов.

Ключевые слова: программно-информационная система, показатель вариабельности сердечного ритма, факторный анализ, тест сферичности Бартлетта, критерий Кайзера, матрица факторных нагрузок, фреймворк Extreme.Statistics.

GRIGOREV A. O., FIRSOVA S. A.

DEVELOPMENT OF A MODULE OF A SOFTWARE AND INFORMATION SYSTEM FOR THE STUDY OF DEPENDENCIES BETWEEN INDICATORS OF HEART RATE VARIABILITY OF ATHLETES BY FACTOR ANALYSIS

Abstract. The article describes the module of the software and information system developed by the authors and designed to study the dependencies between the indicators of heart rate variability of athletes by factor analysis. To carry out factor analysis, the principal component method is used, as well as the orthogonal rotation method in conjunction with the Bartlett sphericity test and the Kaiser method. The results of the analysis are matrices of eigenvalues of factors, matrices of factor loads and ring diagrams of significant factors.

Keywords: software and information system, cardiovascular system index, factor analysis, Bartlett sphericity test, Kaiser criterion, factor load matrix, Extreme.Statistics framework.

В настоящее время методы факторного анализа широко используются в различных сферах деятельности для анализа больших объёмов данных и выявления скрытых закономерностей. В научной периодике появляется всё больше исследований, посвященных применению методов факторного анализа в спорте, что свидетельствует о значимости и популярности этих методов в данной области. Так, в [1] были определены факторы, от которых в большей степени зависит итоговый результат конькобежцев на дистанции 500 м.

Было установлено, что конькобежцам необходимо уделять больше внимание бегу по виражу для развития максимальной скорости. В статье [2] авторами по результатам проведения факторного анализа методом главных компонент были выделены три главных компоненты дисперсии внешней механической работы, влияющих на общую физическую подготовленность гольфистов. Было определено, что первый главный компонент определяется действием мышц верхних конечностей, второй – действием мышц плечевого пояса, участвующих в тех же движениях, третий компонент дисперсии можно отнести к действию мышц передней части туловища и нижних конечностей.

Также в настоящее появляется всё больше исследований, посвященных изучению влияния показателей сердечно-сосудистой системы на спортивную результативность спортсменов. Так, в [3] были выявлены ключевые показатели вариабельности сердечного ритма и центральной нервной системы, которые представляют значимую информацию для прогнозирования соревновательной деятельности высококвалифицированных спортсменов.

Поскольку все подобные исследования сопряжены с получением и обработкой большого объема данных, было решено разработать модуль программно-информационной системы для исследования зависимостей между показателями сердечно-сосудистой системы спортсменов методом факторного анализа, который позволит выявить взаимосвязи и зависимости между различными показателями сердечно-сосудистой системы спортсменов, понять физиологические особенности и индивидуальные потребности каждого спортсмена, оптимизировать тренировочные программы, учитывая выявленные зависимости.

В современном профессиональном спорте для мониторинга показателей вариабельности сердечного ритма (HRV-показателей) нашли широкое применение измерительные устройства, которые позволяют контролировать эти показатели с максимальной точностью и передавать получаемые данные через Bluetooth® или ANT+ в специализированные приложения для их последующей обработки. Так, в данной статье в качестве исходных данных были взяты ежедневные значения HRV-показателей спортсменов-биатлонистов, которые замерялись с помощью пульсометра Polar H10 и синхронизировались с приложением Polar Flow [4].

Разработанный программный модуль включает в себя разнообразные функциональные возможности для проведения факторного анализа, представленные на диаграмме вариантов использования (см. рис. 1).

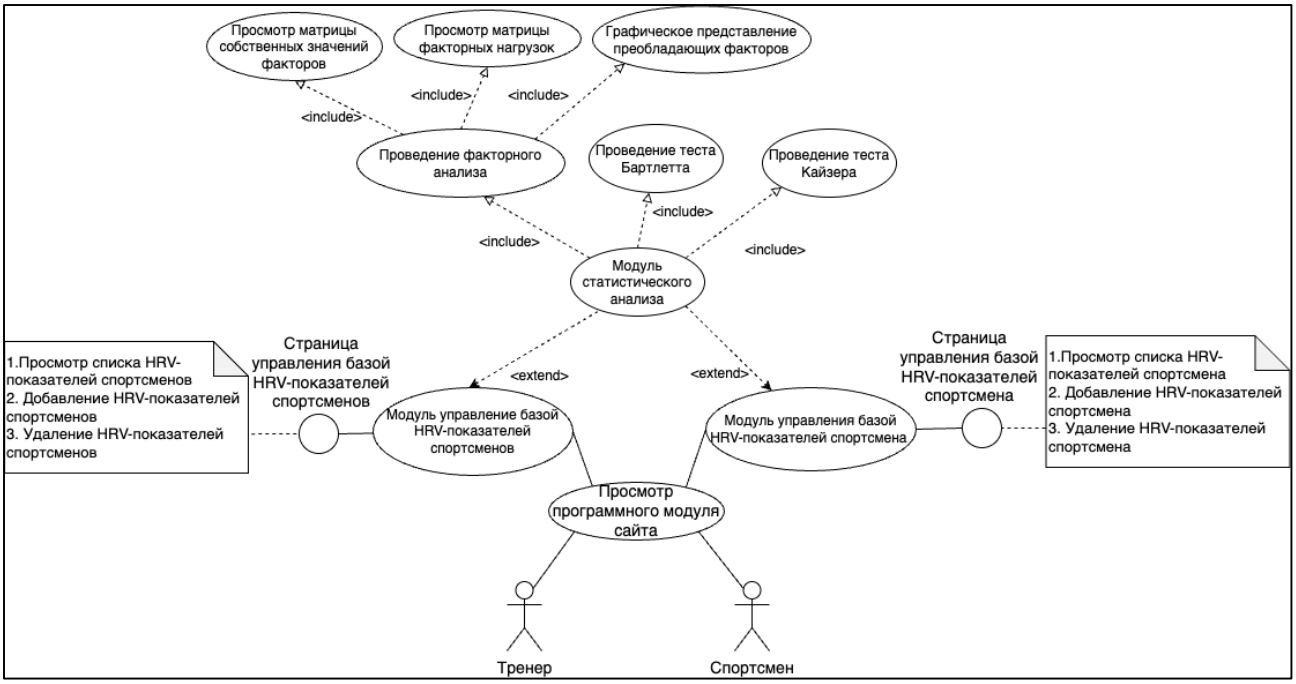


Рис. 1. Диаграмма вариантов использования.

Для реализации программного модуля были использованы следующие инструментальные средства:

- кросс-платформенная интегрированная среда разработки для .NET – Rider;
- язык программирования C# 11 совместно с фреймворком разработки веб-приложений ASP.NET Core и фреймворком Extreme.Statistics для проведения статистического анализа;
- язык программирования для создания интерактивных веб-страниц JavaScript;
- формальный язык описания внешнего вида страницы CSS;
- язык гипертекстовой разметки HTML;
- среда разработки Azure Data Studio для администрирования базы данных MySQL.

В программном модуле была создана база данных, которая хранит как ежедневно измеряемые значения HRV-показателей каждого спортсмена, так и параметры тестов, проводимых методом факторного анализа, а также результаты этих тестов. Структура базы данных приведена на рисунке 2:

Разработанная база данных, которая состоит из 4 таблиц:

– таблица **Пользователи** хранит данные о пользователях системы (спортсменов и тренеров) и содержит поля: идентификатор пользователя, ФИО, дата рождения, пол, спортивный разряд, логин, пароль, статус (спортсмен/тренер);

– таблица **HRV-показатели** содержит данные о показателях спортсмена в тренировочные и выходные дни: идентификатор снятия показателей, дата снятия

показателей, идентификатор спортсмена, HRV-показатели, такие как: READINESS, RMSSD, RR, SDNN, SD1, TP, HF, LF, SI, Heart, Load.

– таблица **Параметры факторного анализа** хранит данные о параметрах факторного анализа: идентификатор анализа, дата проведения факторного анализа и идентификатор спортсмена, данные которого использовались при проведении расчетов;

– таблица **Результаты факторного анализа** содержит информацию о матрицах факторных нагрузок: идентификатор результата, идентификатор факторного анализа, фактор и значения факторных нагрузок по вышеперечисленным показателям сердечно-сосудистой системы.

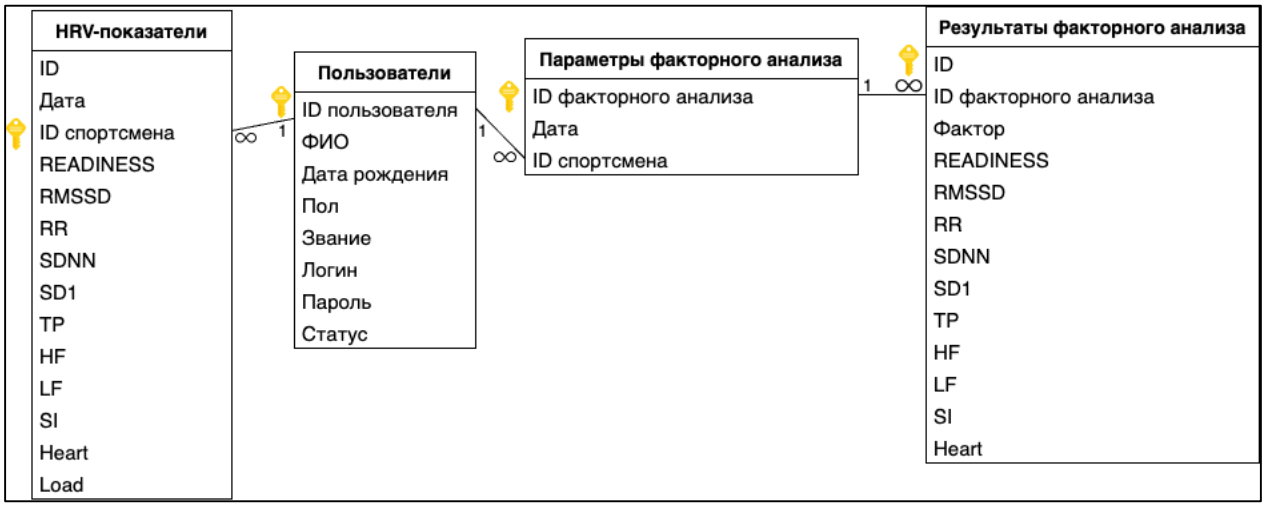


Рис. 2. Схема базы данных.

Исследование данных HRV-показателей спортсменов с помощью метода факторного анализа будет состоять из следующих этапов:

1. загрузка данных;
2. предварительная обработка данных;
3. оценка факторизации набора данных тестом сферичности Бартлетта;
4. выбор оптимального количества факторов методом Кайзера;
5. проведение факторного анализа, которое включает вывод матрицы собственных значений факторов, матрицы факторных нагрузок и кольцевой диаграммы факторных нагрузок.

В качестве примера применения разработанного модуля программно-информационной системы рассмотрим обработку данных, полученных в ходе тренировочного процесса спортсменов-биатлонистов группы «Девушки 15-16 лет, 1-й разряд» ГАУ РМ СШОР по зимним видам спорта.

После авторизации пользователя в программно-информационной системе у спортсмена появляется возможность ввести свои HRV-показатели с помощью заполнения соответствующей формы добавления (см. рис. 3).

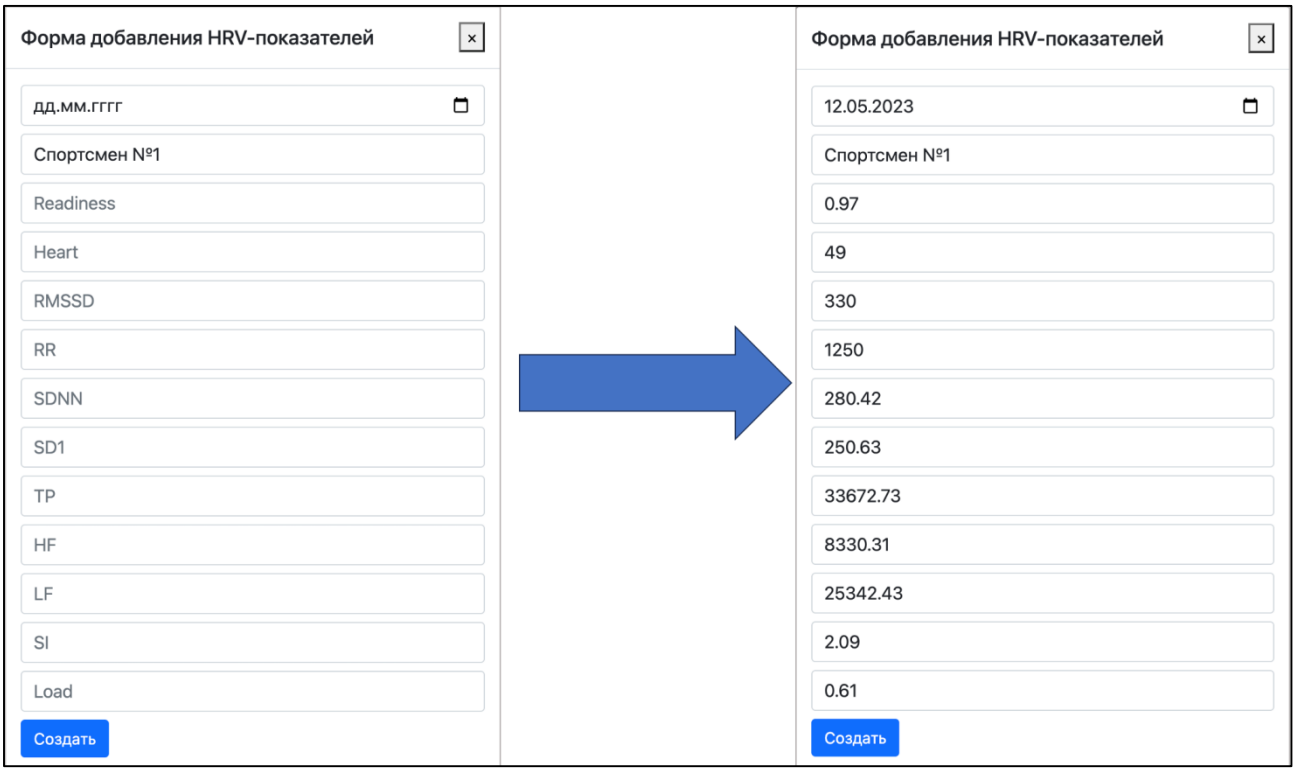


Рис. 3. Форма добавления HRV-показателей.

Также можно увидеть таблицу всех ранее введенных значений HRV-показателей, в которой можно произвести их редактирование или удаление. Внизу таблицы рассчитаны средние значения каждого из показателей за выбранный период времени (см. рис. 4).

HRV-показатели спортсмена № 1

Добавить

Дата	READINESS	RMSSD	RR	SDNN	SD1	TP	HF	LF	SI	Heart	Load	Инструменты
01.05.2023	0,65	63	950,3	60	61	2534,77	1221,94	1312,83	5,32	50	0,44	
02.05.2023	0,63	70	980,35	61	50	4101,65	2000,81	2100,84	5,14	50	0,49	
03.05.2023	0,74	120	1084,25	105,82	85,35	10026,32	6531,95	3494,37	5,36	55	0,47	
04.05.2023	0,70	101	1061,62	96,43	71,29	8503,45	4119,95	4383,5	5,34	53		
05.05.2023	0,73	104	1132,77	83,73	73,99	6488,34	4469,02	2019,32	5,1	50	0,43	
Среднее значение	**0,69**	**91,6**	**1041,86**	**81,396**	**68,326**	**6330,906**	**3668,73**	**2662,17**	**5,252**	**51,6**	**0,4575**	

Рис. 4. Таблица с показателями сердечно-сосудистой системы спортсмена №1.

В нашем исследовании факторный анализ проводился для двух наборов HRV-показателей спортсмена: значений HRV-показателей в тренировочные дни и в выходные дни.

Перед началом проведения факторного анализа требуется произвести оценку потенциала факторизации набора данных. Для этого воспользуемся тестом сферичности Бартлетта, который оценивает взаимосвязи между наблюдаемыми переменными путем оценки их корреляции друг с другом. Этот анализ производится на основе анализа корреляционной матрицы и единичной матрицы. Если результаты анализа не достигают статистической значимости, то это свидетельствует о нецелесообразности проведения факторного анализа.

На рисунке 5 представлен листинг программы для проведения теста сферичности Бартлетта. Создается объект класса BartlettTest, в который передается набор входных данных, после этого необходимо сравнить значение свойства PValue этого объекта с заданным уровнем статистической значимости.

```
Extreme.Mathematics.Matrix<double> da = Extreme.Mathematics.Matrix.Create(data);
var alpha = 0.05;
var bartlettTest = new BartlettTest(data);
var bartlettTestPValue = bartlettTest.PValue;
if(bartlettTestPValue < alpha)
{
    //Тест был статистически значимым, можно выполнять факторный анализ
}
else
{
   //Тест не был статистически значимым, нельзя выполнять факторный анализ
}
```

Рис. 5. Тест сферичности Бартлетта.

Для набора HRV-показателей спортсмена №1, замеренных в тренировочные дни, тест Бартлетта показал, что p-значение равно $2{,}56 \cdot e^{-29}$, что значительно меньше заданного уровня значимости $\alpha=0{,}05$. Таким образом, для такого набора данных можно проводить факторный анализ.

Далее необходимо выбрать оптимальное количество факторов, например, с помощью критерия Кайзера. Нужно отобрать только факторы с собственными значениями, большими 1, поскольку, если фактор не выделяет дисперсию, эквивалентную, по крайней мере, дисперсии одной переменной, то он опускается.

На рисунке 6 представлена реализация метода Кайзера. Для этого создается объект класса FactorAnalysis с числом факторов, равным исходному количеству HRV-показателей, после чего рассчитывается матрица собственных значений факторов и находится число факторов, у которых собственное значение больше 1. Также рисунке 6 показана полученная

матрица собственных значений факторов, по которой мы можем заметить, что у факторов №1 и №2 собственные значения больше 1. Таким образом, оптимальное количество факторов для проведения факторного анализа равно 2.

```
var numberFactors = 0;
var numberIndicators = 10;

var analysis = new FactorAnalysis(data)
{
    NumberOfFactors = numberIndicators
};
analysis.Fit();

var eigenValuesVector = analysis.Eigenvalues;

var numFactors = eigenValuesVector.Count
(eigenvalue => eigenvalue > 1.0);
```

Матрица собственных значений факторов

Фактор №	Собственное значение
1	5,265454
2	1,565738
3	0,697357
4	0,569685
5	0,282659
6	0,094884
7	0,055493
8	0,020513
9	0,002064
10	-0,000016

Рис. 6. Метод Кайзера.

Проведем факторный анализ со следующими параметрами: метод факторного анализа – PrincipalComponents, метод вращения факторов – Varimax, число факторов – numFactors. После этого получаем матрицы дисперсии, объясняемой каждым из факторов varianceExplained, доли общей дисперсии, которые объясняют факторы cumulativeVarianceExplained, а также матрицу нагрузок факторов после вращения rotatedLoadingsMatrix (см. рис. 7).

```
var analysis = new FactorAnalysis(da)
{
    RotationMethod = FactorRotationMethod.Varimax,
    ExtractionMethod = FactorExtractionMethod.PrincipalComponents,
    NumberOfFactors = numFactors,
};
analysis.Fit();

var varianceExplained = analysis.VarianceExplained;
var cumulativeVarianceExplained = analysis.CumulativeVarianceExplained;
var rotatedLoadingsMatrix = analysis.RotatedLoadingsMatrix;

//передача значений на клиентскую часть
```

Рис. 7. Проведение факторного анализа.

Результат проведенных вычислений представлен на рисунке 8. Также имеется возможность представления этих вычислений в виде кольцевой диаграммы, на которой для каждого фактора выделены HRV-показатели, имеющие значения в матрице факторных нагрузок, превышающие по модулю 0,7.

Матрица факторных нагрузок (тренировочные дни)

Переменная	Фактор №1	Фактор №2
READINESS	0,024	**0,862**
RMSSD	**0,765**	-0,200
RR	0,677	0,548
SDNN	-0,081	-0,673
SD1	**0,738**	0,436
TP	0,245	**0,736**
HF	**0,895**	0,282
LF	**0,928**	0,193
SI	0,342	**0,737**
Heart	**-0,790**	-0,332
Доля общей дисперсии	**0,536**	**0,173**

Таким образом, 2 общих фактора объясняют нам 70,96% вариации результатов HRV-показателей спортсмена №1

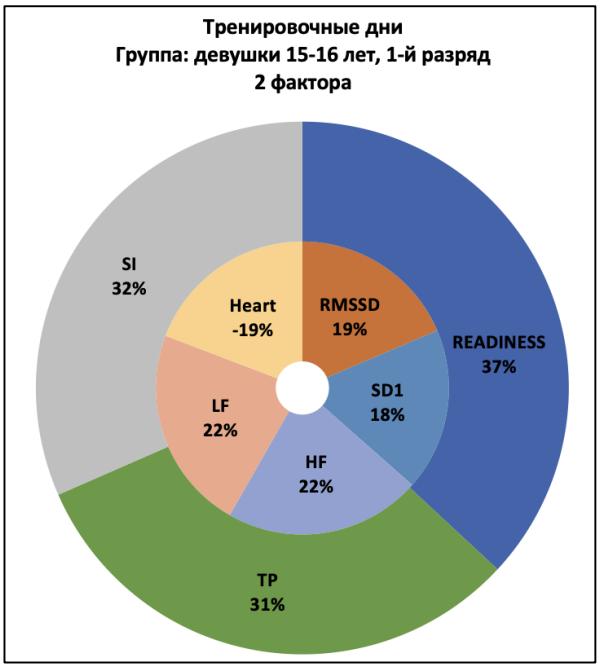


Рис. 8. Матрица факторных нагрузок спортсмена №1 в тренировочные дни.

Аналогично проведем факторный анализ для спортсмена №1 в выходные дни. На рисунке 9 представлена таблица собственных значений факторов, из которой следует, что оптимальное число факторов будет равно 3.

Матрица собственных значений факторов

Фактор №	Собственное значение
1	**4,319390**
2	**2,051849**
3	**1,208351**
4	0,851121
5	0,373689
6	0,209789
7	0,141400
8	0,033798
9	0,016250
10	-0,000025

Рис. 9. Матрица собственных значений факторов спортсмена №1 в выходные дни.

Матрица факторных нагрузок и соответствующая ей кольцевая диаграмма представлена на рисунке 10.

Матрица факторных нагрузок (выходные дни)

Переменная	Фактор №1	Фактор №2	Фактор №3
READINESS	-0,116	**0,747**	0,225
RMSSD	**0,871**	-0,184	-0,116
RR	**0,804**	-0,158	0,461
SDNN	-0,053	**-0,759**	0,153
SD1	0,403	0,684	0,168
TP	0,266	**0,849**	-0,192
HF	**0,855**	0,309	0,043
LF	**0,822**	0,404	-0,049
SI	0,097	0,000	**0,976**
Heart	**-0,762**	-0,324	-0,215
Доля общей дисперсии	**0,437**	**0,217**	**0,123**

Таким образом, 3 общих фактора объясняют нам 77,65% вариации результатов HRV-показателей спортсмена №1

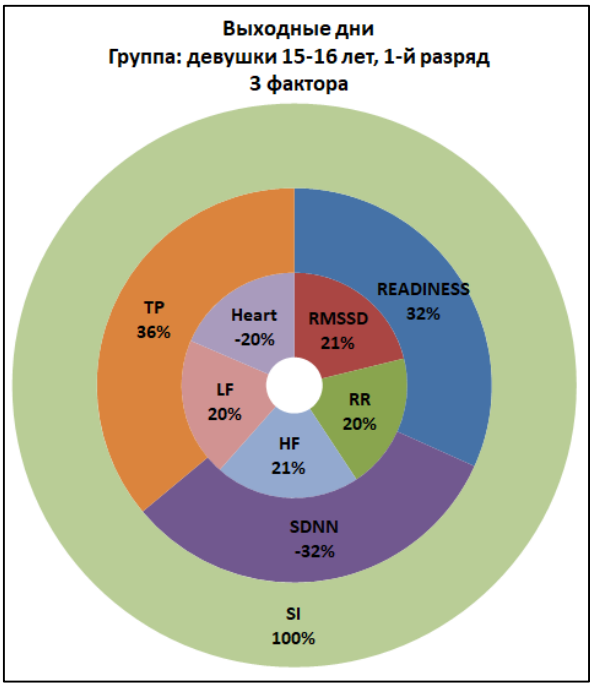


Рис. 10. Диаграмма преобладающих факторов у спортсмена №1 в выходные дни.

Для тренера в программно-информационной системе предусмотрена возможность сравнения результатов проведения факторного анализа для различных групп спортсменов, например, имеющих одинаковый спортивный разряд в разных возрастных группах или имеющих разные спортивные разряды в одном возрасте.

Так, на рисунке 11 представлены матрица факторных нагрузок и кольцевая диаграмма для спортсмена №2 из группы «Девушки 17-18 лет, КМС» в тренировочные дни, а на рисунке 12 – в выходные дни.

Матрица факторных нагрузок (тренировочные дни)

Переменная	Фактор №1	Фактор №2	Фактор №3
READINESS	0,758	0,588	-0,026
RMSSD	0,901	-0,044	-0,343
RR	0,703	-0,162	0,648
SDNN	-0,019	-0,978	0,105
SD1	0,953	0,210	0,051
TP	0,082	0,973	-0,119
HF	0,758	-0,146	0,612
LF	0,953	0,211	0,051
SI	-0,094	-0,128	0,985
Heart	-0,746	0,035	-0,278
Доля общей дисперсии	**0,515**	**0,276**	**0,133**

Таким образом, 3 общих фактора объясняют нам 92,39% вариации результатов HRV-показателей спортсмена №2

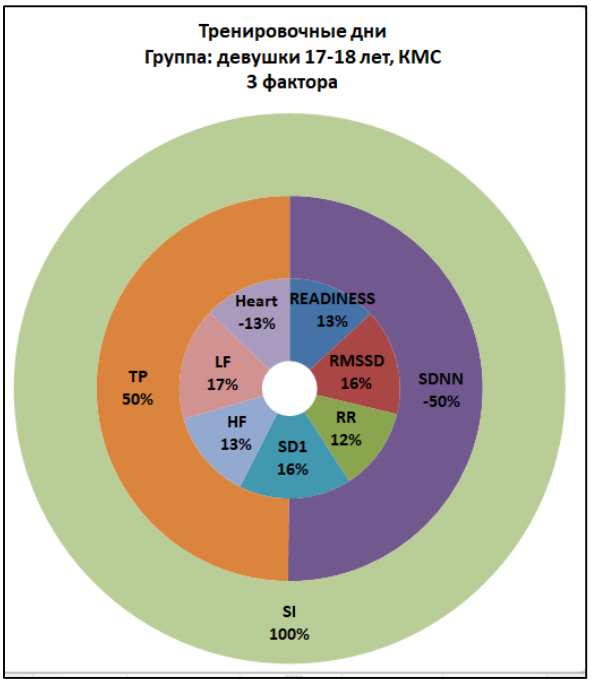


Рис. 11. Матрица факторных нагрузок спортсмена №2 в тренировочные дни.

Матрица факторных нагрузок (выходные дни)

Переменная	Фактор №1	Фактор №2
READINESS	0,834	-0,466
RMSSD	0,932	-0,094
RR	0,847	0,431
SDNN	0,077	0,922
SD1	0,986	-0,076
TP	-0,176	-0,906
HF	0,883	0,423
LF	0,985	-0,077
SI	-0,122	0,758
Heart	-0,826	-0,255
Доля общей дисперсии	**0,581**	**0,284**

Таким образом, 2 общих фактора объясняют нам 86,47% вариации результатов HRV-показателей спортсмена №2

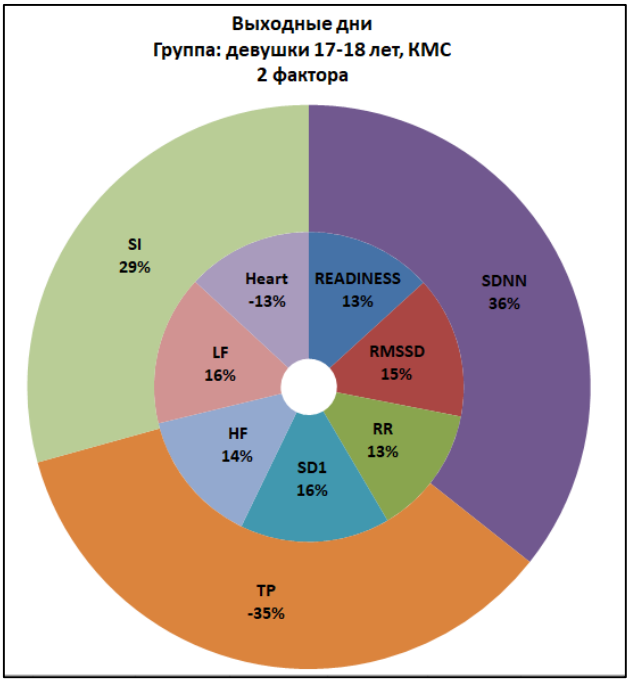


Рис. 12. Матрица факторных нагрузок спортсмена №2 в выходные дни.

Для сравнения можно провести факторный анализ для спортсмена №3 из группы «Девушки 17-18 лет, 1 разряд», результаты которого представлены на рисунках 13-14.

Матрица факторных нагрузок (тренировочные дни)

Переменная	Фактор №1	Фактор №2
READINESS	0,564	**0,740**
RMSSD	**0,975**	-0,010
RR	**0,977**	-0,124
SDNN	0,089	**-0,980**
SD1	**0,897**	0,409
TP	-0,029	**0,980**
HF	**0,954**	0,166
LF	**0,897**	0,410
SI	**0,918**	-0,264
Heart	-0,672	-0,502
Доля общей дисперсии	**0,646**	**0,276**

Таким образом, 2 общих фактора объясняют нам 92,15% вариации результатов HRV-показателей спортсмена №3

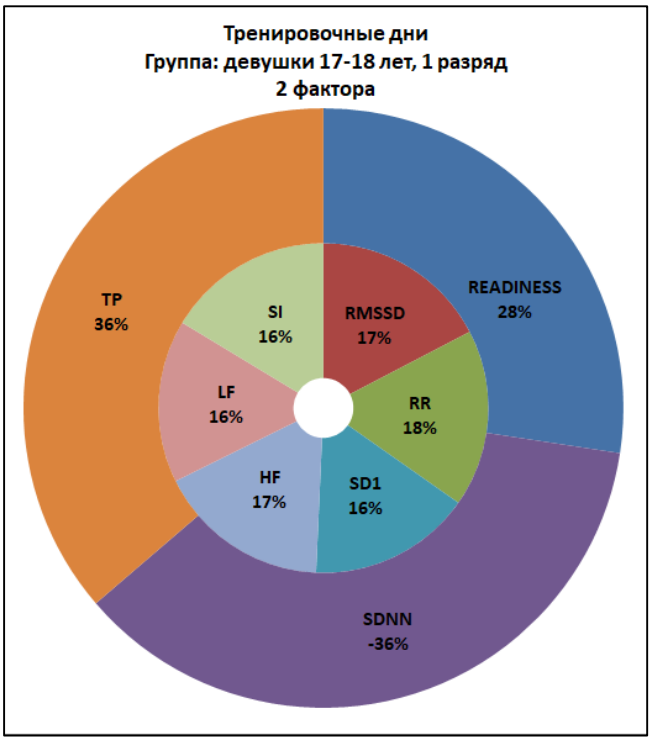


Рис. 13. Матрицы факторных нагрузок спортсмена №3 в тренировочные дни.

Матрица факторных нагрузок (выходные дни)

Переменная	Фактор №1	Фактор №2
READINESS	**0,837**	0,281
RMSSD	**0,928**	0,079
RR	**0,986**	0,102
SDNN	-0,075	**-0,993**
SD1	**0,991**	-0,006
TP	0,104	**0,989**
HF	**0,986**	0,032
LF	**0,991**	-0,007
SI	**0,932**	0,108
Heart	**-0,834**	-0,326
Доля общей дисперсии	**0,724**	**0,199**

Таким образом, 2 общих фактора объясняют нам 92,32% вариации результатов HRV-показателей спортсмена №3

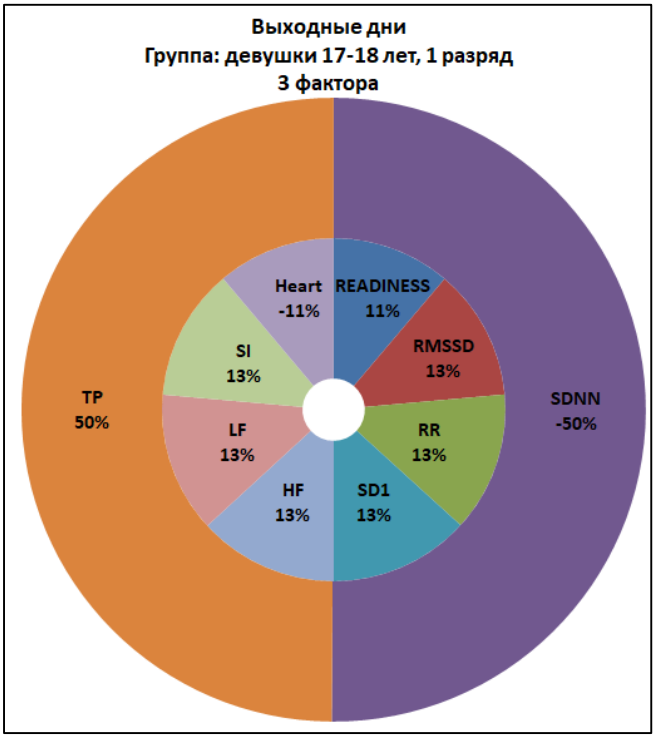


Рис. 14. Матрицы факторных нагрузок спортсмена №3 в выходные дни.

Таким образом, в статье приводится описание разработанного авторами модуля программно-информационной системы, предназначенного для исследования зависимостей между показателями вариабельности сердечного ритма спортсменов методом факторного анализа. Статистическая информация, полученная с помощью данного модуля, будет

способствовать более точной оценке физической подготовки спортсменов, что позволит тренерам и медицинскому персоналу выявить ключевые факторы, оказывающие влияние на здоровье, физическое состояние и функциональную подготовленность каждого спортсмена. Такой подход обеспечит возможность разрабатывать персонализированные тренировочные и реабилитационные программы, снижать риски заболеваний и травм, что, в конечном итоге, позволит повысить уровень спортивных достижений.

Результаты проведенных исследований способствуют развитию методов профилактики и лечения спортивных травм, оптимизации индивидуальных тренировочных программ для разных видов спорта. Указанные выше аспекты определяют актуальность и практическую значимость проделанной авторами работы.

СПИСОК ЛИТЕРАТУРЫ

1. Морозова Е. А. Факторный анализ составляющих спортивного результата сильнейших конькобежцев мира на дистанции 500 м // Педагогико-психологические и медико-биологические проблемы физической культуры и спорта. – 2017. – Т. 12. – № 1. – С. 22–28.
2. Корольков А. Н. Оценка общей физической подготовленности с помощью центроидного метода главных компонентов для многих переменных // Вестник спортивной науки. – 2013. – № 1. – С. 15–19.
3. Реуцкая Е. А., Павлова Н. В., Николаев Е. М. Критерии интегральной подготовленности высококвалифицированных биатлонистов к соревнованиям // Наука и спорт: современные тенденции. – 2017. – Т. 16. – № 3(16). – С. 67–72.
4. HEART RATE VARIABILITY (HRV) SOFTWARE – KUBIOS [Электронный ресурс]. – Режим доступа: https://www.kubios.com/ (дата обращения 29.08.2023).

ЕГОРОВА Д. К., ЗАВАРЮХИНА Ю. В.

ПРИМЕНЕНИЕ ИНСТРУМЕНТАРИЯ KNIME ANALYTICS PLATFORM ДЛЯ АНАЛИЗА СООТВЕТСТВИЯ РАБОЧИХ ПРОГРАММ УЧЕБНЫХ ДИСЦИПЛИН ТРЕБОВАНИЯМ РАБОТОДАТЕЛЕЙ

Аннотация. В статье рассматривается применение инструментария KNIME Analytics Platform для анализа текста в pdf-документах. Приведена реализация алгоритма анализа рабочих программ учебных дисциплин на соответствие требованиям работодателей.

Ключевые слова: KNIME Analytics Platform, алгоритм, workflow, text processing, узел, pdf-документ.

EGOROVA D. K., ZAVARYUKHINA YU. V.

APPLICATION OF KNIME ANALYTICS PLATFORM TOOLS TO ANALYZE THE COMPLIANCE OF SYLLABUSES WITH THE REQUIREMENTS OF EMPLOYERS

Abstract. The article considers the use of the KNIME Analytics Platform toolkit for text analysis in pdf documents. The implementation of the algorithm for analyzing the syllabuses of academic disciplines for compliance with the requirements of employers is described.

Keywords: KNIME Analytics Platform, algorithm, workflow, text processing, node, pdf-document.

Введение. Интеллектуальный анализ текста – это эффективный метод анализа данных, который можно использовать, для обработки текстовых документов, выявления ключевых терминов и тем, а также выявления скрытых отношений. Преимущества интеллектуального анализа текста очевидны, особенно в областях с большим количеством текстовых данных. Однако извлечение ценной информации, которая потенциально скрыта в текстовых данных, и явное отображение отношений между текстовыми данными, например, путем визуального представления ключевых атрибутов текста по отношению к определенным стандартам, может представлять собой довольно сложную задачу.

При решении подобного рода задач можно использовать low-code платформы визуальной разработки сценариев анализа данных. Согласно некоторым оценкам [3], к 2024 году порядка 70% приложений в мире будут разрабатываться на платформах low-code. Отличительной особенностью этих платформ, например, российских PolyAnalyst и Loginom, швейцарской KNIME, является то, что они не содержат встроенной системы анализа текстовой информации, процесс анализа следует «собирать» из своего рода узлов-nodes в которых используются библиотеки Python, R, JavaScript.

Далее будет приведено решение задачи анализа текстовой информации, извлеченной из интернет-источников и файлов pdf с использованием инструментария KNIME Analytics Platform.

Плагин Text Processing. Аналитическая платформа KNIME содержит плагин для обработки текста KNIME Text Processing [1], [2]. Плагин имеет открытый код и поддерживает многоуровневый процесс обработки текста – от чтения и синтаксического анализа через распознавание сущностей, фильтрации и манипуляции до подсчета количества слов, выделения ключевых понятий и, наконец, визуализации. Все это дает пользователю широкие возможности работы с текстом.

На рисунке 1 представлен репозиторий узлов обработки и анализа текстовой информации KNIME.

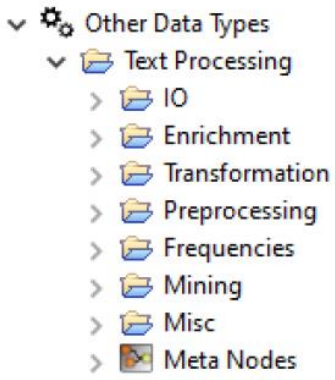


Рис. 1. Репозиторий узлов Text Processing.

Рассмотрим некоторые возможности указанного плагина для обработки текста.

Категория IO содержит узлы, позволяющие выполнить первоначальный анализ и чтение данных, а именно узлы синтаксического анализа, которые могут анализировать тексты из различных форматов, таких как DML, SDML, PubMed (формат XML), PDF, Word и др. Здесь текстовые документы представляются, в том числе, экземпляром класса Java Document, с использованием типов данных KNIME DocumentCell и DocumentValue.

Категория Enrichment содержит узлы, которые распознают стандартные именованные объекты (имена собственные, специфические обозначения, например, биомедицинские и др.), присваивая каждому распознанному именованному объекту значение тега. Тип тега представляет домен или тип маркера, а значение представляет конкретную характеристику в этом домене. Документ после процесса пометки называют «обогащенным», так как он содержит больше информации, чем первоначально.

Категория Transformation содержит узлы (вектор документа, вектор термина), позволяющие преобразовать текстовые данные в числовые. Эти узлы создают двоичное или числовое векторное представление для каждого термина.

Узлы категории Preprocessing используются на этапе глубокой предварительной обработки, заключающейся в очистке данных от стоп-слов, слов не имеющих содержания, знаков препинания и т.д. В итоге остаются только те термины, которые впоследствии используются для создания числового вектора или визуализируются в облаке тегов.

Узлы, осуществляющие вычисление терминальной частоты tf (относительной или абсолютной), обратной частоты документа idf, фильтрацию терминов с использованием показателя оценки релевантности, графическое представление документа для поиска ключевых слов, находятся в категории Frequency.

Еще более тонкая работа с текстом осуществляется с помощью узлов категорий Mining (Извлечение), Misc (Разное) и Meta Nodes (Мета узлы).

Реализация алгоритма. Использование узлов плагина для обработки текста KNIME Text Processing позволяет решить актуальную задачу анализа рабочих программ учебных дисциплин на соответствие требованиям работодателей. Требования работодателей можно получить, например, из описания вакансий, размещенных на соответствующих интернет площадках. Рабочие программы учебных дисциплин, в свою очередь, содержат перечень компетенций, знаний и умений, которыми должен обладать студент, в результате освоения той или иной дисциплины. Эти данные являются текстовыми. Применяя алгоритмы кластеризации и анализа текстовых данных определим, к какому запросу вакансий из области ИТ относится выбранная рабочая программа.

Для реализации задачи необходимо выполнить следующие шаги:

1. создать запросы для поиска данных о вакансиях;
2. присвоить каждой вакансии свой класс;
3. сформировать пакет слов для извлеченных вакансий и pdf-документа рабочей программы;
4. из пакетов слов построить векторы документов;
5. применить наивный байесовский классификатор для сопоставления рабочей программы и запроса.

Поиск данных о вакансиях осуществлялся на сайте https://hh.ru. Для реализации алгоритма были выбраны следующие вакансии в г.о. Саранск: мобильный разработчик, разработчик игр, веб-разработчик.

Шаг 1. Создаем в KNIME Analytics Platform рабочий процесс (Workflow). Считываем данные при помощи узла GET Request (рис. 2). Первоначально работаем с вакансиями. Для

этого создаем три одинаковые ветки, так как планируется извлечение трех разных типов вакансий.

Рис. 2. Узел GET Request.

В настройках конфигурации узла добавляем запрос о вакансиях написанный на Python, имеющий, для каждой ветки следующий вид:

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=вебANDразработчик&search_field=name

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=разработчикANDигр&search_field=name

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=мобильныйANDразработчик&search_field=name

Данные запросы реализуют извлечение по сто вакансий по г.о. Саранск, названия которых содержат слова «веб-разработчик» (первый запрос), «разработчик игр» (второй запрос), «мрбильный разработчик» (третий запрос) с указанными размерами заработной платы. При необходимости данные параметров извлечения информации можно менять, руководствуясь документацией HeadHunter API.

Далее выполняем узел и непосредственно извлекаем в каждой ветке по 100 вакансий, воспользлвавшись узлом JSON Path. Соединяем его с узлом GET Request (Рис. 3).

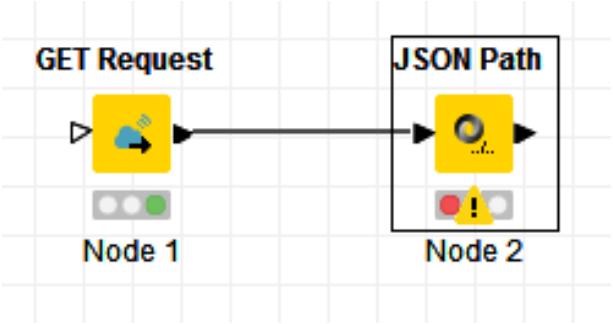


Рис. 3. Соединение узла JSON Path.

Переходим в конфигурацию узла JSON Path. Во вкладке Setting выбираем параметр идентификатора записи (id) и с помощью правой кнопки мыши добавляем путь (Add JSONPath). В строках содержащих название вакансии (name) и параметры заработной платы (currency – валюта, gross – сумма до уплаты подоходного налога, to и from в salary – размер заработной платы от/до, responsibility в snippet – требования к кандидату, name в schedule – рабочий режим) меняем значение «0» после ['items'] на «*» (Рис. 4), для того, чтобы иметь возможность работать со всеми вакансиями. В противном случае из всей базы будет извлечена только одна вакансия.

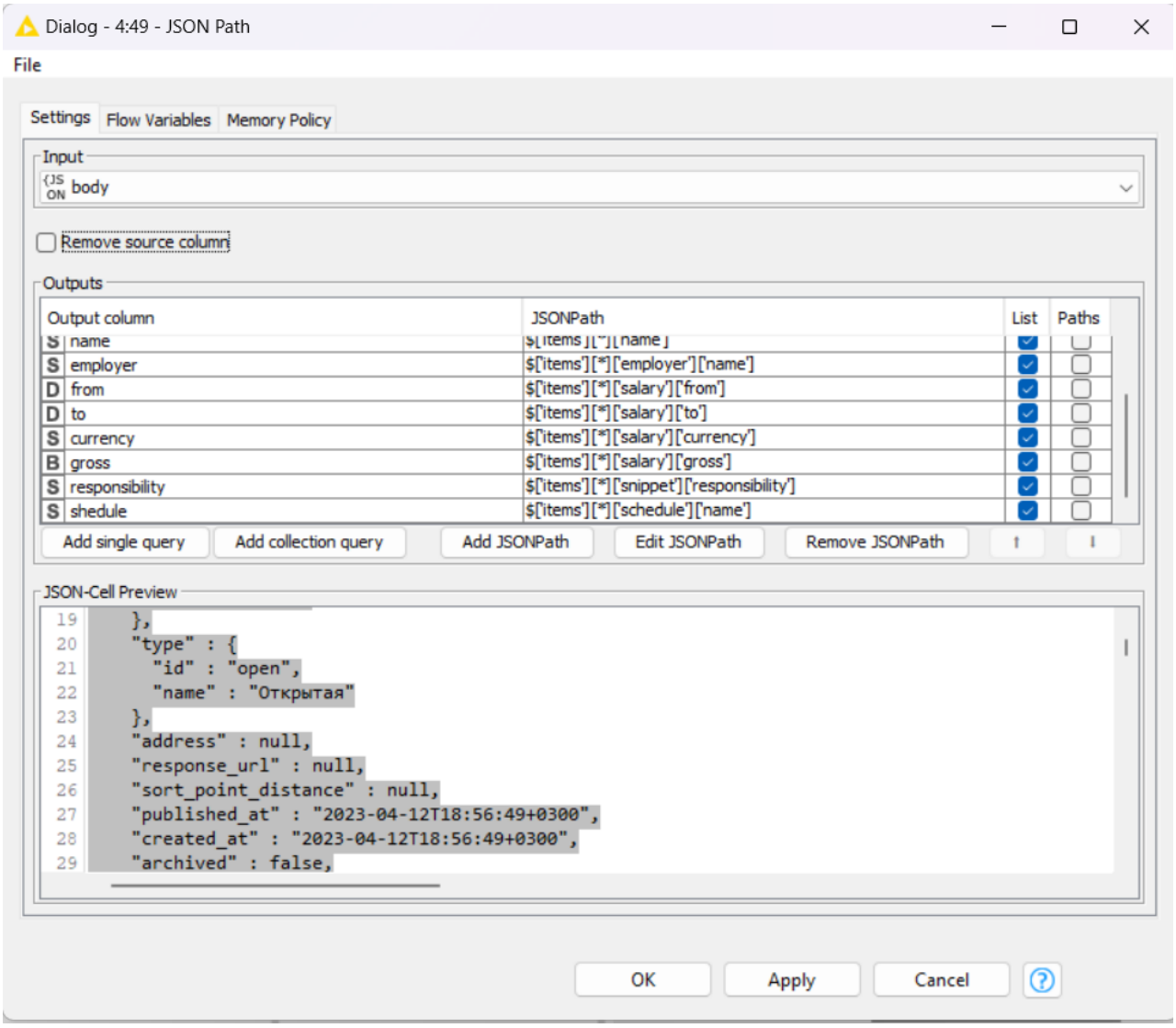


Рис. 4. Настройка конфигурации узла JSON Path.

Приводим узел в действие. Получаем список запрашиваемых данных, который необходимо разделить на отдельные документы (вакансии). Для этого воспользуемся узлом

Ungroup. Данный узел создает список строк с извлеченными вакансиями и их параметрами. Соединяем узлы JSON Path и Ungroup, затем запускаем их выполнение.

Шаг 2. Далее используем узел Math Formula для присвоения номера класса ветки. После запуска процесса получаем список с добавленым столбцом номера класса в таблице данных каждой ветки. Затем дважды используем узел Concatenate для объединения таблиц полученных на разных ветках.Таблица на входе 0 задается как первая входная таблица (верхний входной порт), таблица на входе 1 является второй таблицей, соответственно. Получаем одну общую таблицу, содержащую весь перечень извлеченных вакансий с присвоенными им классами.

Далее используем узел Column Filter, он необходим для фильтрации данных, так как для решения задачи необходимо оставить класс ваканссии (class) и перечень требований к кандидату (responsibility) (рис. 5).

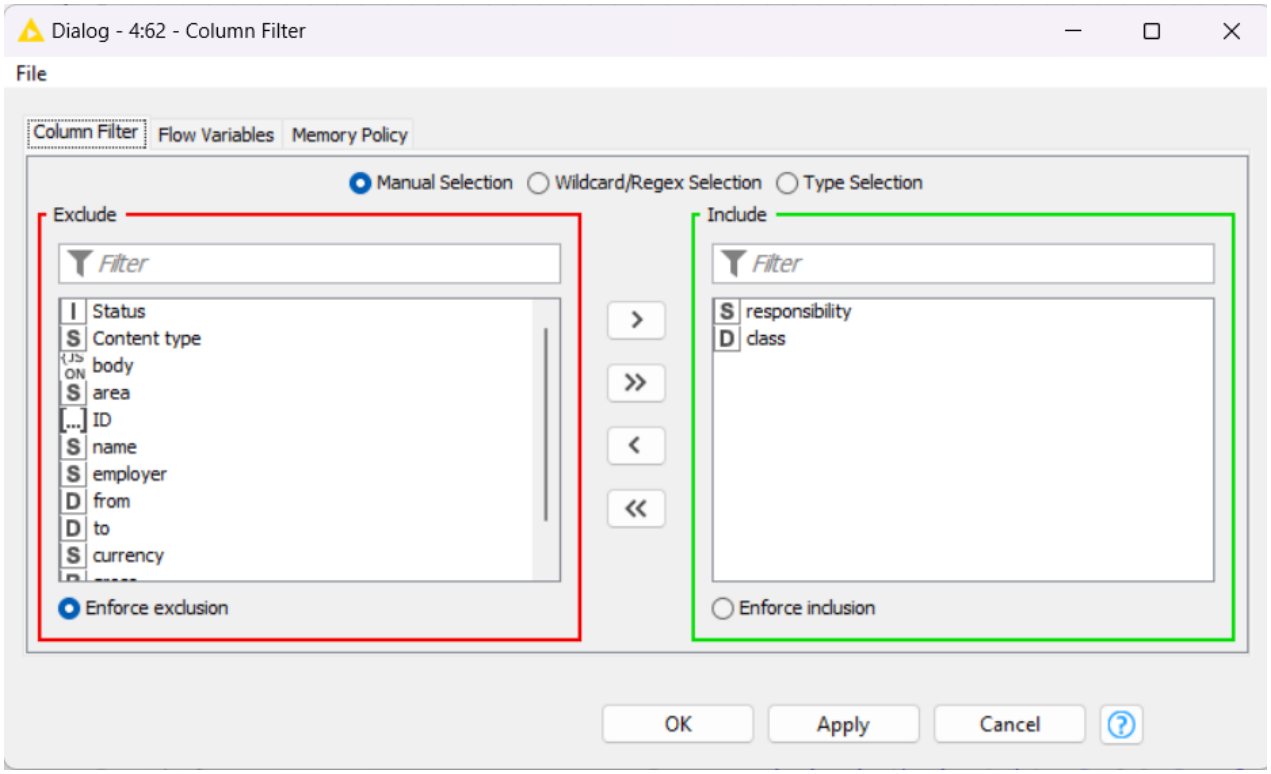


Рис. 5. Настройки узла Column Filter.

Используем узел Missing Value для заполнения недостающих значений, если таковые имеют место быть. Затем необходимо преобразовать указанные строки в документы. Используем узел Strings To Document (рис. 6).

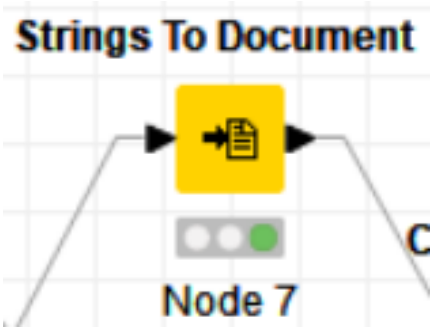


Рис. 6. Узел Strings To Document.

Шаг 3. Переходим к формированию пакета слов для вакансий. Используем узел Bag Of Words Creator. Пакет слов состоит из столбцов, содержащих термины, встречающиеся в соответствующем документе (вакансии). Все столбцы, связанные с терминами, такие как столбец документа, можно выбрать в диалоговом окне конфигурации узла. В нашем случае в конфигурации выбираем параметры «Document», «class» и «responsobility» (рис. 7).

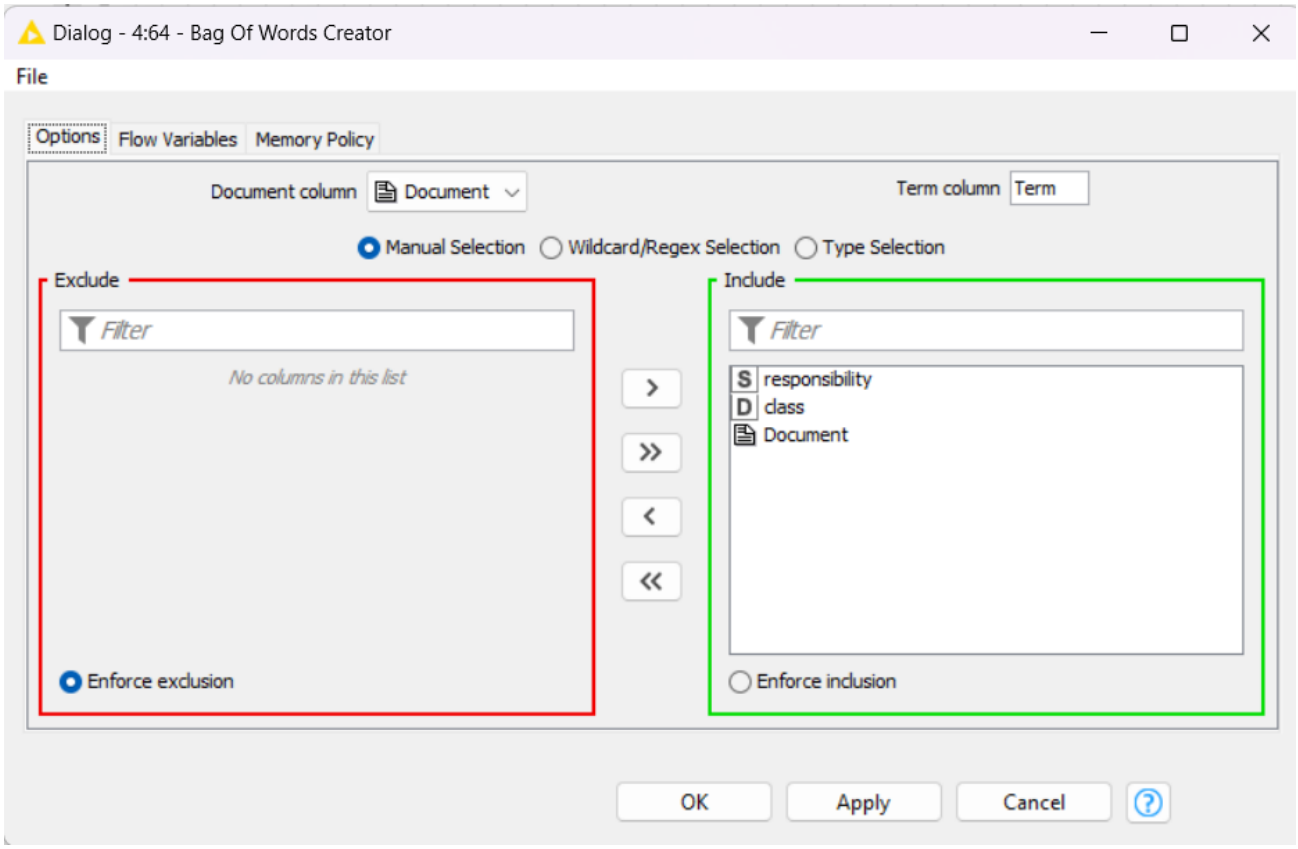


Рис. 7. Настройки узла Bag Of Words Creator.

Шаг 4. Создаем вектор документа для каждой вакансии с помощью узла Document Vector. Этот узел создает вектор документа, представляющего его в пространстве терминов.

Размерность векторов будет равна количеству различных термов полученных в пакете слов. В конфигурации данного узла выбираем параметр «Document» (рис. 8).

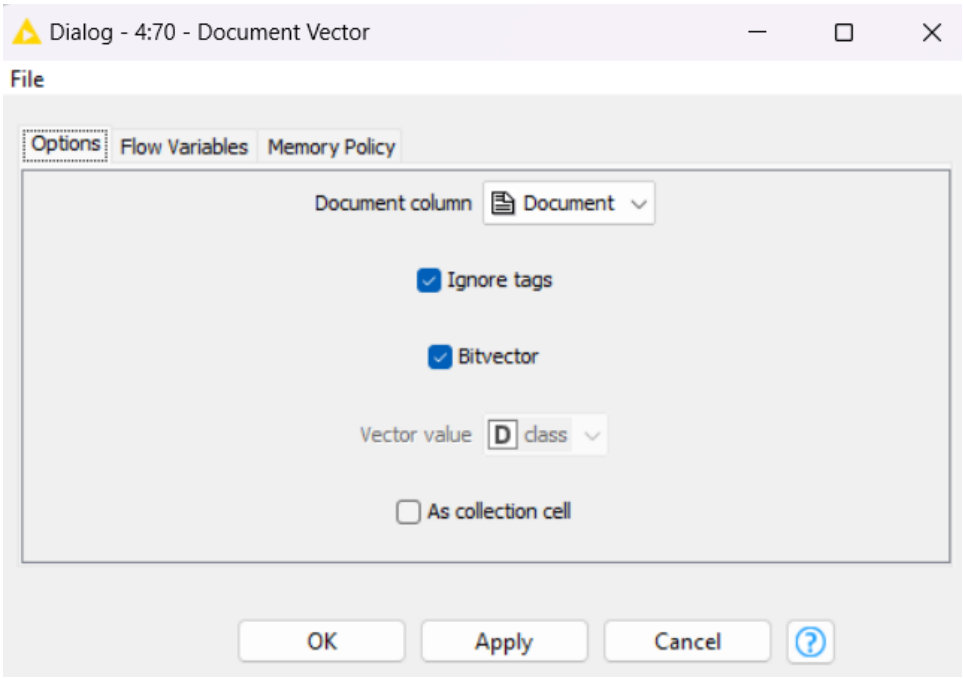


Рис. 8. Настройки узла Document Vector.

Также необходимо выполнить фильтрацию столбцов только по классу (узл Column Filter) и преобразование чисел в строки (узел Number to string) для формирования таблицы и их объединение (узел Column Appender) .

Параллельно создаем ветку для загрузки рабочей программы учебной дисциплины, например, «Работа с удаленными базами данных» и реализации описанных выше шагов 3 и 4. Здесь используем узел PDF Parser, в настройках конфигурации которого выбираем путь к файлу. Затем используем узел RegEx Filter для удаления ненужных символов, прописывая их в настройках конфигурации узла (рис. 9).

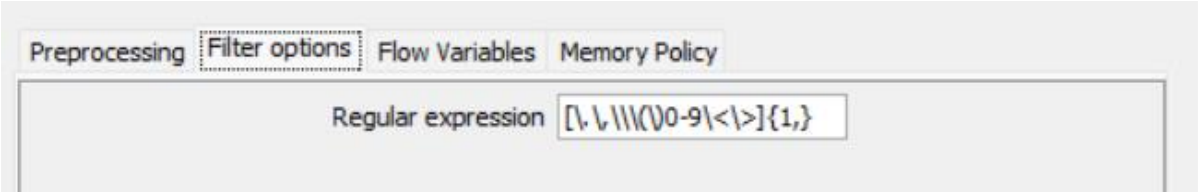


Рис. 9. Настройки узла RegEx Filter.

Шаг 5. Далее используем узел Naive Bayes Learner для «обучения» байесовской модели предсказывающей класс для каждой вакансии на основе соответствующего вектора

документа (вакансии). В результате получили 3 класса соответствующие трем выбранным типам вакансий (таб. 1).

Таблица 1

Соответствие класса выбранному типу вакансии

Класс	Соответсвующий тип ваканси на hh.ru
1	Мобильный разработчик
2	Веб-разработчик
3	Разработчик игр

Узел Naive Bayes Learner является входным для узла Naive Bayes Predictor, который прогнозирует класс документа содержащего рабочую программу на основе обученной байесовской модели.

В завершении соединяем две ветки реализующие обработку вакансий и текста рабочей программы в узле Naive Bayes Predictor.

Результатом выполнения узла Naive Bayes Predictor является тип вакансий с сайта hh.ru, которому соответствует загруженная программа учебной дисциплины. Открывая вкладку The classified data (с анг. – секретные данные) конфигурации узла Naive Bayes Predictor в последнем столбце Prediction (с анг. – прогноз) получили класс 3 (рис. 10), что соответствует запросу работодателей по типу вакансий «Разработчик игр» (табл. 1).

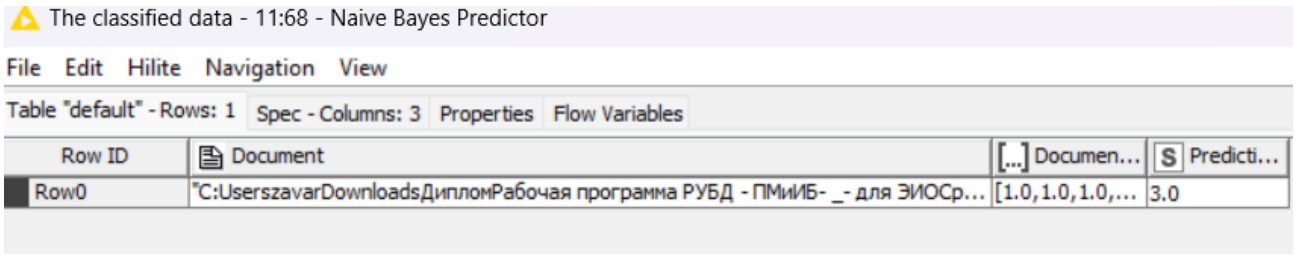


Рис. 10. Результат работы алгоритма.

Предлагаемый алгоритм, состоящий из 28 узлов (рис. 11), разработан согласно модели low-code и не является desktop-приложением, вследствие чего может быть полезен узкому кругу специалистов, занимающихся анализом текстовой информации, получаемой из различных источников.

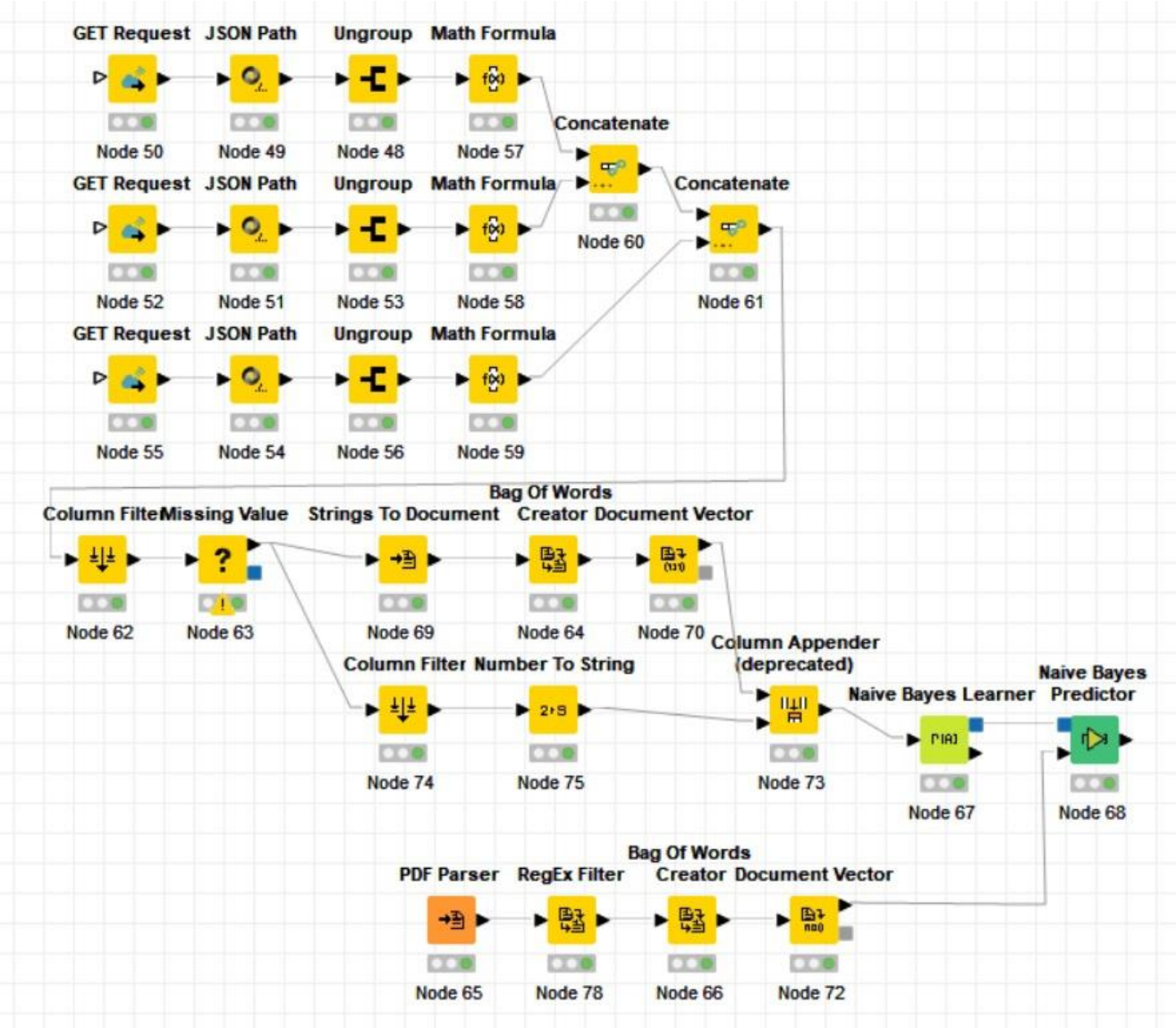


Рис. 11. Итоговый рабочий процесс.

СПИСОК ЛИТЕРАТУРЫ

1. From data collection to text mining [Электронный ресурс]. – Режим доступа: https://www.knime.com/blog/data-collection-to-text-mining (дата обращения 01.09.2023).
2. Knime Analytics Platform [Электронный ресурс]. – Режим доступа: https://www.knime.com/ (дата обращения 01.09.2023).
3. Low-code в России: Быстрый рост без иностранных конкурентов [Электронный ресурс]. – Режим доступа: https://startpack.ru/articles/low-code-v-rossii (дата обращения 30.10.2023).

БУТКИНА А. А., ТРЕМАСКИН К. Д., ШАМАЕВ А. В.

РАЗРАБОТКА ПРОГРАММНО-ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ ВЕДЕНИЯ АРХИВА ЛИЧНЫХ ДОКУМЕНТОВ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ ТЕГОВ

Аннотация. В статье описана разработанная авторами программно-информационная система, предназначенная для автоматизации процесса ведения архива личных документов с использованием системы тегов, повышающих эффективность фильтрации данных. Описана реализация интеграции разработанного веб-приложения с Яндекс Диском.

Ключевые слова: веб-приложение, архив, документы, автоматизация, система тегов, PHP, Laravel, базы данных, СУБД MySQL, облачные сервисы, Яндекс Диск.

BUTKINA A. A., TREMASKIN K. D., SHAMAEV A. V.

DEVELOPMENT OF SOFTWARE INFORMATION SYSTEM TO AUTOMATE MAINTENANCE OF PERSONAL DOCUMENTS ARCHIVE USING TAG SYSTEM

Abstract. The article presents the software information system developed by the authors and designed to automate the process of maintaining an archive of personal documents using a tag system to improve the efficiency of data filtering. The implementation of the integration of the developed web application with Yandex Disk is described.

Keywords: web application, archive, documents, automation, tag system, PHP, Laravel, database, MySQL DBMS, cloud services, Yandex Disk.

Введение. Каждый человек в повседневной жизни человек рано или поздно сталкивается с проблемой удобного ведения архива личных документов. К документам любого личного архива относятся: биографические данные (свидетельства, удостоверения, справки, результаты исследований и т.п.), документы служебной и общественной деятельности, переписка, фотографии, изобразительные материалы. При этом зачастую происходят события (например, оформление наследства), связанные с оформлением, хранением и обработкой большого количества документов. Не запутаться в таком множестве разнообразных документов без использования специального инструментария для их упорядоченного хранения крайне тяжело.

Ведение архива личных документов только в физическом виде является крайне нерациональным, так как в любой момент может возникнуть необходимость в получении копии определённого документа или проверке наличия искомого документа в архиве. Использование в качестве такого архива существующих облачных хранилищ в том виде, в каком они представлены сейчас, так же представляет собой неидеальный вариант, так как с

ростом количества документов, будет все сложнее выполнять добавление новых файлов и поиск существующих. Об эффективном хранении локальных электронных копий документов на домашнем компьютере или смартфоне тем более говорить не приходится.

Данное исследование посвящено созданию программно-информационной системы в виде веб-приложения, предназначенного для автоматизации процесса ведения архива личных документов пользователя с использованием системы тегов, обеспечивающих удобную и эффективную фильтрацию данных. Данное решение позволит пользователям упростить ориентирование в архиве своих документов благодаря наличию системы тегов и грамотно организованному пользовательскому интерфейсу. При этом пользователь сможет буквально в несколько кликов находить именно то, что ему необходимо, что позволит сэкономить время и усилия, затраченные на поиск, по сравнению, например, с использованием того же физического архива.

Актуальность работы состоит в том, что существующие сервисы, предназначенные для хранения данных, неудобны при работе с большим количеством файлов, а их системы фильтрации неэффективны, что доставляет массу неудобств пользователям. Также следует отметить, что сильная вовлеченность людей в использование интернет-технологий позволит пользователям иметь оперативный доступ к своим документам в любой момент времени.

Для достижения указанной цели были поставлены и решены следующие ***задачи***:

- выполнить анализ предметной области;
- выполнить анализ и выбор инструментов разработки;
- описать входные данные и построить базу данных системы;
- разработать архитектуру системы с помощью UML-диаграмм;
- реализовать автоматизированное заполнение базы данных (БД) файлами, размещенными в облачном хранилище пользователя;
- реализовать систему на выбранных языках программирования;
- выполнить проверку работоспособности разработанной системы.

Анализ предметной области. Первоочередным вопросом, вставшим перед авторами, являлся выбор облачного хранилища, предназначенного для интеграции с разрабатываемой программно-информационной системой. Были изучены научные статьи, посвященные указанной тематике, в частности, в статье [1] рассмотрены облачные хранилища, которые являются наиболее популярными на сегодняшний день – Dropbox, Яндекс Диск, GoogleDrive, MicrosoftSkyDrive и SugarSync, выполнено сравнение их характеристик и предлагаемого ими функционала: тарифные планы, предоставляемые объемы памяти, скорость соединения и стабильность работы серверов. Авторами также было выполнено собственное исследование описанных сервисов, по результатам которого можно сформулировать следующие выводы.

На сегодняшний день отсутствуют по-настоящему удобные и эффективные сервисы для ведения личного архива документов. Рассмотренные облачные хранилища либо не имеют систем фильтрации, либо имеют их весьма неудачные реализации. Отметим, что на данный момент не существует сервисов, специализирующихся на хранении документов. В качестве оптимального решения для интеграции разрабатываемой системы с облачным хранилищем был выбран отечественный сервис Яндекс Диск.

Выбор технологии реализации. Так как разрабатываемое приложение включает клиентскую и серверные части, далее опишем инструменты, выбранные для их разработки.

Для реализации серверной части могут быть использованы различные инструменты и языки программирования. Наиболее популярными языками программирования для backend разработки в настоящее время являются Java, Ruby, Golang, PHP, Python. У каждого из них есть свои преимущества и недостатки, начиная с распространенности и заканчивая специфичностью работы. В техническом плане для большинства проектов отсутствуют какие-либо ограничения на выбор языка – практически любой функционал может быть успешно реализован на любом серверном языке, поэтому выбор языка не накладывает никаких ограничений на проект. Однако разница существует с экономической точки зрения. Одним из наиболее распространенных языков программирования в России на сегодня является PHP, поэтому при его использовании достаточно просто найти как отдельных программистов, так и аутсорсинговые компании, которые смогут работать над проектом. Из минусов — порог вхождения в сферу PHP-программирования невысок, поэтому на рынке присутствует немало «непрофессионалов». Если рассматривать в качестве альтернативы другие языки из представленного выше списка, то средний уровень специалистов на кадровом рынке, как правило, выше, но найти их уже не так просто и их услуги стоят несколько дороже. Наш проект, как и любой другой, необходимо сопровождать в ходе эксплуатации, поэтому экономическая составляющая вопроса занимает не последнее место.

С точки зрения разработчиков наилучшим вариантом также будет PHP. Для него существует хорошо описанная документация, существует большое сообщество программистов. Кроме того, PHP-приложение будет относительно просто адаптировать при реализации сервера для мобильного приложения, что несомненно является большим преимуществом. Резюмируя выбор языка программирования, можно сказать, что ***PHP*** является наиболее оптимальным вариантом для реализации нашего проекта.

В качестве фреймворка для разработки был выбран ***Laravel***, так как в настоящее время ему не существует значимой альтернативы, поскольку остальные фреймворки существенно уступают ему по предоставляемому функционалу. В качестве отличительных особенностей Laravel можно выделить его простоту и наличие большого количества модулей.

Далее был выполнен выбор системы управления базами данных (СУБД) с учетом того ограничения, что в качестве основного языка программирования будет использоваться PHP. Были проанализированы такие СУБД как MySQL, MariaDB, PostgreSQL, SQLite, MongoDB, Redis, CouchDB. В результате проведенного анализа была выбрана СУБД ***MySQL***.

В статье [2] проводится анализ двух популярных механизмов хранения данных (движков) в MySQL – InnoDB и MyISAM. Первый является механизмом по умолчанию в MySQL 5.1 и не поддерживает механизмы транзакций и внешних ключей. Его преимущество заключается в том, что он имеет высокую скорость доступа. Второй, InnoDB, более распространен на практике – он поддерживает транзакции, внешние ключи, блокировку на уровне строк, однако является более медленным в работе, по сравнению с MyISAM. Поэтому, было решено использовать оба движка: InnoDB – для всей информации проекта, а MyISAM – для хранения информации, к которой необходим постоянный доступ на чтение.

При разработке frontend части веб-приложения использовался фреймворк ***Bootstrap.***

В качестве дополнительных инструментов разработки использовались интегрированная среда разработки PhpStorm, редакторы Visual Studio Code и Notepad++ (для работы с конфигурационными файлами PHP), система контроля версий Git, инструмент для тестирования API Postman, препроцессор SASS**.**

Построение базы данных. Эффективность любого приложения, использующего в своей работе базу данных, зависит от того, насколько оптимально спроектирована ее схема.

Поскольку разрабатываемая система должна хранить информацию о документах, пользователях, тегах, а также связи между всеми сущностями системы, то для этого необходимо создать базу данных с подходящей структурой, которая будет готова к масштабированию в ходе развития проекта. Опишем подробнее ключевые понятия:

– Тег – метка (дескриптор, описание), которая создается администраторами (подготавливаются базовые заготовки) или самими пользователями для её последующего присваивания документу или группе документов для облегчения поиска документов;

– Документ – сущность, которая содержит в себе данные о загружаемом файле (название, описание, тип и т.д.) и его расположении в хранилище на сервере или в облаке;

– Группа документов – документы, которые пользователь объединил в единое целое, например, через присваивание им тегов;

– Пользователь – объект, который представляет действующее лицо системы, это может быть как рядовой пользователь системы, так и её администратор.

В процессе разработки основного модуля веб-приложения, реализующего базовый функционал (авторизация, работа с документами, присваивание им тегов, фильтрация данных) была спроектирована схема базы данных, состоящая из семи таблиц (рис. 1).

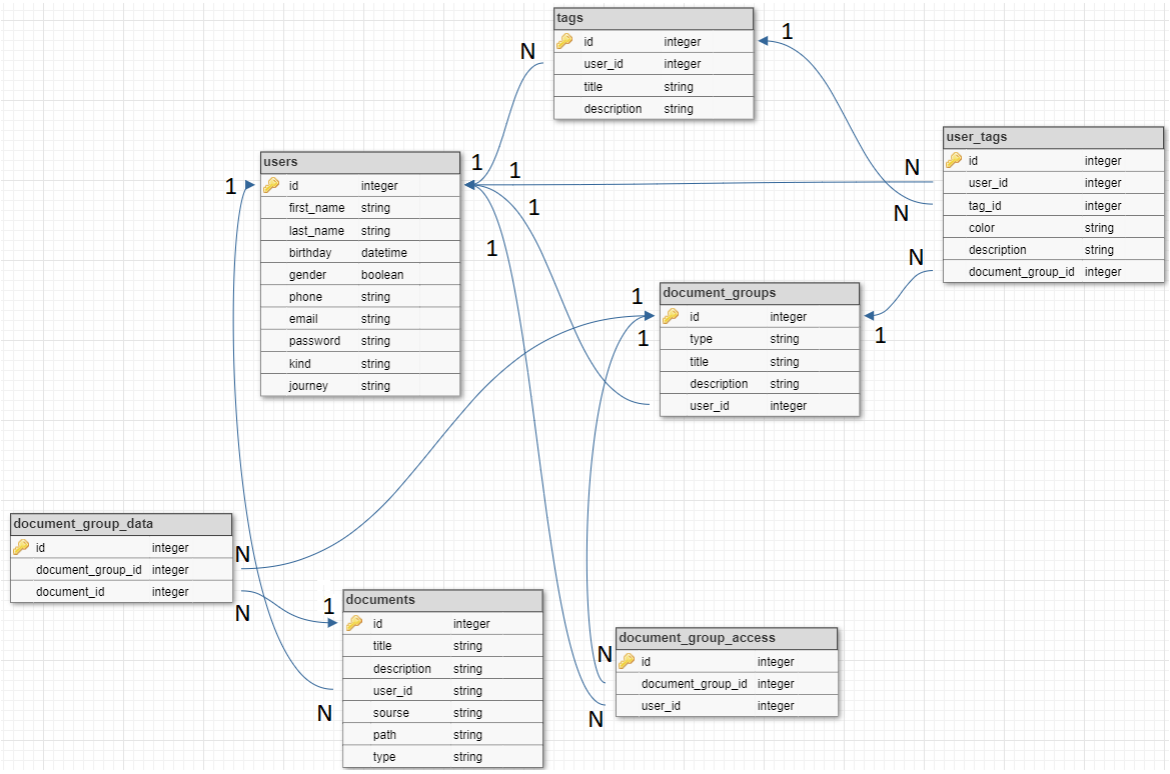


Рис. 1. Схема основного модуля базы данных.

Опишем назначение каждой из таблиц, приведенной на рисунке 1:

– Таблица «users» хранит данные обо всех пользователя и содержит поля с личной информацией, поля для аутентификации и служебные поля (тип пользователя, особые отметки). Эта таблица имеет следующие связи с другими таблицами: documents (один ко многим) – один пользователь может загрузить много документов, tags (один ко многим) – один пользователь может создать множество тегов, user_tags (один ко многим) – один пользователь может присвоить много тегов разным группам, document_group (один ко многим) – один пользователь может создать много групп документов, document_group_access (один ко многим) – один пользователь может иметь доступ ко многим группам документов;

– Таблица «documents» содержит информацию о документах, загруженных пользователями: название и описание документа, его тип, принадлежность конкретному пользователю, способ загрузки документа и полный путь к его расположению. Эта таблица имеет связь с таблицей document_group_data (один ко многим) – один документ может принадлежать ко многих группам документов одновременно;

– Таблица «documents_groups» хранит данные о группе документов, созданной пользователем, в качестве основных характеристик которой задаются её тип, название и описание. Эта таблица имеет такие связи с другими таблицами: document_group_data (один ко многим) – одна группа документов может содержать много документов, user_tags (один ко многим) – одной группе может быть присвоено несколько тегов, document_group_access (один ко многим) – доступ к одной группе могут иметь много пользователей;

– Таблица «documents_group_data» содержит информацию о группах – это связи группы и конкретных документов, где каждая запись содержит id документа и группы, к которой он отнесен. Документ может быть отнесен сразу к нескольким группам;

– Таблица «documents_group_access» хранит данные о пользователях, которые имеют доступ к этой группе документов. Каждая запись в данной таблице содержит id группы документов и id пользователя, к которому она относится;

– Таблица «tags» хранит данные о базовых и публичных тегах, которые может использовать каждый пользователь, а также основную модель тега, которая включает название, описание и автора тега. Эта таблица имеет связь с таблицей user_tags (один ко многим) – один базовый тег может быть использован многими пользователями;

– Таблица «user_tags» содержит теги, которые использует только определенный пользователь или группа документов. Данная модель может существовать как независимо, так и базироваться на модели из таблицы «tags», «наследуя» от нее различные свойства. В таблице также задаются дополнительные параметры тега (например, цвет отображения тега).

Создание таблиц осуществлялось с помощью ***механизма миграций***. Миграции позволяют определять схемы базы данных приложения и совместно использовать их. Чтобы сгенерировать новую миграцию базы данных, использовалась команда `php artisan make:migration`. Эта команда помещает новый класс миграции в каталог приложения `database/migrations`. Каждое имя файла миграции содержит временную метку, которая позволит Laravel определять порядок применения миграций. Дополнительно в каждую таблицу были добавлены timestamps-поля `created_at`, `updated_at` и `deleted_at`, которые необходимы для корректной работы Eloquent ORM Laravel.

После создания классов миграций для каждой из таблиц, приведенных на рисунке 1, был запущен механизм миграции консольной командой `php artisan migrate`. В результате в базе данных были созданы описанные таблицы (рисунок 2).

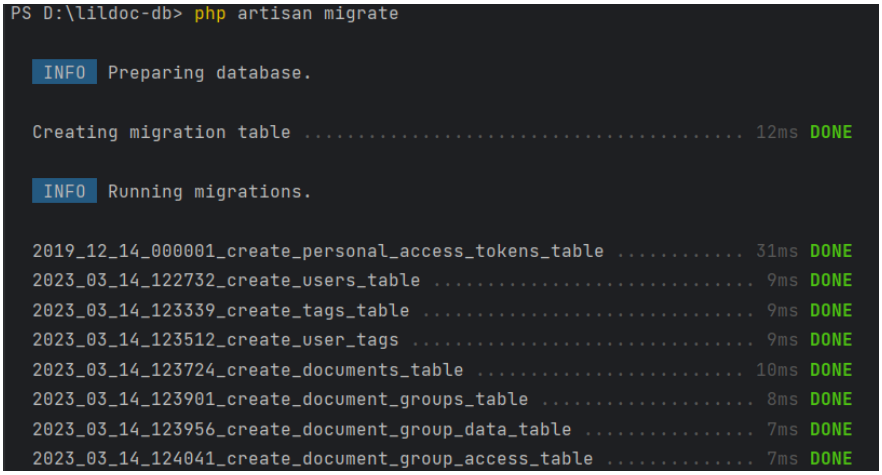


Рис. 2. Вызов консольной команды для запуска механизма миграций.

После формирования таблиц для каждой из них были созданы Eloquent модели, используемые для взаимодействия с родительскими таблицами. Помимо получения записей из таблицы БД модели Eloquent позволяют вставлять, обновлять и удалять их из таблицы. Модели расширяют класс Illuminate\Database\Eloquent\Model. Для генерации новой модели Eloquent использовалась команда `php artisan make:model`. Она помещает новый класс модели в каталог приложения app/Models. Также внутри класса модели были установлены зависимости между таблицами для организации удобного взаимодействия с данными из таблиц. Пример вызова консольной команды для создания модели представлен на рисунке 3.

```
PS D:\lildoc-db> php artisan make:model User

  INFO  Model [D:\lildoc-db\app/Models/User.php] created successfully.
```

Рис. 3. Вызов консольной команды для создания модели для таблицы «users».

После создания Eloquent моделей можно обращаться к данным таблицы, создавая экземпляр класса, которой относится к требуемой таблице.

Проектирование. Далее авторами была разработана архитектура системы с помощью следующих UML-диаграмм: диаграммы вариантов использования (рис. 4), диаграммы размещения и диаграммы компонентов. Рассмотрим их более подробно.

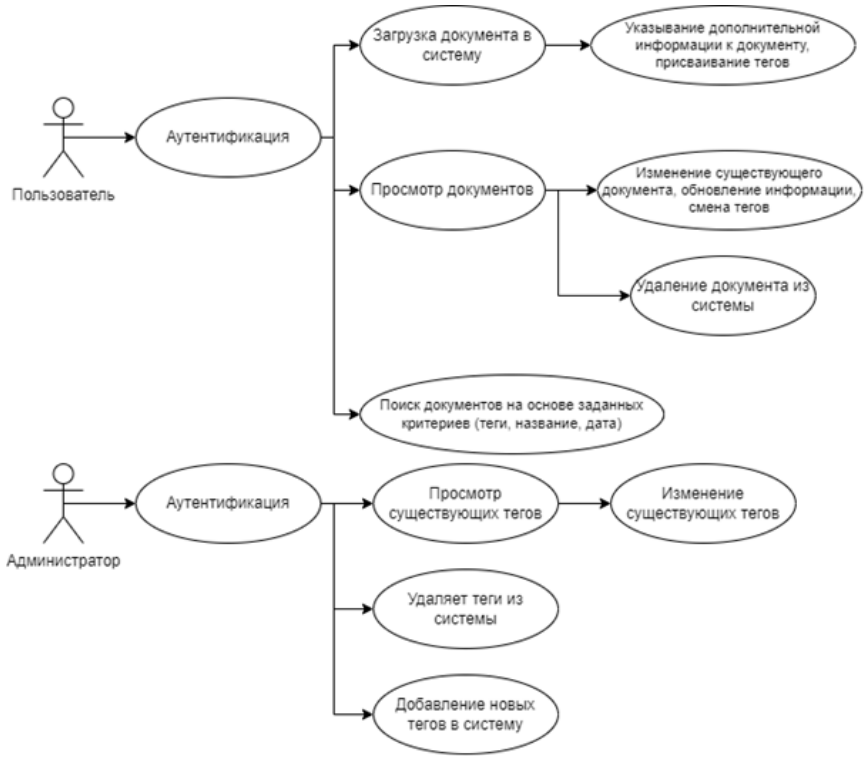


Рис. 4. Диаграмма вариантов использования.

При построении диаграммы вариантов использования были выделены два актера: администратор – актер, который имеет возможность управления базовыми тегами и фильтрами, а также пользователь – актер, использующий функционал приложения по его

назначению. Следует отметить, что доступ ко всему функционалу пользователь получает только после успешного прохождения процедуры регистрации и аутентификации. Основные сценарии для обоих актеров показаны на рисунке 4.

Диаграмма размещения является важным инструментом при проектировании и анализе системы, помогая понять её структуру и взаимодействие компонентов (рис. 5).

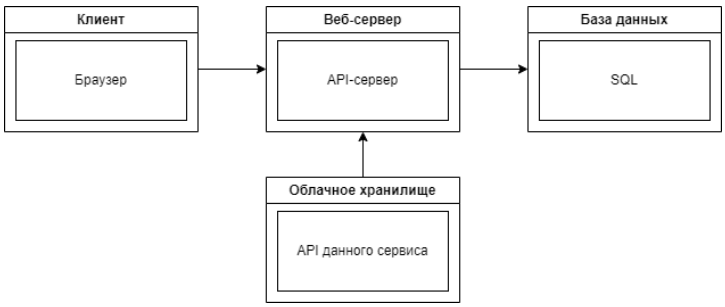


Рис. 5. Диаграмма размещения.

Связь между компонентами осуществляется путем взаимодействия через сеть интернет. В данном случае видно, что на сервере разворачивается веб-приложение, которое имеет связь с сервером базы данных. При этом пользователи обращаются к веб-серверу с различных устройств через браузеры. Опишем схему работы системы подробнее:

1. ***Клиентское приложение***. Это программное обеспечение, запущенное на устройстве пользователя и обеспечивающее интерфейс для взаимодействия с системой. С его помощью пользователь отправляет запросы на веб-сервер для обработки и получения данных. Оно также отображает результаты выполненных запросов;

2. ***Веб-сервер***. Этот компонент принимает и обрабатывает запросы от клиентского приложения, служит промежуточным звеном между клиентским приложением и БД;

3. ***База данных***. База данных является центральным хранилищем информации о личных документах. В ней хранятся данные, такие как документы, метаданные и система тегов. База данных позволяет сохранять, извлекать, обновлять и удалять данные в архиве;

4. ***Облачное хранилище***. Данный компонент позволяет получить доступ к файлам пользователя, хранящимся на стороннем сервисе (в нашем случае Яндекс Диск). Подключение к нему осуществляется через соответствующий API и используется с целью переноса файлов пользователя в разрабатываемую систему.

Опишем диаграмму компонентов для разрабатываемого веб-приложения (рис. 6):

1. ***Компоненты управления состоянием*** **(*StateManagement*)**. Используются для централизованного хранения и управления состоянием приложения.

2. ***Сетевой слой*** (***API***). Взаимодействует с серверным API для обмена данными.

3. ***Управление маршрутизацией*** **(*Router*)**. Отвечает за навигацию веб-приложения на основе текущего URL или действий пользователя.

4. ***Клиентский интерфейс*** (***UI***). Отвечает за отображение пользовательского интерфейса веб-приложения. Включает страницы, компоненты и элементы интерфейса.

5. ***Бизнес-логика и обработка данных*** (***business logic***). Выполняет бизнес-логику и обрабатывает данные приложения, например, поиск документов по тегам.

6. ***Серверная часть*** (***Backend***). Обеспечивает функциональность, требующую обработки на сервере, такую как аутентификация, хранение и обработка данных.

7. ***База данных*** (***SQL DataBase***). Хранит данные приложения, такие как информация о пользователях, документах и тегах.

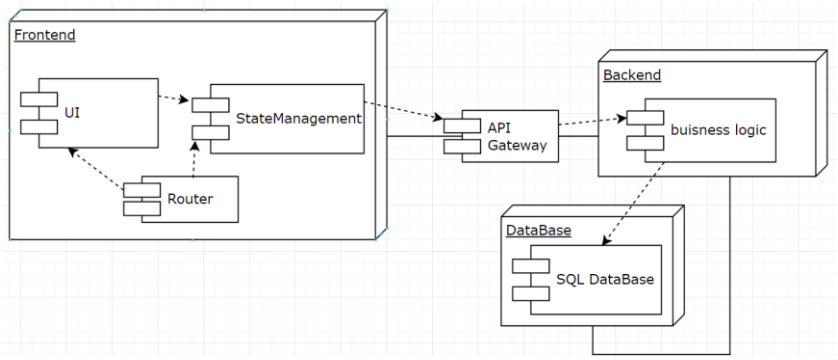


Рис. 6. Диаграмма компонентов.

Между компонентами организованы следующие связи (рис. 6):

1. Клиентский интерфейс взаимодействует с компонентами управления состоянием для отображения данных и реагирования на действия пользователей;

2. Компоненты управления состоянием: обращаются к сетевому слою для обмена данными с сервером, взаимодействуют с компонентами управления маршрутизацией для изменения отображаемого контента на основе текущего URL или действий пользователя и выполняют бизнес-логику и обработку данных, связанных с обучением и заданиями;

3. Сетевой слой взаимодействует с серверной частью для обработки запросов, аутентификации и обработки данных;

4. Серверная часть взаимодействует с БД для чтения и записи данных приложения.

Реализация приложения. Одной из основных задач данного исследования является настройка соединения с облачным сервисом (в частности рассматривается интеграция с Яндекс Диском) для получения доступа к файлам пользователя, хранящимся в облаке. Для установления стабильного соединения с Яндексом необходимо пройти несколько этапов:

1. **Регистрация приложения** в личном кабинете разработчика в Яндекс. На данном этапе были указаны запрашиваемые у пользователя права доступа к его личной информации: дате рождения, адресу электронной почты, логину, имени и фамилии, полу, номеру телефона, и права доступа к информации Яндекс Диска: доступ к папке приложения на Диске, чтение всего Диска, запись в любом месте на Диске, доступ к информации о Диске.

После регистрации приложения Яндекс сгенерировал ClientID – ключ приложения, требуемый для выполнения авторизации в приложении, и Client Secret – специальный ключ, с помощью которого можно расшифровывать данные, полученные от API Яндекса (рис.7).

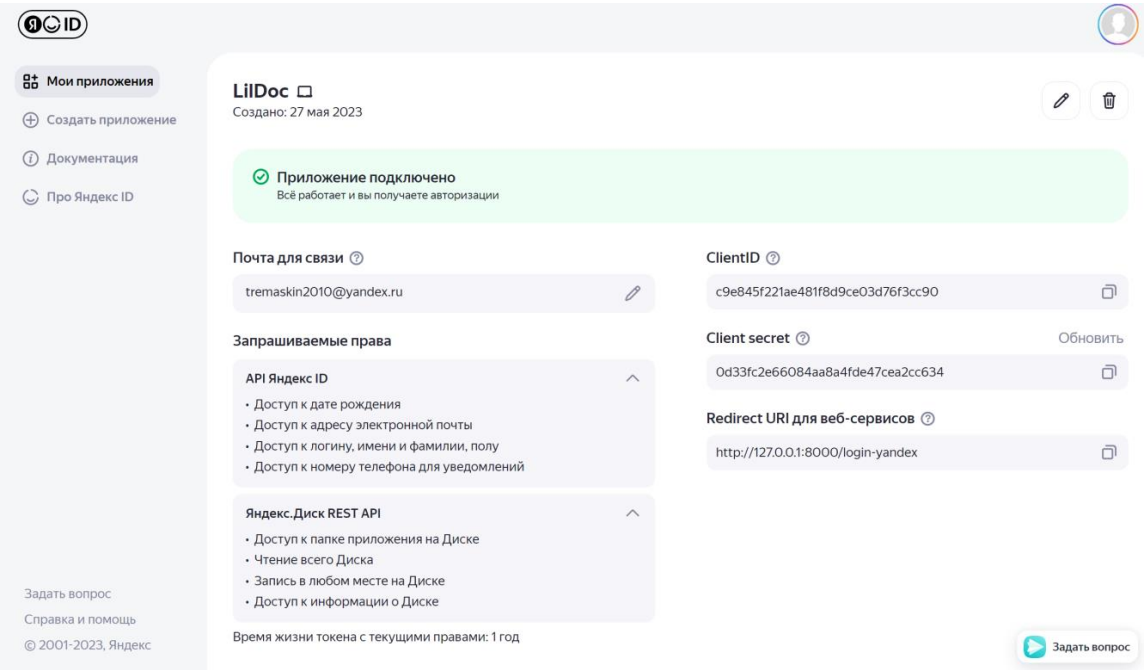


Рис. 7. Личный кабинет разработчика при регистрации приложения в Яндекс.

2. Получение OAuth-токена. Для того чтобы пользователь мог войти в разрабатываемое приложение через сервисы Яндекс, нужно настроить работу с протоколом OAuth. Для этого требуется получить OAuth-токен, который предоставляет возможность обращаться к сервисам Яндекса от имени конкретного пользователя, предоставившего соответствующие разрешения. Токен необходимо запрашивать для каждого нового пользователя, входящего через сервис Яндекс. Получить его можно несколькими способами: с помощью виджета LoginSDK (на текущий момент работает только для мобильных устройств) или используя отладочный токен. В работе был выбран вариант с использованием виджета от Яндекса. Для этого в html код страницы, содержащий виджет, добавляется код на языке JavaScript, загружающий кнопку с помощью конструкции iFrame. Данный код также позволяет управлять внешним видом кнопки и настройками авторизации в приложении.

Затем, на странице, принимающей токен, был добавлен скрипт, который «подтверждает» получение токена и загружает токен разрабатываемое приложение:

```
window.onload = function() {
    window.YaSendSuggestToken("http://127.0.0.1:8000");};
```

3. Получение пользовательской информации. После авторизации пользователя в системе с помощью Яндекса, появляется возможность отправлять запросы к API на: получение информации о пользователе, о конкретном файле, получение списка всех файлов на Диске, скачивание конкретного файла. Данные запросы осуществляются с помощью

встроенного в Laravel класса `Illuminate\Support\Facades\Http`. Он позволяет удобно конструировать запросы, задавать заголовки и передавать параметры.

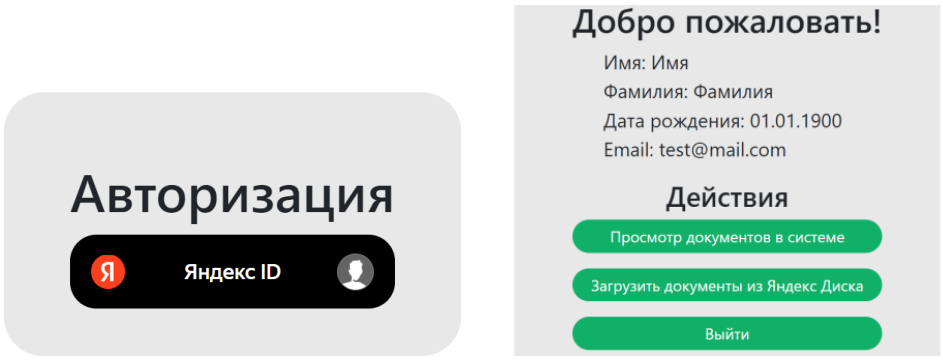
Далее была выполнена реализация основного функционала системы, заключающаяся в извлечении необходимых данных из Яндекс Диска в созданную БД, оптимизации работы с ней и организации взаимодействия между серверной частью и интерфейсом пользователя.

При разработке системы использовалась архитектуры MVC (Model-View-Controller), которая зарекомендовала себя как решение по созданию эффективной структуры приложений, позволяющее разделить бизнес-логику, работу с БД и визуальную составляющую проекта друг от друга. Это позволяет сделать код читабельным, а процесс разработки более комфортным, разграничивая работу frontend и backend разработчиков.

Для работы системы с целью аутентификации пользователя необходимо хранить и передавать с каждым запросом его ID. Также для удобства и уменьшения нагрузки на БД выполняется хранение и передача информации о токене Яндекса во время сессии.

Для реализации процесса загрузки файлов из Яндекс Диска, при получении OAuth-токена одновременно оформляется учетная запись пользователя в разрабатываемой системе. При этом выполняется проверка того, был ли пользователь авторизирован в системе ранее, и если это так – выполняется обновление информации о нем и запись данных в сессию.

Опишем алгоритм работы приложения со стороны пользователя. Работа с приложением начинается со страницы авторизации (рис. 8а), поскольку весь функционал приложения в соответствии с рисунком 4 доступен только зарегистрированным пользователям. После успешной авторизации открывается главная страница приложения, содержащая информацию о пользователе и кнопки с доступными действиями: просмотр документов в системе, загрузка документов из Яндекс Диска и выход (рис. 8б).



а) б)

Рис. 8. Страница приложения а) до авторизации и б) после успешной авторизации.

Далее рассмотрим страницу загрузки документов в систему из Яндекс Диска, где выводится список документов, кнопки для загрузки в систему всех файлов и выбранного документа (рис. 9). Для вывода списка файлов использовалась Blade директива @foreach.

а) б)

Рис. 9. Страницы приложения а) со списком документов из Яндекс Диска и б) с уведомлением об успешной загрузке документа.

Опишем алгоритм добавления тега документу. Его основная сложность заключается в определении того, как создавать тег: есть ли у пользователя нужные теги, подходят ли они, и как определить требуемый. Сначала выполняется запрос об отборе всех тегов, доступных пользователю. Он весьма ресурсоемок и деление его на части может замедлить систему, но при нормализации БД он будет выполняться быстрее. После обработки полученных данных определяется, нужно ли создавать новый тег или можно добавить тэг из имеющейся группы.

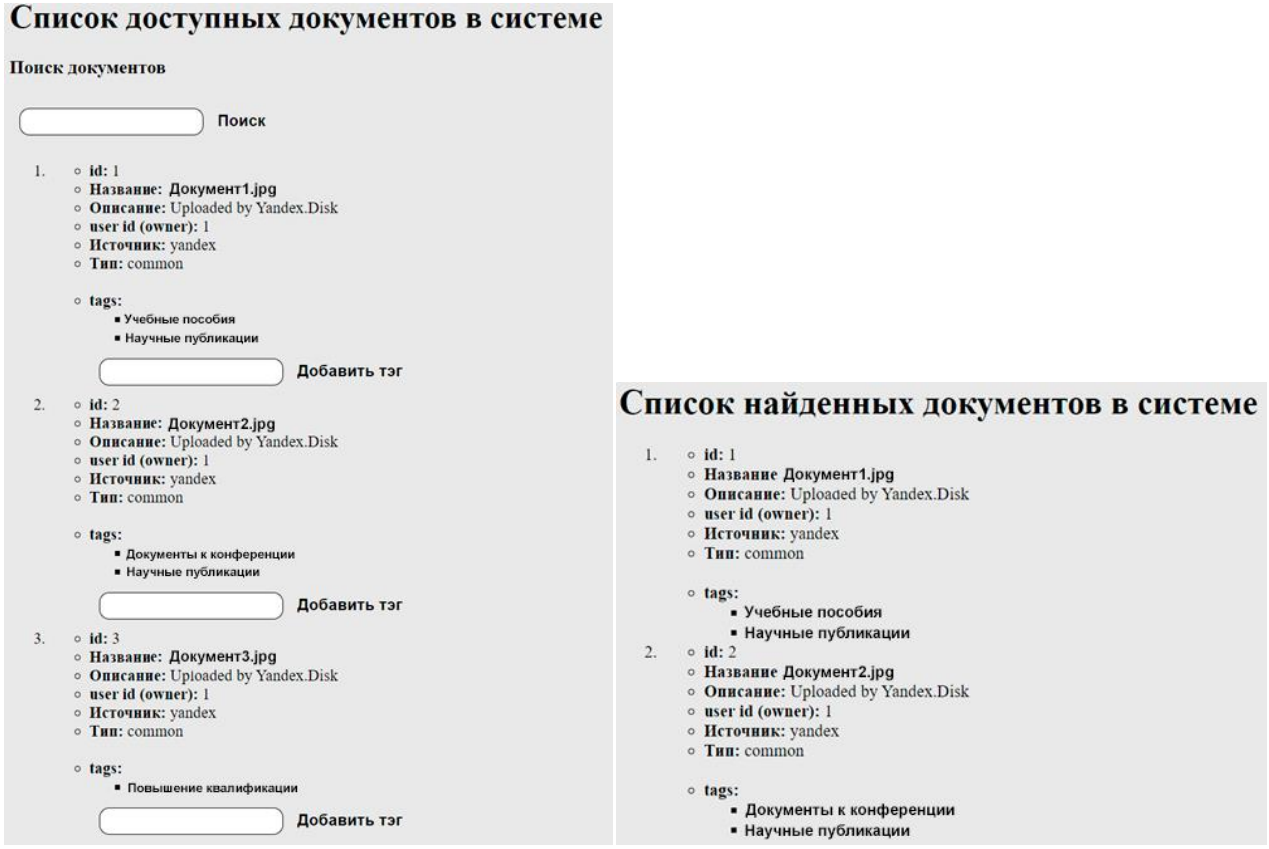
Поясним суть алгоритма поиска документов в системе: сначала выполняется проверка того, к каким группам документов есть доступ у пользователя, затем просматриваются теги, привязанные к этим группам и документов из этой группы. Можно было бы сформировать единый запрос, выполняющий описанные выше действия, однако он будет неэффективен при большом количестве данных, так как при этом потребуется выполнить обращение ко множеству таблиц. Поэтому для повышения эффективности запрос был разделен на 3 части:

1. Выбор групп документов, к которым пользователь имеет доступ. Этот запрос будет выполняться быстро, так как нет обращения к другим таблицам, и задается одно условие.

2. Поиск в описаниях документов. Выполняется обращение к таблицам document_group_data и documents, в которых ищется совпадение в заглавии и описании документа с использованием оператора like, позволяющего осуществлять поиск по шаблону.

3. Поиск по тегам. В данном запросе выполняется поиск совпадения в названиях и описаниях тегов по таблицам tags и user_tags с использованием оператора like.

Перейдем к странице с загруженными в систему документами, где отображается поле поиска документов, список документов с данными о каждом из них, и кнопки добавления тегов (рис.10а). Последняя созданная в приложении страница отображает список найденных документов. На рисунке 10б показан результат поиска по тегам «Научные публикации» и «Документы к конференции» для списка документов, отображенных на рисунке 10а.



а) б)

Рис. 10. Страницы со списком документов: а) доступных и б) найденных по тегам «Научные публикации» и «Документы к конференции».

Заключение. Все задачи, поставленные в рамках данного исследования, были решены. Результатом работы является разработанное веб-приложение, предназначенное для автоматизации процесса ведения архива личных документов пользователя с использованием системы тегов, обеспечивающих удобную и эффективную фильтрацию данных. Особенностью данного приложения, помимо использования системы тегов для организации поиска, является возможность хранения документов не только на устройствах пользователя, но и в облачном хранилище сервиса Яндекс Диск, с которым выполнена интеграция по API.

СПИСОК ЛИТЕРАТУРЫ

1. Онуфриенко С. Г., Хоняк Ю. А. Облачные хранилища данных // Материалы докладов 51-й Международной научно-технической конференции преподавателей и студентов в двух томах (г. Витебск, 25 апреля 2018 года): Том 2. – Витебск: ВГТУ, 2018. – С. 63–66.
2. Хромушкин Р. Р. Оптимизация работы MYSQL (innodb и myisam) // ADVANCED SCIENCE: сборник статей X Международной научно-практической конференции (г. Пенза, 12 ноября 2019 года). – Пенза: "Наука и Просвещение" (ИП Гуляев Г.Ю.), 2019. – С. 54–56.

ПОТАПКИНА Ю. Ю., ПЕСКОВА Е. Е.[1]

О ПРИМЕНЕНИИ WENO-СХЕМ К МОДЕЛИРОВАНИЮ РЕАГИРУЮЩИХ ГАЗОВЫХ ПОТОКОВ

Аннотация. В работе исследованы WENO-схемы 5 и 7 порядка с оптимальными весовыми коэффициентами без учета индикаторов гладкости решения и с индикаторами гладкости. Показано, что при их применении к решению задач многокомпонентной газовой динамики с химическими реакциями, диффузией, вязкостью и теплопроводностью WENO-схема 5 порядка с оптимальными весовыми коэффициентами дает более точный результат при меньшей трудоемкости вычислений.

Ключевые слова: математическое моделирование, уравнения Навье-Стокса, WENO-схемы, дозвуковые реагирующие потоки.

POTAPKINA YU. YU., PESKOVA E. E.

ON THE APPLICATION OF WENO SCHEMES TO MODELING REACTING GAS FLOWS

Abstract. The 5th and 7th order WENO schemes with optimal weight coefficients without solution's smoothness indicators and with solution's smoothness indicators are investigated in the article. It is shown that the 5th order WENO scheme with optimal weight coefficients gives a more accurate result with less complexity of calculations to solve problems of multicomponent gas dynamics with chemical reactions, diffusion, viscosity and thermal conductivity.

Keywords: mathematical modeling, Navier-Stokes equations, WENO schemes, subsonic reactive flows.

Введение. В настоящее время широкое распространение получило исследование газодинамических течений в задачах химической промышленности, поскольку технологам для получения целевых продуктов необходимо знать большое количество параметров проведения реакции. Наиболее значимым методом исследования стало математическое моделирование, которое позволяет рассмотреть поведение различных реакций в разных условиях без проведения лабораторных экспериментов. Аппарат математического моделирования расширяется и улучшается в прямой зависимости от появления новых задач: создаются новые математические модели, эффективные вычислительные алгоритмы, повышается точность расчетов. Настоящая работа направлена на исследование применения схем высокого порядка точности (WENO-схема 5 и 7 порядка) к решению задач

[1] Исследование Песковой Е. Е. выполнено за счет гранта Российского научного фонда № 23-21-00202.

многокомпонентной газовой динамики с учетом вязкости, диффузии, теплопроводности и химических реакций.

Математическая модель и численный алгоритм. Поскольку нас интересует исследование возможности повышения порядка точности расчетов с помощью WENO-схем, в настоящей работе мы рассматриваем одномерную модель уравнений вязкой дозвуковой динамики химически активной газовой смеси:

$$\frac{\partial U}{\partial t} + \frac{\partial (F(U) - H(U))}{\partial x} = W(U).$$

Векторы $U, F(U), H(U), W(U)$ имеют следующий вид:

$$U = \begin{pmatrix} \rho Y_m \\ \rho u \\ \rho h \end{pmatrix}, F(U) = \begin{pmatrix} \rho u Y_m \\ \rho u^2 \\ \rho h u \end{pmatrix}, H(U) = \begin{pmatrix} J_{mx} \\ \tau_{xx} \\ q_x \end{pmatrix}, W(U) = \begin{pmatrix} R_m \\ 0 \\ 0 \end{pmatrix}.$$

$$U = \begin{pmatrix} \rho Y_m \\ \rho u \\ \rho h \end{pmatrix}, F(U) = \begin{pmatrix} \rho u Y_m \\ \rho u^2 \\ \rho h u \end{pmatrix}, H(U) = \begin{pmatrix} J_{mx} \\ \tau_{xx} \\ q_x \end{pmatrix}, W(U) = \begin{pmatrix} R_m \\ 0 \\ 0 \end{pmatrix}.$$

В этой системе уравнений $m = 1, \dots, M$, M - число компонент в газовой смеси, ρ - плотность смеси, Y_m- массовая доля -ой компоненты смеси, u - скорость, h - энтальпия смеси, J_{mx} - диффузионный поток, R_m - скорость образования или расхода m-ой компоненты смеси, $\pi = p - p_0$ - динамическая составляющая давления, p_0 - термодинамическая составляющая давления, постоянная в области, τ_{xx} – вязкий поток, q_x - потока тепла.

Система дополняется условием на дивергенцию вектора скорости:

$$\nabla \cdot \vec{u} = \frac{1}{\rho C_p T}\left(\nabla \cdot \lambda \nabla T + \sum_m \rho D_{m,mix} \nabla Y_m \nabla h_m\right) + \frac{1}{\rho}\sum_m \frac{M_w}{M_{wm}}\left(\nabla \cdot \rho D_{m,mix} \nabla Y_m\right) + \frac{1}{\rho}\sum_m \left(\frac{M_w}{M_{wm}} - \frac{h_m}{C_p T}\right) R_m.$$

Здесь C_p – теплоемкость смеси при постоянном давлении, T – температура смеси, λ – теплопроводность смеси, $D_{m,mix}$ – коэффициент диффузии, h_m – энтальпия образования компоненты смеси, M_w – молекулярная масса смеси, M_{wm} – молекулярная масса компоненты смеси. Более подробно математическая модель представлена в работе [1].

Для построения вычислительного алгоритма используем равномерную сетку отрезков:

$$\Omega_{\Delta x} = \{\Delta_i, i = 1, \dots, N, \Delta_i = [x_{i-1}, x_i], |\Delta_i| = x_i - x_{i-1} = h_x, h_x N_x = L_x\}$$

Численный алгоритм строится по схеме расщепления по физическим процессам [2]. На первом этапе в этом расщеплении решается система уравнений химической кинетики, на втором этапе интегрируются уравнения законов сохранения без учета давления, далее рассчитывается поле поправок к давлению из решения уравнения Пуассона, на последнем

этапе корректируются поле давления и поле скорости [2]. Поскольку мы исследуем применение WENO-схем, приведем подробно алгоритм второго этапа. Для интегрирования законов сохранения используется разностная схема вида:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{F_{i+1/2} - F_{i-1/2}}{h_x} - \frac{H_{i+1/2} - H_{i-1/2}}{h_x} = 0.$$

Здесь $H_{i+1/2},\ H_{i-1/2}$ – диффузионные, вязкие и тепловые потоки на границе i и $i+1$, $i-1$ и i ячейками соответственно, которые рассчитываются по схеме с центральными разностями, $F_{i+1/2},\ F_{i-1/2}$ – конвективные потоки на границе i и $i+1$, $i-1$ и i ячейками соответственно, которые рассчитываются с использованием потоков Русанова [3]:

$$F_{i+1/2} = 0.5\left(F\left(U_{i+1/2}^r\right) + F\left(U_{i+1/2}^l\right) - \alpha\left(U_{i+1/2}^r - U_{i+1/2}^l\right)\right),$$

$$\alpha = max\left(|u_{i+1/2}^r|, |u_{i+1/2}^l|\right),$$

где $U_{i+1/2}^r$ и $U_{i+1/2}^l$ – значения вектора переменных U справа и слева от границы между i и i+1 ячейками, которые рассчитываем с использованием WENO схем 5 и 7 порядка точности [4; 5]. Их суть заключается в следующем. Для нахождения значения вектора переменных U справа и слева от границы между i и i+1 используется выражение:

$$U_{i+1/2} = \sum_{\mathrm{v}=1}^{K} \Omega^{(\mathrm{v})}\, U_{i+1/2}^{(\mathrm{v})}.$$

Здесь $K = 3$ для случая схемы 5-го порядка, $K = 4$ для схемы 7-го порядка точности, $\Omega^{(\mathrm{v})}$ – оптимальные весовые коэффициенты, $U_{i+1/2}^{(\mathrm{v})}$ – значение на границе, полученное на шаблоне $S = \{x_{i-\mathrm{v}}, \dots, x_i, \dots, x_{i-\mathrm{v}+\mathrm{K}-1}\}$:

$$U_{i+1/2}^{(\mathrm{v})} = \sum_{p=0}^{K-1} c_{\mathrm{vp}} U_{i-\mathrm{v}+\mathrm{p}}.$$

В случае использования весовых коэффициентов с индикаторами гладкости значения вектора переменных U справа и слева от границы между i и i+1 находятся из выражения:

$$U_{i+1/2} = \sum_{\mathrm{v}=1}^{K} \omega^{(\mathrm{v})}\, U_{i+1/2}^{(\mathrm{v})}, \omega^{(\mathrm{v})} = \frac{\sigma^{(\mathrm{v})}}{\sum_{\mathrm{v}=1}^{K} \sigma^{(\mathrm{v})}}, \sigma^{(\mathrm{v})} = \frac{\Omega^{(\mathrm{v})}}{[\varepsilon + IS^{(\mathrm{v})}]^p}.$$

Здесь $IS^{(\mathrm{v})}$ – это индикаторы гладкости, ε – некоторое малое число, вводимое, чтоб предотвратить деление на ноль. Величины c_{vp}, $\Omega^{(\mathrm{v})}$, $IS^{(\mathrm{v})}$ для $K = 3$, $K = 4$ приведены в работах [4; 5].

Вычислительные эксперименты. Рассмотрим следующую одномерную постановку задачи. Размер области 0.2 м, шаг по пространству $2 \cdot 10^{-3}$ м, шаг по времени $1 \cdot 10^{-5}$ с, расчет ведем до 0.2 секунд. Принимаются следующие начальные данные: в области от 0 м до 0.1 м

температура газа 800ºС; в области от 0.1 м до 0.2 м температура газа 1200ºС; скорость потока 0.1 м/с; давление 101325Па; состав газовой смеси – метан 100% (CH4). Граничные условия: на границе слева со скоростью 0.1 м/с втекает метан с температурой 800ºС, на границе справа задаются условия вытекания. Такая постановка задачи принята для того, чтобы выяснить какая из схем будет лучше считать в области перепада температур. В этой же области начнут происходить интенсивные реакции с расходом метана и образования продуктов реакции.

Рассматриваем 3 варианта расчета значений вектора U на границе ячеек: 1) схемой первого порядка точности, т.е. $U^r_{i+1/2} = U_{i+1}$, $U^l_{i+1/2} = U_i$; 2) WENO схемой 5-го порядка аппроксимации; 3) WENO-схемой 7-го порядка аппроксимации. На рисунках 1 и 2 представлены распределения температуры и метана. Из графиков можно сделать вывод, что схема первого порядка аппроксимации сглаживает решение в областях резкого изменения газодинамических параметров, разница температур составляет величину 10ºС. Такой результат является достаточно большим расхождением в случае проведения лабораторных экспериментов. Расчет по схемам WENO 5-го и 7-го порядка практически совпадает, что говорит о преимуществе схемы WENO5, поскольку она использует более компактный шаблон. На рисунках 9 и 10 представлены расчеты по WENO-схемам с оптимальными весами и с расчетом индикаторов гладкости. Из графиков можно сделать вывод, что для рассматриваемых задач использование WENO-схем с оптимальными весами является преимущественным, поскольку графики полностью совпадают, а они являются менее трудоемкими. Полученный результат полного совпадения графиков можно объяснить преобладанием диффузионных процессов над конвективным переносом.

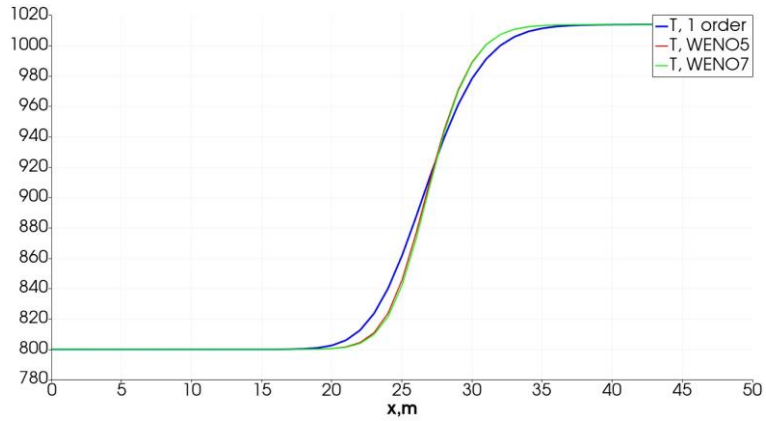


Рис. 1. Распределение температуры. Расчет по схемам 1-го порядка (синяя линия), WENO5 (красная линия), WENO7 (зеленая линия).

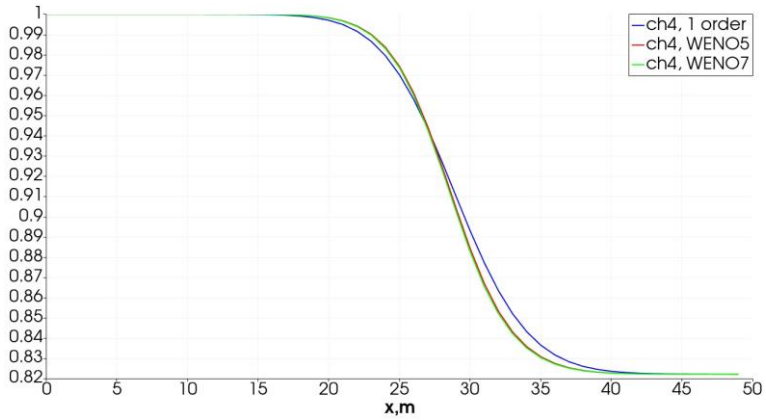


Рис. 2. Распределение метана. Расчет по схемам 1-го порядка (синяя линия), WENO5 (красная линия), WENO7 (зеленая линия).

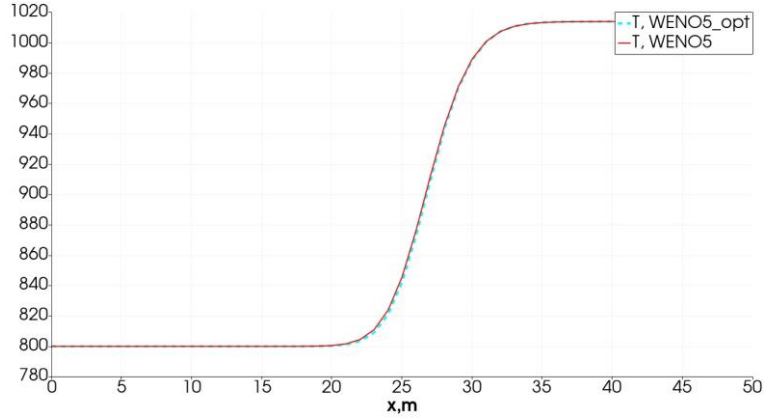


Рис. 3. Распределение температуры. Расчет по схеме WENO5 (красная линия) и схеме WENO5 с оптимальными весами (бирюзовый пунктир).

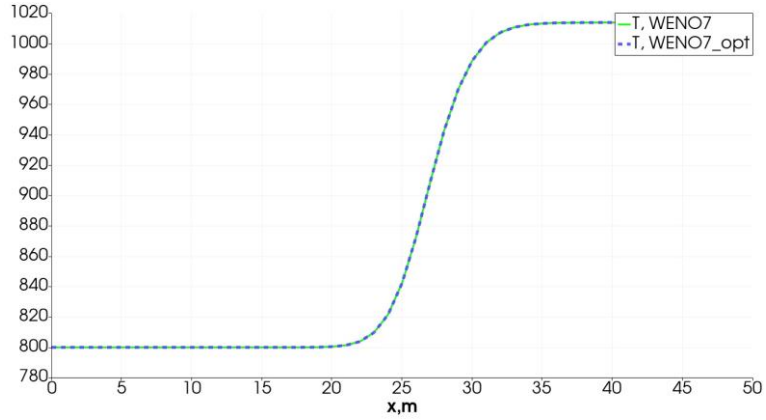


Рис. 4. Распределение температуры. Расчет по схеме WENO7 (зеленая линия) и схеме WENO7 с оптимальными весами (синий пунктир).

Выводы. В работе проведено исследование применения WENO-схем 5-го и 7-го порядка точности с оптимальными весовыми коэффициентами и с весовыми коэффициентами, рассчитанными с использованием индикаторов гладкости к задачам многокомпонентных

реагирующих течений с вязкостью, диффузией и теплопроводностью. Показано, что по сравнению со схемами первого порядка точности WENO-схемы меньше сглаживают решение в областях резкого изменения газодинамических параметров и концентраций компонент смеси. Сравнение результатов вычислительных экспериментов, в которых весовые коэффициенты в WENO-схеме принимались оптимальными и рассчитывались с индикаторами гладкости показало, что результаты расчетов совпадают. Таким образом, можно сделать вывод о преимуществе использования оптимальных весовых коэффициентов в WENO-схеме, поскольку в данном случае алгоритм является менее трудоемким.

СПИСОК ЛИТЕРАТУРЫ

1. Жалнин Р. В., Пескова Е. Е., Стадниченко О. А., Тишкин В. Ф. Моделирование течения многокомпонентного химически активного газа на примере пиролиза углеводородов // Препринты ИПМ им. М.В. Келдыша. – 2017. – № 101. – 16 с.
2. Пескова Е. Е., Снытников В. Н. Численное исследование конверсии метановых смесей под воздействием лазерного излучения // Журнал Средневолжского математического общества. – 2023. – Т. 25, № 3. – С. 159–173.
3. Русанов В. В. Расчет взаимодействия нестационарных ударных волн с препятствиями // Журнал вычислительной математики и математической физики. – 1961. – Т. 1, № 2. – С. 267–279.
4. Shu C.-W. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws // Advanced Numerical Approximation of Nonlinear Hyperbolic Equations. – 2006. – Vol. 1697. – P. 325–432.
5. Евстигнеев Н. М. О построении и свойствах WENO-схем пятого, седьмого, девятого, одиннадцатого и тринадцатого порядков. Часть 1. Построение и устойчивость // Компьютерные исследования и моделирование. – 2016. – Т. 8, № 5. – С. 721–753.

ГЕРАСИМОВ А. Д., ФИРСОВА С. А.

РАЗРАБОТКА ПРОГРАММНО-ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ОЦЕНКИ ТОЧНОСТИ И НАДЕЖНОСТИ ЧАСОВЫХ МЕХАНИЗМОВ

Аннотация. В статье описывается разработанная авторами программно-информационная система, предназначенная для использования в мастерских по ремонту и обслуживанию механических часов. Особенностями предлагаемой системы являются: оценка точности часовых механизмов методом анализа звука хода часов, а также оценка их надежности с помощью статистического анализа ремонтных работ большой выборки механизмов, проходящих через часовой сервис.

Ключевые слова: программно-информационная система, часовой сервис, часовой механизм, звуковой анализ, фильтр Винера, точность хода часов.

GERASIMOV A. D., FIRSOVA S. A.

DEVELOPMENT OF SOFTWARE AND INFORMATION SYSTEM FOR EVALUATION OF ACCURACY AND RELIABILITY OF CLOCK MECHANISMS

Abstract. The article describes the software and information system developed by the authors and designed for use in workshops for the repair and maintenance of mechanical watches. The features of the proposed system include an assessment of the accuracy of clock mechanisms by analyzing the sound of the watch as well as an assessment of their reliability using statistical analysis of repair work of a large sample of mechanisms passing through the watch service.

Keywords: software and information system, clock service, clock mechanism, sound analysis, Wiener filter, clock running accuracy.

В современном мире необходимость в механических часах практически отпала, потому что почти у каждого в кармане есть смартфон, но несмотря на это многие люди покупают и носят наручные часы, так как наручные часы давно перестали были лишь прибором для определения времени. Если раньше главными качествами часов считались точность и долговечность (часто они передавались из поколения в поколение), то сейчас главное – бренд и дизайн [1].

В настоящее время в мире растет спрос на «люксовые» механические часы. Аналитики отмечают серьезный рост интереса к механическим часам среди миллениалов, которым нравится возможность проверить время, не наткнувшись при этом на уведомления из мессенджеров и соцсетей [2]. Экспорт часов высокой ценовой категории из Швейцарии в 2022 году достиг восьмилетнего максимума. При этом, производители наручных часов не

чувствуют угрозу со стороны производителей электронных устройств. Более того, некоторые из них полагают, что «умные» часы могут даже подстегнуть спрос на механические модели.

В 2020 году мировой рынок наручных часов оценивался в $60 млрд. По прогнозам аналитиков, к 2027-му он достигнет $73,4 млрд, из которых $48,1 млрд придется на кварцевые часы.

В 2021 году в России было продано более 4 млн наручных часов. Доля импорта составила более 96%, или $307 млн. Ведущими брендами, представленными на рынке наручных часов, стали Casio, Audemars Piguet, Rolex, Patek Philippe, TAG Heuer, Tissot, ЗАО ЧЧЗ «Восток», Breguet. В натуральном выражении основную долю в объеме импорта заняли кварцевые и электронные часы (94%), при этом в денежном выражении лидируют механические изделия из Швейцарии (81,6%). В России крупнейшим производителем часов остается завод «Восток». На него приходится 19,3% от общего объема производства.

Рассматриваемая в данной статье программно-информационная система предназначена для оценки точности и надежности функционирования часовых механизмов наручных часов. Для оценки точности хода механических часов использовался метод звукового анализа аудиофайла, полученного в результате выполнения аудиозаписи звука хода часов. Для оценки надежности приводится статистика обращений пользователей и результатов их обслуживания в сервисном центре по ремонту и обслуживанию часов.

Статистическая информация, собранная с помощью разработанного программного обеспечения, поможет потенциальному покупателю получить информацию о среднем сроке безотказной работы той модели часов, которую он планирует приобрести, оценить точность их работы, а также сравнить понравившуюся модель с другими вариантами по указанным выше параметрам, что позволит ему более осознано сделать свой выбор.

Сервисным организациям по ремонту и обслуживанию часов применение разработанной системы поможет упростить и снизить стоимость производимых работ, так как мастер будет заранее иметь представление о типовых проблемах конкретного часового механизма, а также просмотреть историю предыдущих поломок аналогичных механизмов. Таким образом мастером может заранее оценить стоимость ремонта механических наручных часов еще до их вскрытия и осмотра, что сокращает затраты времени на обслуживание клиентов. Указанные выше аспекты определяют актуальность и практическую значимость разработки данной системы.

Программно-информационная система, реализованная в виде веб-приложения, подразумевает работу с тремя типами пользователей – Администратор, Работник часового сервиса, Клиент. Основные возможности, предоставляемые системой для каждого типа пользователей, отображены на диаграмме вариантов использования (см. рис. 1).

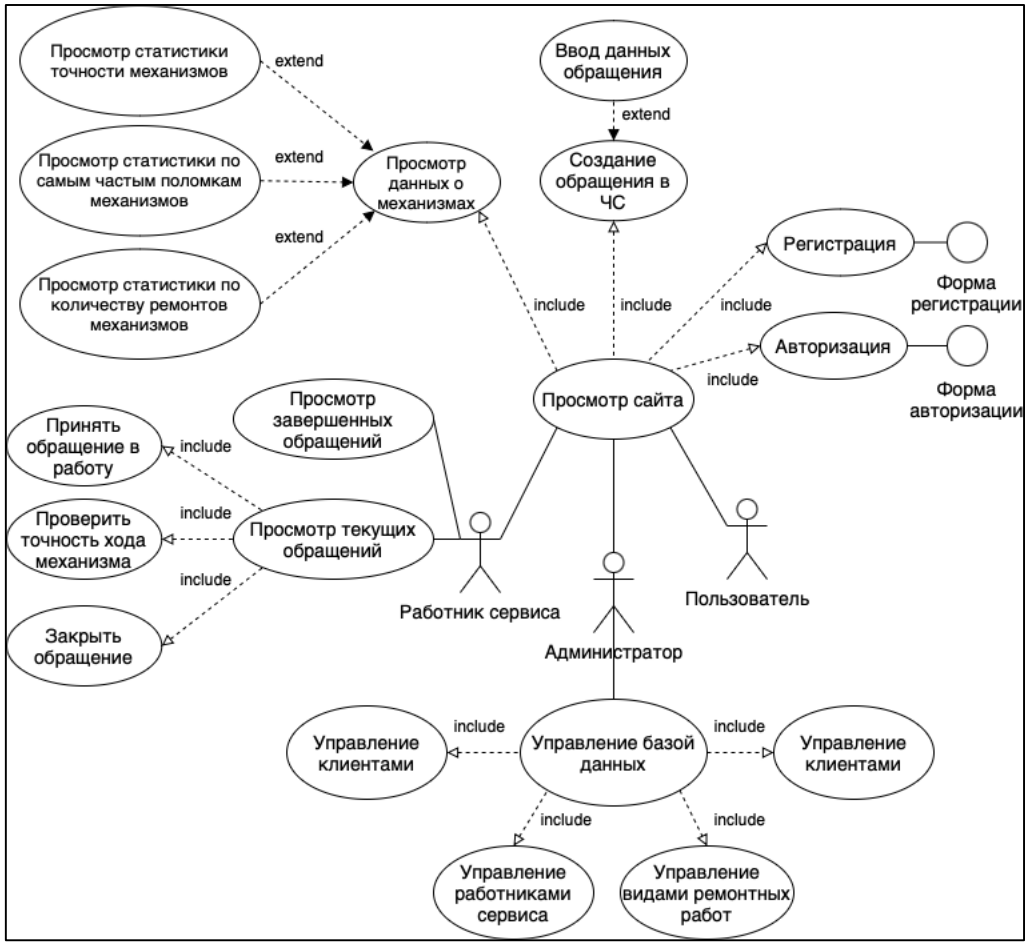


Рис. 1. Диаграмма вариантов использования.

Диаграмма развертывания системы представлена на рисунке 2.

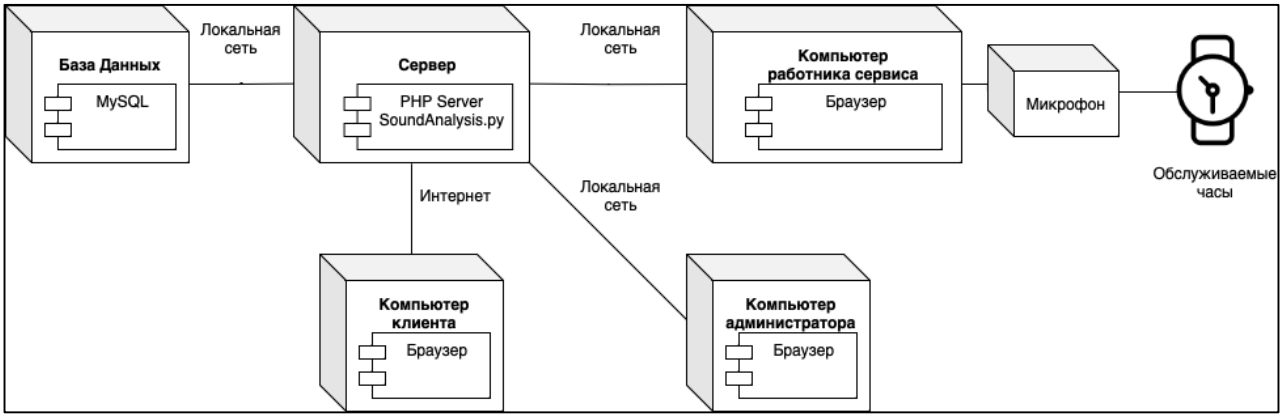


Рис. 2. Диаграмма размещения.

Пользователь заходит на сайт часового сервиса со своего компьютера по сети Интернет (Клиент) или по локальной сети (Администратор, Работник сервиса), регистрируется или авторизуется на сайте. Сайт размещен на сервере, который имеет прямой доступ к базе данных MySQL, хранящей всю необходимую информацию о работе часового

сервиса. Пользователь в соответствии со своими полномочиями и функциональными возможностями, определенными диаграммой вариантов использования, осуществляет запросы к базе данных. Также на сервере выполняется скрипт SoundAnalysis.py для звукового анализа присылаемых на него аудиофайлов. Эти аудиофайлы получает Работник сервиса в результате записи звука хода обслуживаемых часов через микрофон. После проведения звукового анализа сервер отправляет результат проверки точности хода Работнику Сервиса, а также сохраняет его в базу данных.

Рассмотрим набор средств, использованных при реализации программно-информационной системы. Для серверной части был выбран язык программирования PHP, такой выбор связан с простотой его использования и отличной работой с парадигмой «клиент-сервер». Для клиентской части сервиса использовался классический набор фронтенд-разработки: JavaScript, HTML, CSS; хранение данных производилось в БД MySQL. Для обработки и анализа звукового сигнала на сервере импортировались библиотеки Scipy и Numpy языка программирования Python. В качестве среды разработки применялся редактор Visual Studio Code, который является нетребовательным к ресурсам и, благодаря огромному количеству пользовательских расширений, обладает большим функционалом и поддерживает большое количество языков программирования.

Рассмотрим основные действия пользователей при работе с программно-информационной системой.

После авторизации пользователь попадает на главную страницу, на которой представлен прайс-лист на услуги часового сервиса с возможностями сортировки и поиска по различным критериям, а также экспортом результата в pdf-формат (см. рис. 3).

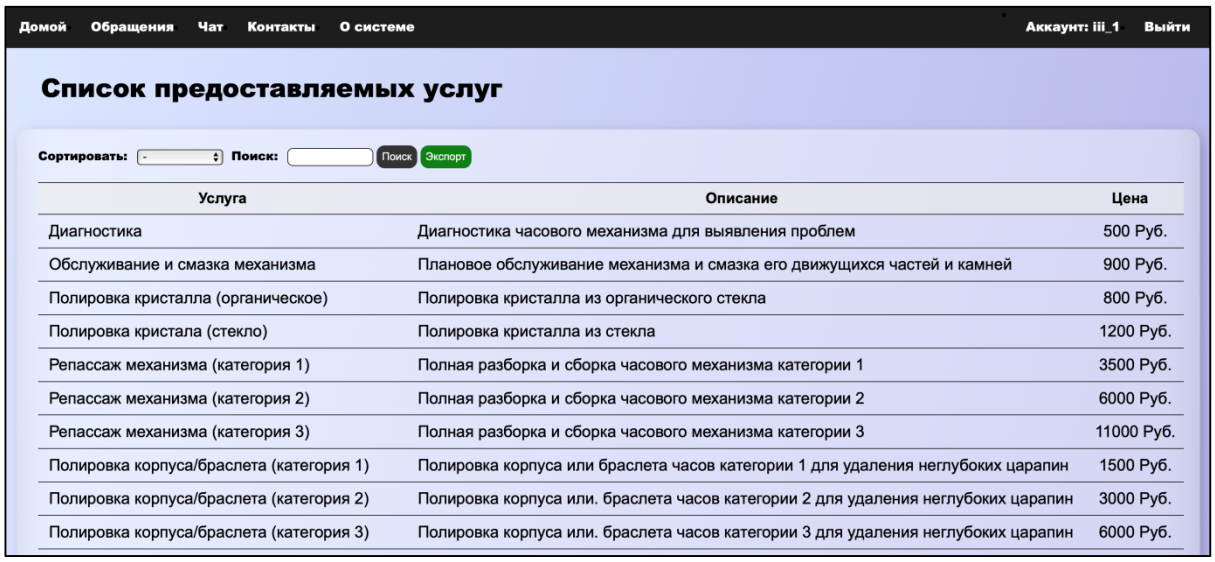


Рис. 3. Вид главной страницы после авторизации пользователя.

При переходе на страницу «Обращения» пользователь может создать свое обращение, в котором необходимо указать модель механизма и причину обращения в сервис (см. рис. 4).

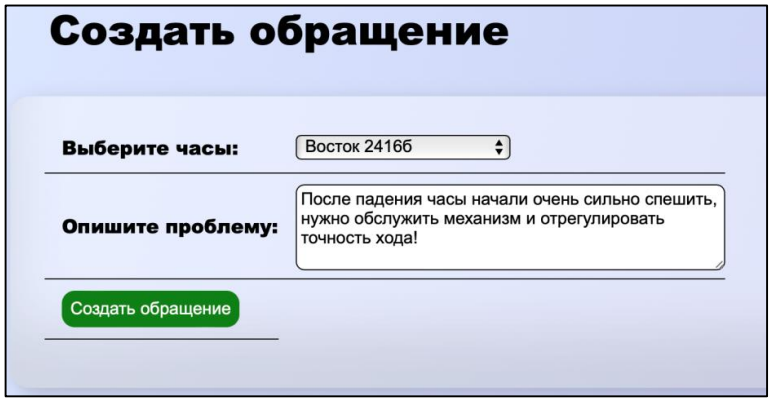


Рис. 4. Создание обращения.

Работник сервиса просматривает список пользовательских обращений; при нажатии на кнопку «Подробнее» происходит переход на страницу с детальной информацией по выбранному обращению, здесь же работник может принять обращение в работу (см. рис. 5).

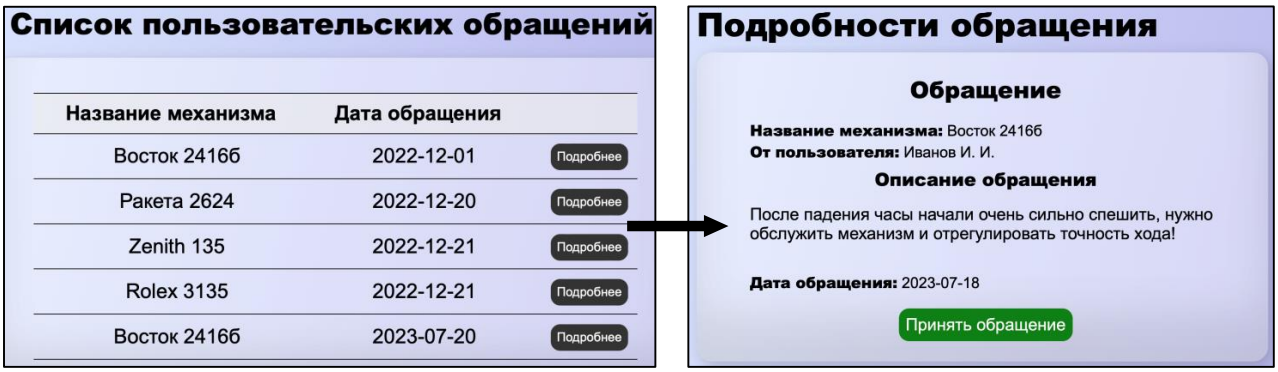


Рис. 5. Просмотр списка обращений в сервис.

В личном кабинете работника представлен список выполненных и текущих обращений с указанием стоимости проведенных работ (см. рис. 6).

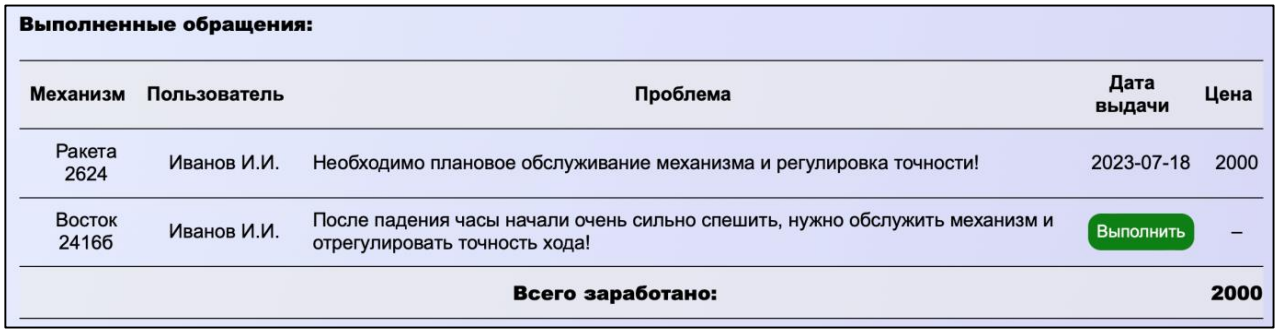


Рис. 6. Список обращений работника.

После завершения работы над обращением необходимо добавить предоставленные в ходе ремонта услуги (см. рис. 7).

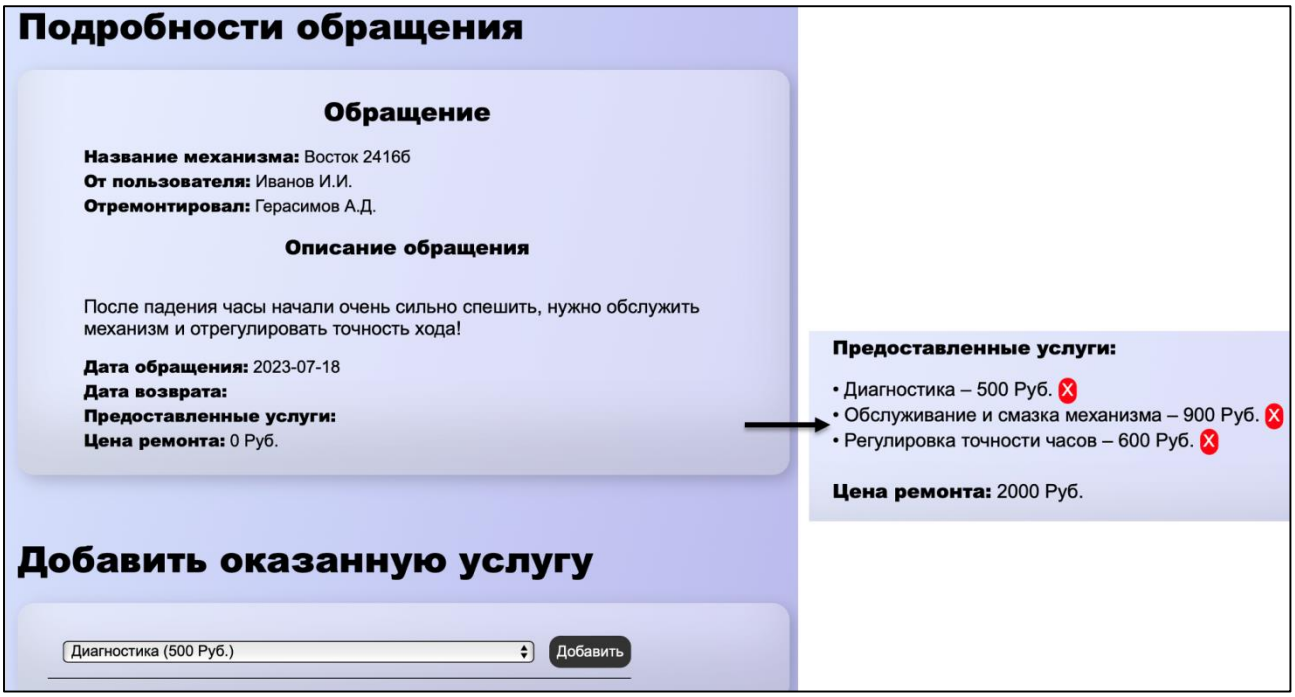


Рис. 7. Завершение работы с обращением.

При этом в личном кабинете клиента отображается информация о выполненном обращении (см. рис. 8).

Ваши обращения:

Механизм	Проблема	Выполнен	Работник	Цена
Восток 2416б	Требуется небольшое обслуживание механизма, поскольку точность часов сильно выбивается из значений в документах	2022-11-07	Григорьев А.О.	900
Ракета 2624	Необходимо плановое обслуживание механизма и регулировка точности!	2023-07-18	Герасимов А.Д.	2000
Восток 2416б	После падения часы начали очень сильно спешить, нужно обслужить механизм и отрегулировать точность хода!	2023-07-18	Герасимов А.Д.	2000
	Всего потрачено:			**4900**

Рис. 8. Информация о выполненном обращении.

Важной функциональной возможностью разработанной программно-информационной системы является оценка точности хода часов. Для этого, как правило, в часовых мастерских используется специализированные устройства, например, тестер для проверки параметров механических часов TIMEGRAPHER MTG-1000. В процессе тестирования часов на ЖК-дисплее отображаются диаграмма хода часов, цифровые данные

измерения точности хода, амплитуды и погрешности хода. Принцип работы этого тестера основывается на анализе звука хода часового механизма, вычислении ошибки хода механизма за один удар и экстраполяции результата для получения отклонения за сутки.

Помимо звукового анализа существуют и другие способы оценки точности хода часов. Один из них – это использование нескольких фотографий часового циферблата с информацией о точном времени сделанных фотографий для нахождения дрейфа показания часов от эталонного времени [3].

Описанная в данной статье программно-информационная система использует комбинированный метод оценки точности часов: при наличии микрофона оценка производится посредством звукового анализа хода часов, при его отсутствии – используется метод сравнения показываемого часами времени с эталонным для вычисления дрейфа за определенный период времени. Это позволяет точно и быстро оценивать ход механизма без необходимости применения дополнительного оборудования.

После окончания ремонта работник должен проверить точность хода часового механизма. Наиболее простой способ – ввести время, показываемое часами, вручную в поля «час» и «минута», после чего оно будут соотнесено с реальным временем для расчета отклонения хода часов. Эти показатели помогут работнику сравнить точность хода механизма до и после ремонта (см. рис. 9).

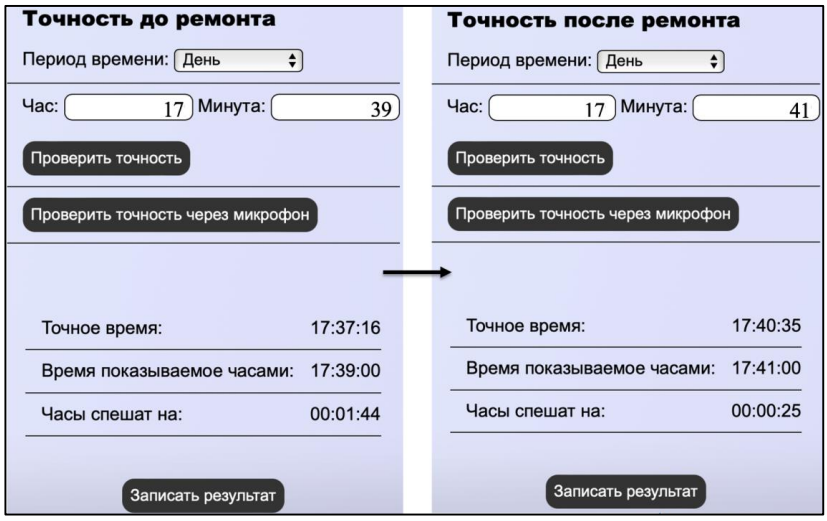


Рис. 9. Получение отклонения хода часов до и после ремонта.

Второй способ — это оценка точности через микрофон компьютера. Работнику необходимо приложить часы к микрофону и начать запись, после чего аудиофайл будет отправлен на сервер для дальнейшей обработки (см. рис. 10).

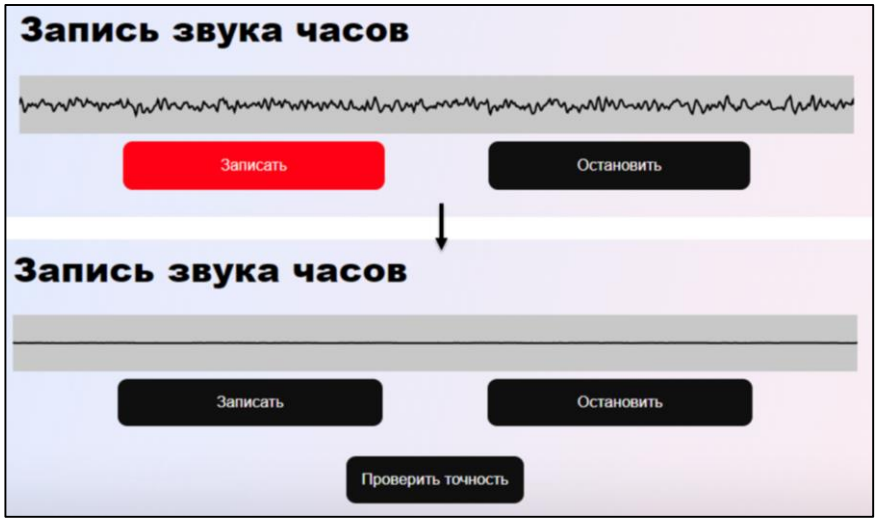


Рис. 10. Процесс записи звука.

На сервере аудиофайл редактируется: на него накладывается фильтр высоких частот, который позволяет избавиться от низкочастотных шумов, и фильтр Винера для сглаживания звуковой волны.

Фильтрация высоких частот происходит путем плавного подавления сигнала, частота которого находится ниже заданного порога. В данном случае используется реализация этого фильтра в виде функции signal.butter() из библиотеки scipy [4]. Она принимает в качестве аргументов порядок фильтрации, который влияет на степень фильтрации, пороговую частоту для фильтрации, частоту дискретизации аудиофайла и тип фильтрации. В качестве выходных данных эта функция возвращает параметры фильтрации, принимаемые в качестве аргументов функции signal.filtfilt(), которая накладывает фильтрацию на сигнал, наряду с данными, которые необходимо отфильтровать (см. рис. 11).

```
def butter_highpass_filter(data: list[float], highcut_freq: float, fs: float, order: int = 5):
    b, a = signal.butter(order, highcut_freq, fs=fs, btype='high')
    return signal.filtfilt(b, a, data)
```

Рис. 11. Реализация фильтра высоких частот.

Винеровская фильтрация заключается в оценке шума в аудиофайле и вычитании его из исходного сигнала для сглаживания звуковой волны. В данном случае используется реализация этого фильтра в виде функции signal.wienier() из библиотеки scipy. В качестве аргумента эта функция принимает исходный сигнал, а возвращает сглаженный сигнал в том же формате (см. рис. 12).

```
def applyWiener(self, data):
    wData = signal.wiener(data)
    return wData
```

Рис. 12. Реализация фильтра Винера.

После обработки аудиофайла начинается его анализ. Для этого в аудиофайле находятся все пики, являющиеся ударами часового механизма. Информация об их положении во времени сохраняется в отдельный массив, который далее будет анализироваться.

Для нахождения пиков используется функция signal.find_peaks() из библиотеки scipy. В качества аргументов она принимает исходный сигнал, предполагаемое расстояние между пиками и их высоту (см. рис. 13).

```
def peakSignalDetection(self, data1D):
    return ss.find_peaks(data1D, distance=0.9*3600*self.sampleRate/self.reffrequency, height=800)
```

Рис. 13. Код функции нахождения пиков в аудиофайле.

В качестве выходных данных эта функция возвращает массив индексов всех пиков из массива исходного сигнала. Визуализация результата, на которой найденные индексы пиков отмечены крестиками и наложены на исходный сигнал, представлена на рисунке 14.

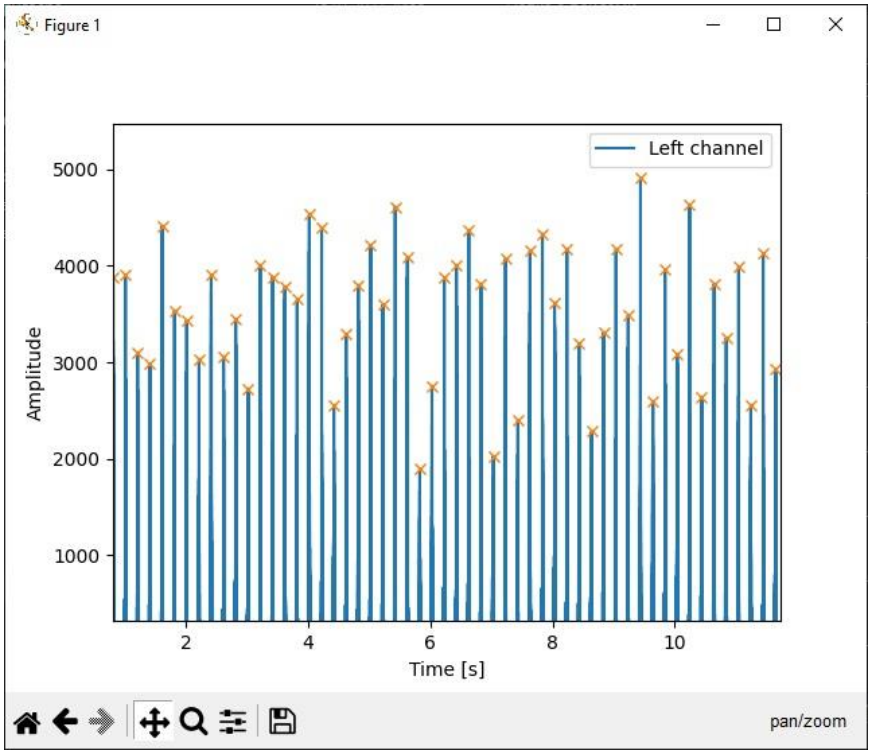


Рис. 14. Визуализация результата нахождения всех пиков в аудиофайле.

Информация о пиках обрабатывается для получения ошибки хода часов за одни удар. Для этого находится среднее отклонение в первом ударе (называемый «тик») и втором ударе (называемый «ток») от эталонного. Таким образом, мы получаем среднее отклонение хода за один удар в миллисекундах (см. рис. 15).

```
def getBeatError(self, peaks):
        # Создание массива для "тика" и "тока"
        teven = []
        todd = []
        oneFramemSec = 1000/self.sampleRate
        previous = 0
        for i, pi in enumerate(peaks[0], start=1):
            if i % 2 == 0:
                teven.append((pi-previous)*oneFramemSec)
            else:
                todd.append((pi-previous)*oneFramemSec)
            previous = pi
        # Разницы медиан между "тиком" и "током"
        return np.abs(np.median(teven)-np.median(todd))
```

Рис. 15. Код алгоритма нахождения ошибки за один удар.

Затем, анализируя созданный ранее массив пиков в аудиофайле, находится фактическое количество полуколебаний часового механизма и, сравнивая это значение с эталонным для данного механизма, высчитывается отклонение часов за сутки в секундах (см. рис. 16). Это и является основным показателем точности часового механизма, которое нам необходимо.

```
def getMovementFrequencyFromPeakIndex(self, sampleRate, peaks):
        fs = []
        for i in range(len(peaks[0])-1):
            delta = (peaks[0][i+1]-peaks[0][i])
            fs.append(sampleRate/delta)
        return 60*60*np.mean(fs)

def getDrift(self, refFrequency, realFrequency):
    # Высчитываем отклонение фактической частоты от эталонной
    deltaBeat = realFrequency-refFrequency
    frac = deltaBeat/refFrequency
    return frac
```

Рис. 16. Алгоритмы нахождения частоты механизма и сравнения его с эталонной частотой.

Итоговый фрагмент кода, объединяющий в себе все вышеперечисленные функции и возвращающий значение отклонения хода за сутки и одни удар, представлен на рисунке 17.

```
def analysisGUI(self, filename):
        self.sampleRate, self.frames = self.importWAV(filename)
        oldFrames = self.frames
        self.frames = butter_highpass_filter(self.frames, 1000, self.sampleRate, 5)
        self.frames = self.applyWiener(self.frames)
        # Запись обработанного файла
        scipy.io.wavfile.write("high_data.wav", self.sampleRate, self.frames.astype(np.int16))
        p = self.peakSignalDetection(self.frames[:])
        # Получение ошибки за один удар, частоты полуколебаний и отклонения хода
        err = self.getBeatError(p)
        freqT = self.getMovementFrequencyFromPeakIndex(self.sampleRate, p)
        timeDev = self.getDrift(self.reffrequency, freqT)
        return [timeDev*3600, err]
```

Рис. 17. Фрагмент кода после объединения.

Таким образом, работник выбирает эталонную частоту полуколебаний [6] колеса баланса механизма, с которым он работает в данный момент. После нажатия на кнопку

«Анализировать» на сервере запускается вышеописанный процесс, который оценивает точность хода часов. Результат оценки представлен в виде показателей отклонения хода за один удар и за сутки (см. рис. 18).

Рис. 18. Результат анализа аудиофайла.

После того как была оценена точность за сутки, работник может оставить часы для дальнейшей проверки точности за неделю и месяц, либо интерполировать результат предыдущих оценок на заданный момент времени.

Завершив все проверки, работник закрывает обращение, оно автоматически добавляется в историю ремонтов данного механизма и записывается в базу данных системы.

На главной странице часового сервиса помимо списка предоставляемых услуг приведен список часовых механизмов, когда-либо обслуженных сервисом (см. рис. 19).

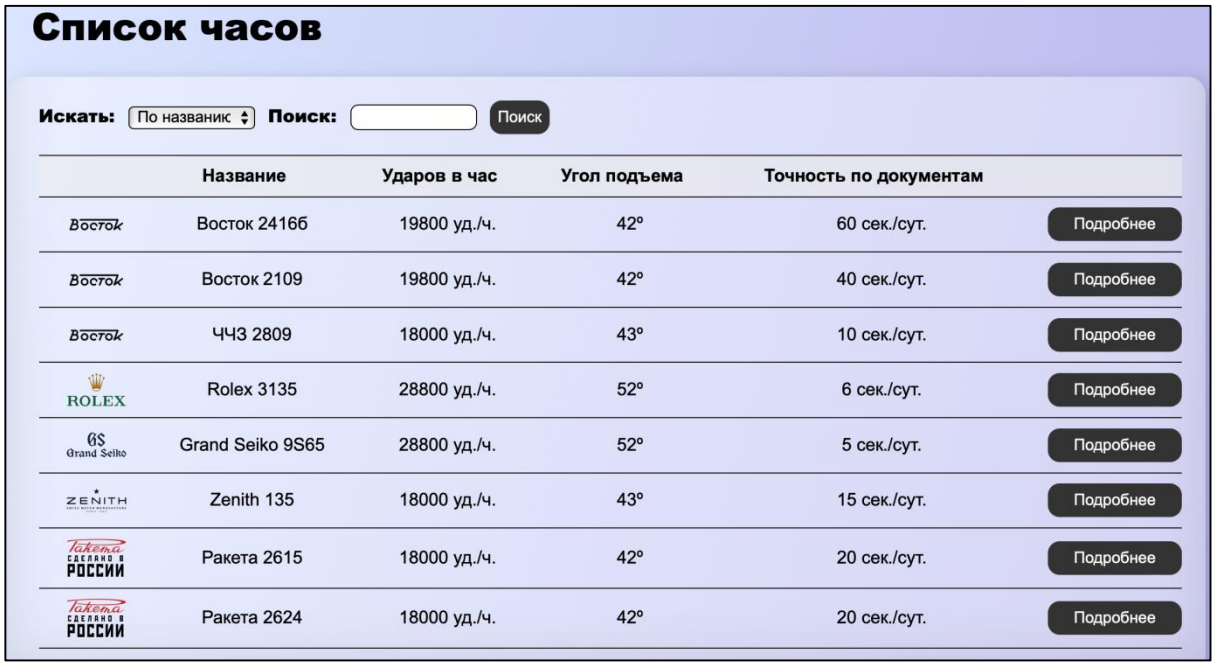


Рис. 19. Страница со списком всех механизмов, обслуживаемых сервисом.

При выборе кнопки «Подробнее» можно увидеть заявленные в паспорте характеристики выбранного механизма, историю обращений по нему в данный часовой сервис, а также точность хода после обслуживания (см. рис. 20).

Рис. 20. Просмотр истории обслуживания механизма.

Также можно просмотреть среднюю точность хода данного механизма, отличие реальной точности от заявленной точности в паспорте механизма, а также соответствующий класс точности. Далее находится статистическая информация о частоте обращений в сервис по данному механизму, процент обращений, по которым был необходим ремонт, а также средняя цена за обслуживание (см. рис. 21).

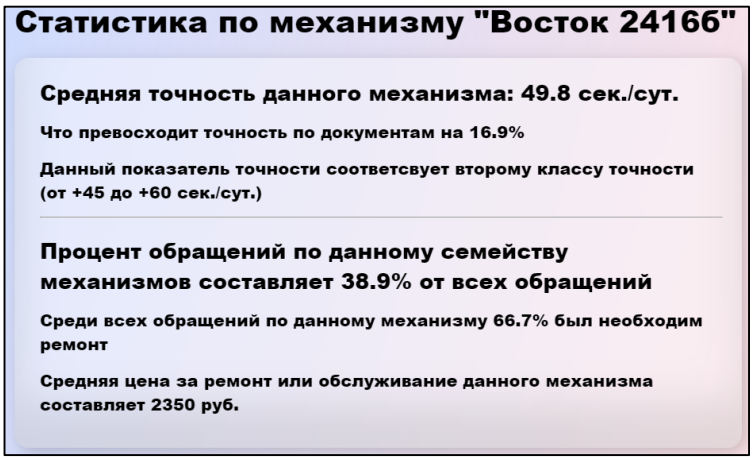


Рис. 21. Просмотр статистики по точности и надежности механизма.

Таким образом, в статье приводится описание разработанной авторами программно-информационной системы, предназначенной для оценки точности и надежности часовых

механизмов. Для оценки точности механических часов использовался метод звукового анализа аудиофайла, полученного в результате записи звука хода часов через микрофон. Следует отметить, что существуют и другие методы и алгоритмы для определения точности хода (см., например, [6], [7]). Для оценки надежности приводится статистика пользовательских обращений и результатов их обслуживания в часовом сервисе.

СПИСОК ЛИТЕРАТУРЫ

1. Дробница И. К. Россия на мировом рынке часов // Международная экономика. – 2019. – № 11. – С. 69–83.
2. Каплун Д. Е. Сегментация рынка швейцарских часов в России // Modern Science. – 2019. – № 3. – С. 141–144.
3. Elibrary.ru – Патент: способ измерения точности хода механических часов [Электронный ресурс]. – Режим доступа: https://www.elibrary.ru/download/elibrary_37826652_15421426.pdf (дата обращения: 14.07.2023).
4. Пирогов А. А., Акишкин Р. М., Гончаренко И. В., Сёмка Э. В., Турецкая Е. В. Реализация цифрового фильтра Баттерворта с использованием библиотек языка программирования Python // Радиотехника. – 2023. – Т. 87, № 8. – С. 54–58.
5. Яшин В. Н. Адаптивные алгоритмы в информационно-измерительных системах // Современные наукоемкие технологии. – 2023. – № 8. – С. 96–102.
6. Самсонов Г. П., Амосов А. Г., Чуракова Е. Ю., Михайлова Е. В. Геометрическая зависимость определения точности механизмов // Инновации и инвестиции. – 2021. – № 3. – С. 153–157.

АКИМОВА Д. А.

РАЗРАБОТКА ЭЛЕКТРОННОГО ПЛАНИРОВЩИКА ПРОЦЕССА ОБУЧЕНИЯ

Аннотация. Статья посвящена описанию разработки программного обеспечения, предлагающего пользователю помощь в проектировании образовательной программы. Визуализируя вводимые данные в двух вариантах – ориентированный граф и списки по семестрам, приложение облегчает задачу планировщику.

Ключевые слова: образовательная программа, планирование последовательности обучения, учебный план, визуальное представление информации, Microsoft Visual Studio 2022, Windows Presentation Foundation, .NET Core.

AKIMOVA D. A.

DEVELOPMENT OF ELECTRONIC PLANNER FOR DESIGING THE CURRICULUM

Abstract. The article presents a description of the development of the software to assist in designing the curriculum. The application facilitates the work of the planner by visualizing the data in two variants, oriented graph and lists by semester.

Keywords: training program, planning a sequence of the education, curriculum, visual presentation of information, Microsoft Visual Studio 2022, Windows Presentation Foundation, .NET Core.

Качество подготовки специалиста во многом определяется программой его обучения и учебным планом вуза. Задача последнего состоит в том, чтобы верно распределить нагрузку на студента на протяжении всего обучения, при этом сохраняя корректную последовательность изучаемых предметов и связей между ними. Здесь на помощь приходит программа, помогающая визуализировать ее.

Приложения из подобной направленности, выполняющие в той или иной степени вышеуказанное требование, уже существуют. Например, Шахтинская программа [1], обладающая большой базой данных, которую заполняет пользователь на основе уже заранее разработанного образовательного плана. Программное обеспечение (ПО) имеет обширный функционал, однако не является хорошим помощником в создании программы обучения учащихся. Описанное ниже приложение никак не конфликтует с Шахтинской программой, поскольку и выполняют они разные задачи. Также стоит упомянуть программу Obsidian [2], представляющую собой хранилище данных, предлагающую широкий набор действий по манипулированию данными. Это ПО и подобные ей решают общие задачи, но не узко направленные.

Суть программы состоит в том, чтобы провести человека через весь путь проектирования учебного плана, начиная с его «чернового» варианта, где известны лишь взаимосвязи нескольких дисциплин, а представление о полной очередности всех позиций еще не сформировано (т.е. визуализация происходит в виде ориентированного графа), и заканчивая уже готовым табличным видом, т.е. списком дисциплин, упорядоченным по семестру и алфавиту. При том, программа была направлена на пользователей ОС Windows, поскольку это самая распространенная и популярная операционная система на портативный компьютер на момент создания приложения.

Касательно функционала, программа должна была упрощать построение учебного плана, также обеспечивая систему проверок во избежание ошибок системы. Разрабатываемое ПО должно было позволять, как добавление дисциплины, так и ее удаление, а также создание связей между уже зарегистрированными позициями. К тому же существовала необходимость в редактировании уже созданной дисциплины и возможность меняться между режимом ориентированного графа и табличным видом, где столбцами выступают семестры. Программа должна была быть интуитивно понятна пользователю, легко передаваема через различные сервисы (Google Drive, Yandex Disk) или запоминающие устройства (USB-флеш-накопитель) и реализована в средах, позволяющих запуск программы на ОС Windows актуальных версий.

Любая программа имеет различные условия своего создания и представление того, как она должна выглядеть. Вышеописанные требования были сформулированы и написаны с помощью одного из международных стандартов ISO/IEC TR 19759:2005 [3].

Для реализации программы с учетом всех требований был выбран язык программирования C#. Так как этот язык изначально был предназначен для разработчиков на Windows, то платформа .NET тесно связана с этой операционной системой. Работа осуществлялась в интегрированной среде разработки (IDE) Microsoft Visual Studio 2022 с применением графической подсистемы Windows Presentation Foundation (WPF) в составе .NET Framework, использующей язык XAML. Эти инструменты для реализации программы также выбирались исходя из критерия относительно простой установки без скачивания сторонних программ. Все, что возможно потребуется для поддержки ПО, система предлагает скачать сама с прямыми ссылками.

С самого начала стоял вопрос о способе хранения данных: готовая база данных или же прописанная лично структура. В конечном счете, сделан был выбор в сторону второго во избежание дополнительных сложностей с работой базы данных, а также возможных последующих запретов со стороны разработчиков на ее эксплуатацию в связи с территориальным происхождением программы. Также, собственная структура данных предполагает гибкость по отношению к нуждам разработчика и пользователя, и единственный

ее недостаток — это отдельная реализация, поскольку не является готовым инструментом, а требует затраты временных ресурсов на свою реализацию. Было создано несколько классов, а также глобальных листов с типом этих классов для удобства, чтобы обращаться к ним напрямую, а не передавать из метода в метод.

Поскольку большая часть графических объектов создавалась динамически, то их создание было прописано через код на языке программирования C#, тогда как в целом интерфейсные области и кнопки (объекты, которые присутствуют статически) через XAML.

Для хранения информации о дисциплинах и взаимосвязях между ними использовался специальный класс ObservableCollection<T>, по функциональности похожий на список List за тем исключением, что позволяет известить внешние объекты о том, что коллекция была изменена. Хранилище данных являлось глобальной переменной для упрощения взаимодействия с ними методов. В качестве типа информации, хранившейся в коллекции, использовался написанный вручную класс Discipline, включающий в себя свойства названия дисциплины, семестра, комментария, а также наличия списков List<Discipline> как предшествующих ей дисциплин, так и последующих.

При запуске программы появляется статичный интерфейс (рис. 1), где изменения можно будет наблюдать лишь в области 1 и 3. Область 2 остается неизменной.

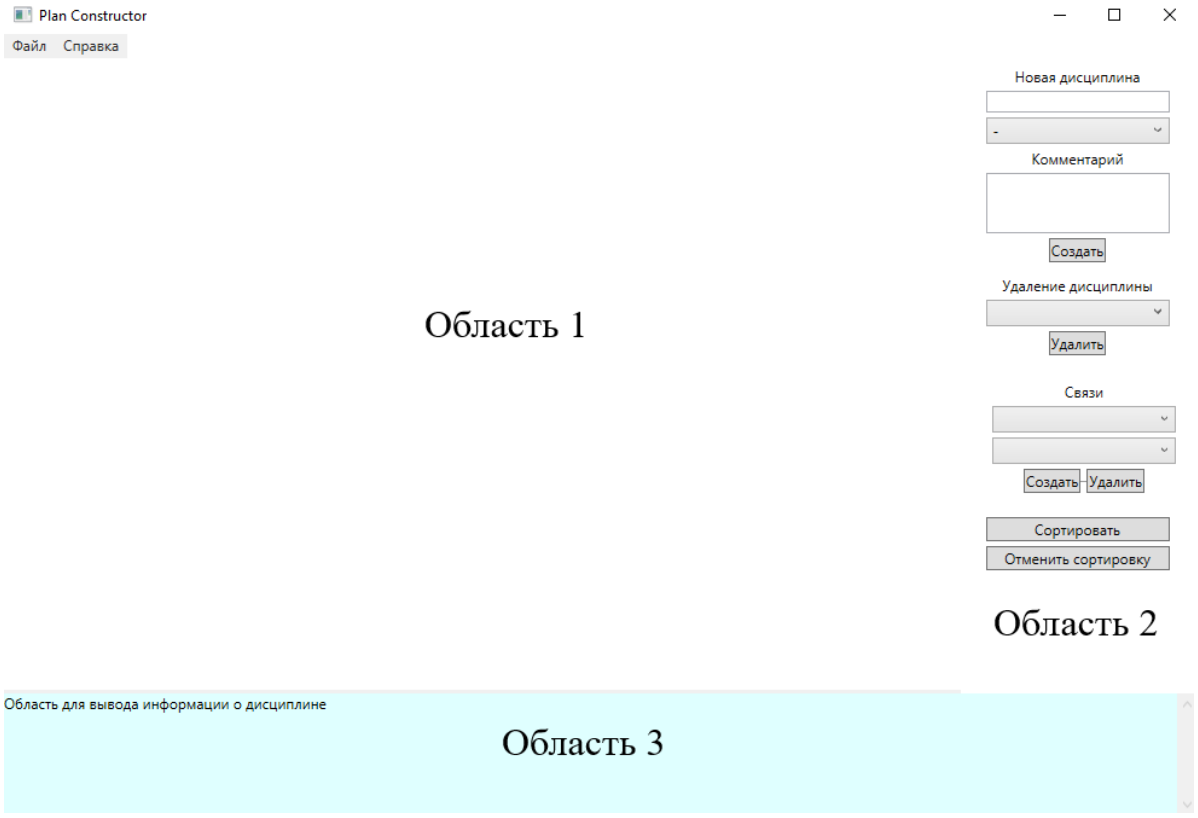


Рис 1. Статичный интерфейс.

Добавление дисциплин происходит через форму в области 2, где вводится ее название, выбирается один из 8 семестров, а также пишется комментарий, после чего пользователь нажимает на кнопку «Создать», либо клавишу Enter. Обязательно лишь поле названия, если программа находится в режиме орграфа. В режиме сортировки она затребует указание семестра (по умолчанию он будет записываться как нулевой). Добавленные дисциплины мгновенно отображаются в области 1 (рис. 2).

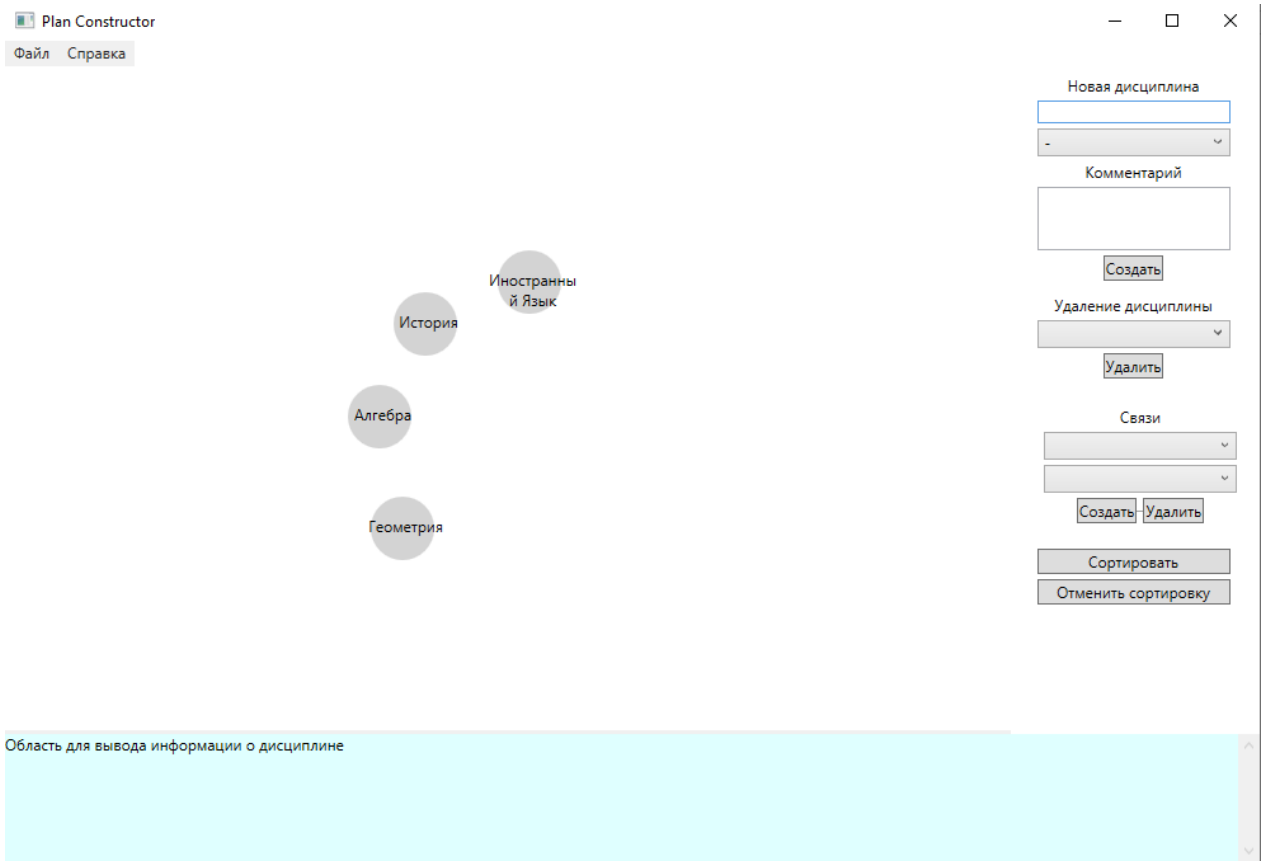


Рис. 2. Добавление дисциплины.

Между дисциплинами могут настраиваться связи. Проектировщику плана может быть точно известно, что одна дисциплина должна следовать за другой или предшествовать третьей. Создание отношения происходит в форме в Области 2 (рис. 3). Из списков зарегистрированных дисциплин выбираются необходимые и создается отношение. Таким же образом связь может и удалиться. При этом, если у предшествующей дисциплины указан семестр больший, чем у последующей, то программа выведет сообщение об ошибке.

Область 1 является интерактивной. При нажатии левой кнопкой мыши на дисциплину, она выделяется (меняет цвет) (рис. 4), а в Области 3 о ней выводится информация. При нажатии правой кнопкой мыши на дисциплину, появляется контекстное меню, предлагающее удаление или же редактирование. Удаление может происходить как через контекстное меню, так и через форму в Области 2.

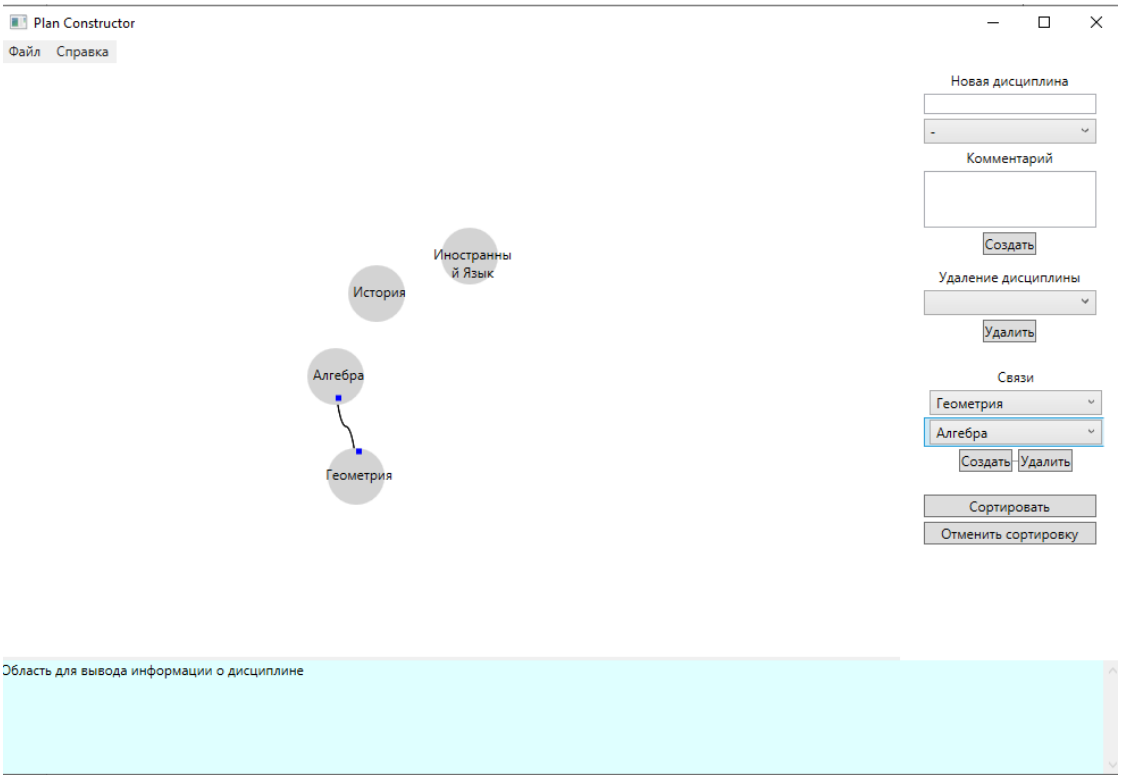


Рис. 3. Добавление связи.

При выборе опции «Редактировать» выводится отдельное окно (рис. 4), обращающееся к глобальным переменным, хранящим запрос. Изменения данных сохраняются и передаются в хранилище, и в основном окне происходит перерисовка графа.

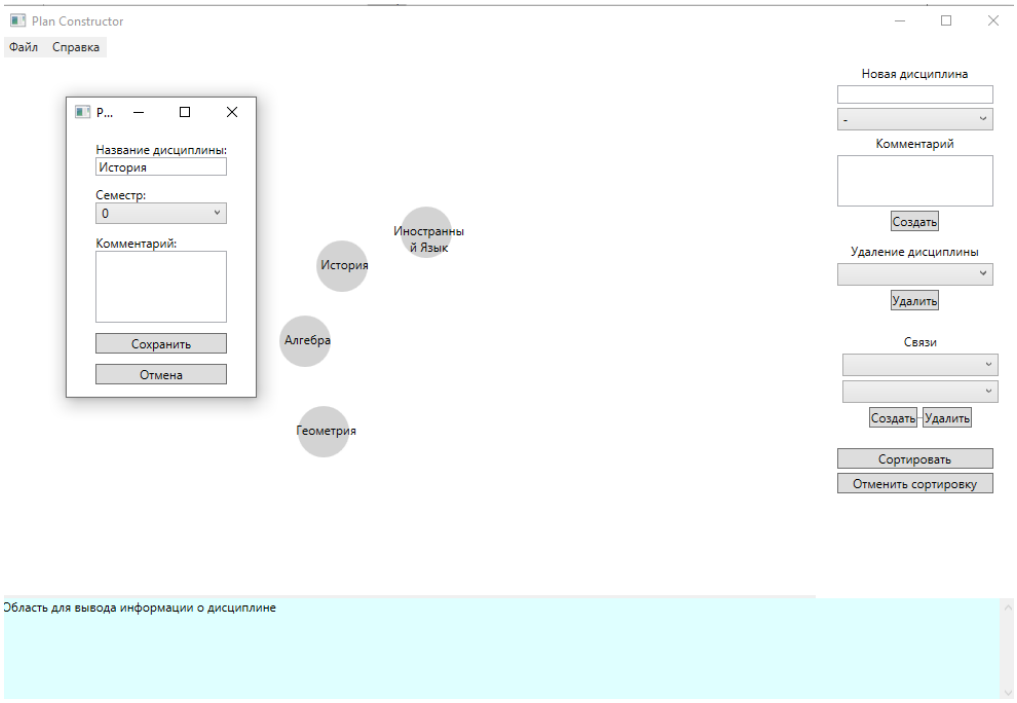


Рис. 4. Выведение окна редактирования.

Программа содержит в себе два режима: ориентированный граф и упорядоченные по семестрам и алфавиту списки (рис. 5, рис. 6). Их различие, помимо визуальной составляющей, заключается в том, что орграф не требует немедленного указания семестра и предполагает редактирование свойств дисциплины, тогда как второй режим подразумевает проверку имеющихся данных на то, чтобы все дисциплины (как уже имеющиеся, так и добавляющиеся) шли с указанием семестра. При том важным аспектом являются взаимосвязи, так как уже на этапе создания отношения между двумя дисциплинами, идет проверка адекватности подобной связи – не поставлена ли предшествующая дисциплина на семестр впереди последующей. Переключение между режимами идет по двум кнопкам – «Сортировать» и «Без сортировки». Их нажатие вызывает методы, которые меняют состояние глобальной переменной, отслеживающей то, какой режим на данный момент активирован в программе.

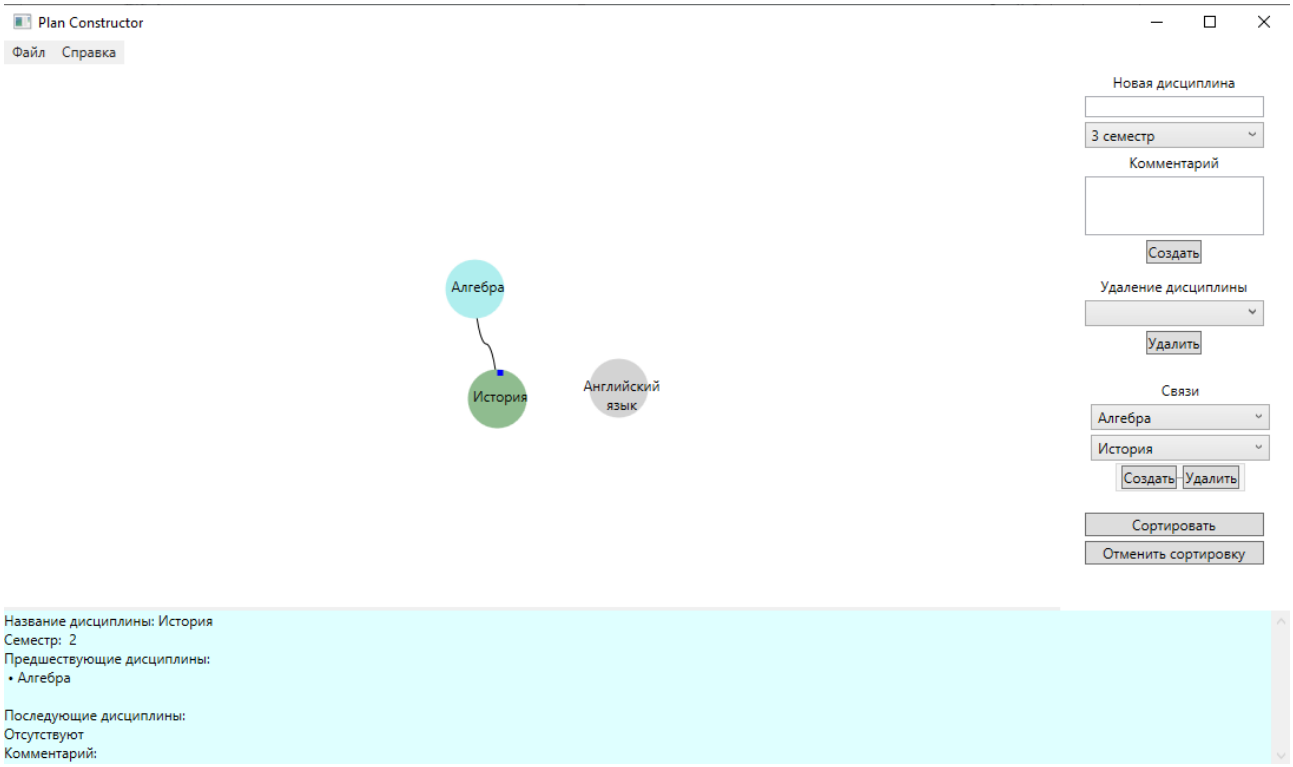


Рис. 5. Режим ориентированного графа.

В зависимости от режима, срабатывают разные методы добавления необходимых компонентов в Canvas (контейнер в WPF). В случае орграфа данные об элементах хранятся в соответствующих глобальных списках, т.к. расположение элементов изначально задается случайным образом, и, если каждый раз при добавлении дисциплины будет идти перерисовка, это негативным образом скажется на эффективности визуализации – если элементы будут постоянно менять свое месторасположение, это может привести к путанице. Поэтому данные о всех графических элементах ориентированного графа хранятся в списках (List<T>), и при

прорисовке новые элементы добавляются в Canvas без необходимости предварительного очищения поля (если только речь не идет о переключении между режимами).

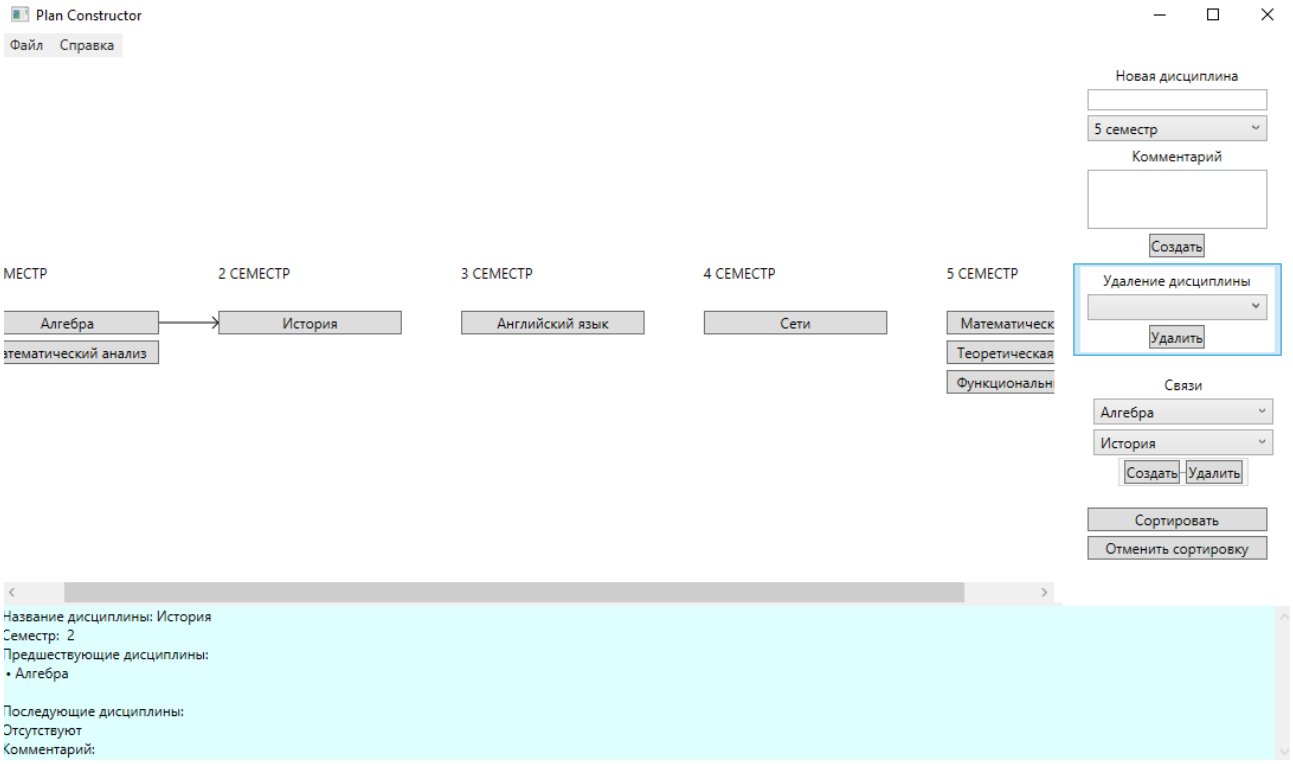


Рис. 6. Режим расписания по семестрам.

В случае режима списков по семестрам при добавлении новых дисциплин без переключения на режим орграфа, удаления или же переключения между двумя состояниями поля каждый раз идет перерисовка. Это осуществляется с той целью, чтобы список всегда был упорядоченным по алфавиту, и между элементами не было ненужных «прогалов», если речь идет об удалении дисциплины. Также расположение визуальных составляющих происходит не случайным образом, а потому нет необходимости в запоминании чьих-либо позиций – достаточно хранить данные о самих дисциплинах (самыми важными свойствами в данном случае будут название и семестр).

Для вызова окна редактирования в режиме орграфа используется новое окно со своими методами. Необходимые данные о дисциплине, которая будет редактироваться, записываются в глобальных переменных, чтобы передавать информацию между окнами (главным окном и окном редактирования). Если же какие-то изменения будут внесены в окне редактирования, то глобальные переменные будут перезаписаны, и пользователь вернется на главное окно, в котором метод перезапишет информацию и внесет необходимые изменения, как в структуры данных, так и в визуальную составляющую.

В конце разработки с соблюдением требований было получено заявленное программное обеспечение для составления учебного плана, позволяющее создать хранилище данных о существующих дисциплинах и их параметрах и прошедшее тестирование.

СПИСОК ЛИТЕРАТУРЫ

1. ММИС Лаборатория: Программы: GosInsp: сайт / Лаборатория ММИС [Электронный ресурс]. – Режим доступа: https://www.mmis.ru/programs/GosInsp (дата обращения: 30.05.2023).
2. Obsidian: сайт / Obsidian, 2023 [Электронный ресурс]. – Режим доступа: https://obsidian.md/ (дата обращения: 30.05.2023).
3. Internet Archive WayBackMachine: Основы Программной Инженерии (по SWEBOK): сайт / "Основы программной инженерии" Copyright, Сергей Орлик, 2004-2010 [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20100604013037/ http://swebok.sorlik.ru/1_software_requirements.html (дата обращения: 27.10.2023).

КАРЧИГАНОВ А. Ф.

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ OPENCL ДЛЯ ВЫЧИСЛЕНИЙ НА СТРУКТУРИРОВАННЫХ СЕТКАХ С ИСПОЛЬЗОВАНИЕМ GPU

Аннотация. В статье рассматривается возможность применения технологии OpenCL для параллельного вычисления численного решения на примере двумерной задачи теплопроводности. Рассмотрены особенности программирования в парадигме параллелизма задач и данных. Показаны стандартные сложности и встроенные в OpenCL способы их решения при построении программы для вычислений на многомерных сетках. Для CPU реализована аналогичная однопоточная программа на языке C++20, сравнена производительность.

Ключевые слова: OpenCL, гетерогенные вычисления, ОКМД, уравнение теплопроводности.

KARCHIGANOV A. F.

USING OPENCL TECHNOLOGY FOR STRUCTURED GRID COMPUTING USING GPU

Abstract. In this article the possibility of using OpenCL technology for parallel calculation of a numerical solution using the example of a two-dimensional heat conduction problem is discussed. The features of programming in the paradigm of task and data parallelism are considered. The standard difficulties and methods built into OpenCL for solving them when building a program for calculations on multidimensional grids are shown. For the CPU, a similar single-threaded program was implemented in C++20, and the performance was compared.

Keywords: OpenCL, heterogeneous computing, SIMD, heat equation.

Появление многоядерных процессоров и массовый их выход в потребление, сделал программирование под суперкомпьютеры доступным для широкого круга. Одно из основных применений суперкомпьютера – численное моделирование, сопряженное с очень большим объёмом сложных вычислений.

Другая веха в развитии численного моделирования – появление возможности использования графических процессоров для неграфических вычислений. Графические процессоры, с одной стороны, имея меньший набор команд по сравнению с процессорами общего назначения, с другой – имеют намного большее число ядер, что, с определенного этапа их развития, сделало целесообразным их использование для математических вычислений.

Одна из первых и наиболее широко используемых технологий остается CUDA [1]. Имея достаточно простой интерфейс взаимодействия под языки C/C++, а также реализованные

обертки для других языков программирования, CUDA имеет и свой основной недостаток – возможность запуска параллельных программ только под видеокарты компании Nvidia.

Появление стандарта для открытого языка вычислений OpenCL сделало доступным возможность написания общих многопоточных программ не только под видеокарты, но и под другие аппаратные ускорители, в том числе и процессоры общего назначения. Имея похожий на CUDA интерфейс взаимодействия через C/C++ и другие языки программирования, для самих математических вычислений OpenCL использует подход близкий к языку шейдеров OpenGL SL, который так же использовал программы на собственном языке, близком к C; а выходные буферы могли быть использованы не только при выводе изображения на экран пользователя, но и для прямого копирования данных с них в общую оперативную память компьютера, чем также пользовались [2].

Целью данной работы было исследование возможности использования технологии OpenCL для решения задач на структурированных сетках, выявление типовых сложностей при разработке программ на принципах параллелизма данных.

В качестве примера выбрана двумерная задача теплопроводности для однородного тела с начальными и краевыми условиями [3]: однородная пластина с коэффициентом теплопроводности λ, шириной W и высотой H, имеет начальную температуру в момент времени $t = 0$, заданную функцией $T_0(x, y)$, где $x \in [0; W]$, $y \in [0; H]$. Грани пластины также могут быть подвержены внешнему воздействию, т.е. на них могут быть заданы функции $T_u(t, x),\ T_d(t, x),\ T_l(t, y),\ T_r(t, y)$ для соответственно верхней, нижней, левой, правой границ. Для подобных задач можно построить явную конечно-разностную схему [3].

Разбив область $x \times y$ на $N_x \times N_y$ ячеек с соответствующими шагами сетки $h_x \times h_y$, установив шаг по времени τ, выразим значение в ячейке i, j для каждого следующего момента времени:

$$T_{i,j}^{+} = T_{i,j} + \tau\lambda\left(\frac{\left(T_{i-1,j} + T_{i+1,j} - 2T_{i,j}\right)}{h_x^2} + \frac{\left(T_{i,j-1} + T_{i,j+1} - 2T_{i,j}\right)}{h_y^2}\right), \quad (1)$$

где $i = 1, \dots, N_x,\ j = 1, \dots, N_y$. Для удобства, будем использовать дополнительные слои при $i \in [0, N_x + 1],\ j \in \left[0, N_y + 1\right]$ для задания значений на левых, правых, верхних и нижних границах соответственно. Тогда полученная схема с ячейками $[j, i]$ будет выглядеть:

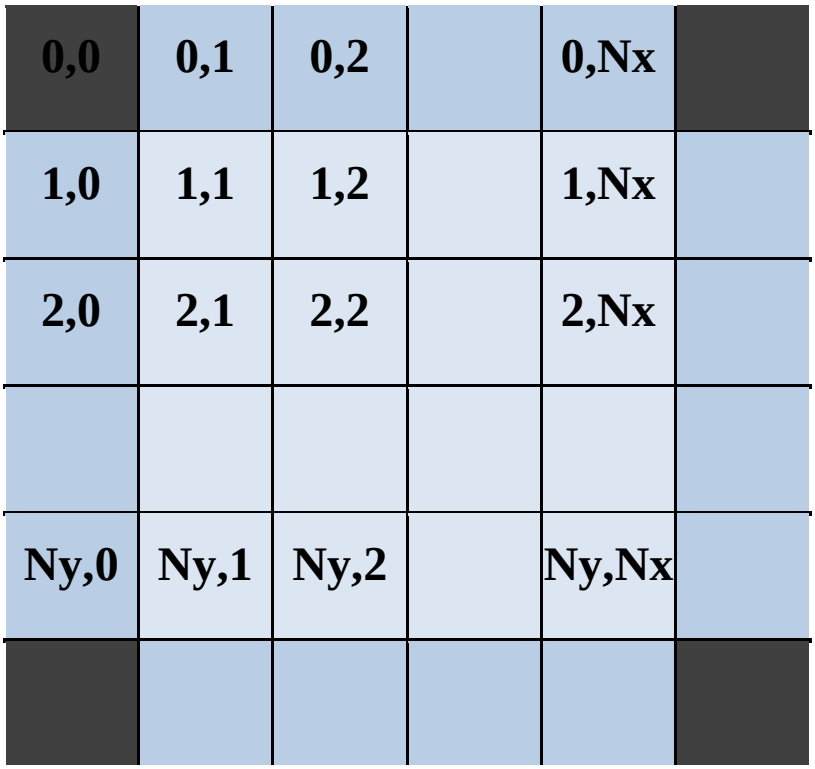


Рис. 1. Разбиение области на ячейки.

где светло-голубым цветом выделены ячейки, рассчитываемые по формуле (1), темно-голубым – через граничные условия, черные – полностью неиспользуемые в программном алгоритме.

Была реализована host-программа, которая связывается с устройствами, поддерживающими драйвер OpenCL, инициализирует начальные данные, собирает kernel-программу, заносит и вытаскивает данные из буферов графической памяти. При составлении host- и kernel-программ необходимо учитывать некоторые особенности работы графического процессора.

Так как в графическом процессоре для достижения больше производительности используются обычно числа с плавающей точкой одинарной точности (32-битные), то и при создании данных на host-программе необходимо использовать их, а не 64-битные. В случае с C++20 – это `float_t`.

При работе с памятью графического ускорителя, невозможно случайным образом занимать и освобождать память в глобальной области, то есть все указатели на массивы данных должны быть известны заранее, а значит невозможно загрузить внутрь графической памяти двумерный массив как указатель на указатели. Соответственно, область из рисунка 1 необходимо представить в виде одномерного массива. Заполним структуру `std::<float_t>`, редуцировав двумерный массив до одномерного:

```
auto valuesArray = std::vector<float_t>(fullSize);
for (size_t j = 0; j < height.Count; ++j)
{
	auto heightOffset = j * width.Count;
	for (size_t i = 0; i < width.Count; ++i)
	{
		valuesArray[heightOffset + i] = initialFunciton(
			width.Coordinates[i],
			height.Coordinates[j]
		);
	}
}
```

Листинг 1. Инициализация начальных данных.

где `initialFunciton` – функция $T_0(x, y)$.

Полученный массив из-за граничных ячеек имеет $\left((N_x + 2) \cdot \left(N_y + 2\right)\right)$ элементов (до Листинга 1 вычислено в переменной `fullSize`), в то время как программа для вычисления по формуле (1) должна быть применена только на $\left(N_x \cdot N_y\right)$ элементов.

Программы на OpenCL пишутся в парадигме параллелизма, то есть необходимо описывать не циклы (как в Листинге 1), а функции, которые будут отрабатывать на каждой рабочей единице. Например, программа для сложения массивов `A` и `B` в массив `C` размерностью 32 будет выглядеть:

```
__kernel void add(__global float *A, __global float *B, __global float *C)
{
	size_t i = get_global_id(0);
	C[i] = A[i] + B[i];
}
```

Листинг 2. Простейший алгоритм сложения массивов.

где `get_global_id(0)` – получение глобального номера рабочей единицы в первом измерении. В случае, когда вычисления на граничных ячейках нужно пропустить, нельзя использовать следующий код:

```
__kernel void add(__global float *A, __global float *B, __global float *C)
{
	size_t i = get_global_id(0);
	if ((i > 0) && (i <= 32)
	{
		C[i] = A[i] + B[i];
	}
	else
	{ … }
}
```

Листинг 3. Пример дивергенции кода.

так как произойдет дивергенция кода – ситуация, когда в целях синхронизации все рабочие единицы затратят время для отработки или ожидания сначала блока `if`, а потом и блока `else`.

Необходимо сделать отступы для соответствующих измерений. Подобная задача для графических ускорителей встречалась и ранее: например, задача Гауссова размытия [4]. OpenCL позволяет задавать размерности редуцированного массива, а также отступы от начала для каждой размерности [5]. Таким образом, для обоих размерностей необходимо сделать отступы размером в 2, а расчетная область будет выглядеть следующим образом:

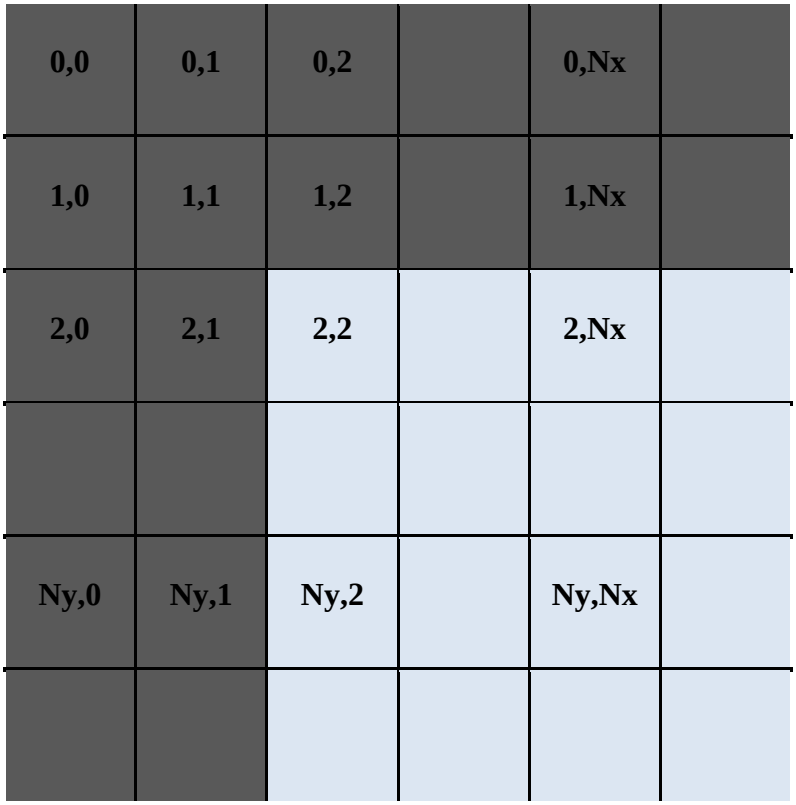


Рис. 2. Рабочее пространство для kernel-программы.

где светло-голубым цветом обозначены ячейки, относительно которых будут распределяться индексы рабочих единиц. Тогда сами индексы $[i, j]$ для соответствия формуле (1) можно будет определить по соответствующим измерениям:

```
int i = get_global_id(0) - 1;
int j = get_global_id(1) - 1;
```

Листинг 4. Определение индексов вычисляемой ячейки.

В host-программе создадим текст kernel-программы, реализующей вычисление по формуле (1):

```
	auto programTextTemplate = R"(
__kernel void conduction(
	float coefficientX, float coefficientY,
	__global float *input, __global float *output
)
{{
	int index0 = get_global_id(0) - 1;
	int index1 = get_global_id(1) - 1;
	int index = (index1 * {0}) + index0;
	output[index] = input[index]
		+ (coefficientX * (
			input[index - 1] + input[index + 1] - (2 * input[index])
		))
		+ (coefficientY * (
			input[index - {0}] + input[index + {0}] - (2 * input[index])
		));
}}
)";
	auto programTextTemplateArgs = std::make_format_args(width.Count);

	auto programText = std::vformat(programTextTemplate, programTextTemplateArgs);
```

Листинг 5. Составление текста kernel-программы.

где в случае, когда в переменной `width.Count` будет находиться значение $N_x = 1024$, в переменной `programText` будет составлен следующий текст kernel-программы:

```
__kernel void conduction(
	float coefficientX, float coefficientY,
	__global float *input, __global float *output
)
{
	int index0 = get_global_id(0) - 1;
	int index1 = get_global_id(1) - 1;
	int index = (index1 * 1024) + index0;
	output[index] = input[index]
		+ (coefficientX * (
			input[index - 1] + input[index + 1] - (2 * input[index])
		))
		+ (coefficientY * (
			input[index - 1024] + input[index + 1024] - (2 * input[index])
		));
}
```

Листинг 6. Пример kernel-программы при $N_x = 1024$.

где `coefficient` и `coefficientY` вычислены заранее как $\frac{\tau\lambda}{h_x^2}$ и $\frac{\tau\lambda}{h_y^2}$ из формулы (1) соответственно.

Так как задачи в OpenCL выполняются асинхронно, то в них присутствуют очереди. Все задачи в одной очереди выполняются друг за другом. Различные очереди выполняются

параллельно. При создании задачи, host-программа автоматически не ждет ее выполнения, но имеет есть возможность ожидания задачи для синхронизации. Составим очередь задач, где данные будут загружаться в память GPU, выполняться по программе из Листинга 6, и выгружаться обратно в общую оперативную память:

```
auto previousEvents = std::vector<cl::Event>(0);
auto writeEvent = cl::Event();
errorCode = commandQueue.enqueueWriteBuffer(
      inputBuffer, CL_TRUE, 0, fullSize * sizeof(float_t), valuesArray.data(),
      &previousEvents, &writeEvent
);
previousEvents.push_back(writeEvent);
auto kernelEvent = cl::Event();
errorCode = commandQueue.enqueueNDRangeKernel(
      kernel, ndOffset, ndGlobal, cl::NullRange,
      &previousEvents, &kernelEvent
);
previousEvents.push_back(kernelEvent);
auto readEvent = cl::Event();
errorCode = commandQueue.enqueueReadBuffer(
      outputBuffer, CL_TRUE, 0, fullSize * sizeof(float_t), valuesArray.data(),
      &previousEvents, &readEvent
);
readEvent.wait();

errorCode = commandQueue.finish();
```

Листинг 7. Очередь задач.

Данный алгоритм выполняется на каждом шаге по времени. Между его выполнениями, на host-программе вычисляются значения для граничных ячеек по функциям T_u, T_d, T_l, T_r.

В соответствии с (1) была также составлена аналогичная однопоточная программа на C++20. Программы на OpenCL и на C++20 были сравнены по производительности: на сетке размерностью 1024×1024 с числом шагов $10'000$ (максимальное время $T = 1.0$, шаг по времени $\tau = 0.0001$) программа на GPU отработала за 15 секунд, на CPU – 610 секунд. Программу на OpenCL можно еще ускорить, реализовав программы для функций T_u, T_d, T_l, T_r, тем самым исключив время для загрузки и выгрузки данных с графической памяти на каждом шаге. Полученные значения массивов совпали побитово.

Вычисления, отдаваемые устройствам под управлением OpenCL, выполняются на всех вычислительных единицах этого устройства. OpenCL имеет поддержку деления устройства только для некоторых процессоров, поэтому проверить эффективность и ускорение алгоритма на ограниченном количестве ядер графического процессора невозможно.

Программы были также проверены при $T = 2.0$, $\tau = 0.0001$ на разном количестве ячеек. Вычислена производительность и эффективность относительно вычисления с предыдущим количеством ячеек (таблица 1).

Таблица 1

Производительность и эффективность работы программы

Время, мс	$N_x \times N_y$	$N_x N_y$	**Производительность, ячеек/мс**	**Эффективность**
5452	256×256	65536	12.02054	
10005	256×512	131072	13.10065	1.089855
18224	512×512	262144	14.38455	1.098003
32202	512×1024	524288	16.28122	1.131855
60771	1024×1024	1048576	17.25455	1.059782

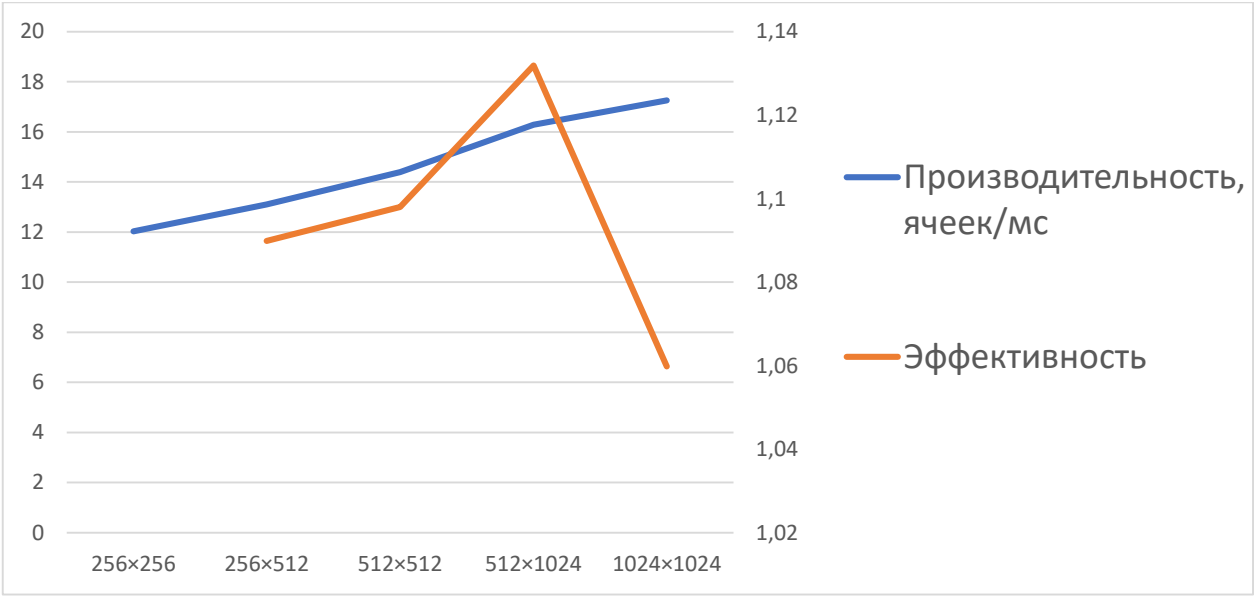


Рис. 3. Производительность и эффективность работы программы.

В ходе выполнения данной работы были изучены особенности написания и работы программ с использованием OpenCL. Была реализована host-программа на языке C++20 для составления и запуска kernel-программы на OpenCL. Производительность и эффективность работы программы на OpenCL были замерены, а также была сравнена производительность с аналогичной однопоточной программой под процессор общего назначения. На основании полученных результатов можно сделать вывод, что программы на OpenCL имеют существенно высокую производительность, а также хорошую эффективность на больших объемах данных.

СПИСОК ЛИТЕРАТУРЫ

1. Антонюк В. А. Программирование на видеокартах (GPGPU). Спецкурс кафедры ММИ. – М.: Физический факультет МГУ им. М.В. Ломоносова, 2015. – 48 с.
2. Вычисления на GPU с помощью OpenGL [Электронный ресурс]. – Режим доступа: https://velikodniy.github.io/2017/08/14/gpgpu-opengles (дата обращения: 22.10.2023).
3. Кузнецов Г. В., Шеремет М. А. Разностные методы решения задач теплопроводности: учебное пособие. – Томск: Изд-во ТПУ, 2007. – 172 с.
4. Compute shaders in graphics: Gaussian blur [Электронный ресурс]. – Режим доступа: https://lisyarus.github.io/blog/graphics/2022/04/21/compute-blur.html (дата обращения: 22.10.2023).
5. Erik S. Gaussian Blur using OpenCL and the built-in Images/Textures [Электронный ресурс]. – Режим доступа: https://www.eriksmistad.no/gaussian-blur-using-opencl-and-the-built-in-images-textures (дата обращения: 22.10.2023).