МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет
«Московский институт электронной техники»

Факультет микроприборов и технической кибернетики
Кафедра информатики и программного обеспечения вычислительных систем

Николаев Олег Владимирович

Бакалаврская работа
по направлению 09.03.04 «Программная инженерия»

Разработка программного модуля по реализации функции интеллектуальной обработки данных для системы 1С-Битрикс

Студент _____ Николаев О.В.
Научный руководитель,
к. т. н, доцент _____ Слюсарь В.В.

Москва 2016

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД – база данных

БУС – система «1С-Битрикс: Управление сайтом»

ИОД – интеллектуальная обработка данных

ПМ – программный модуль

ОС – операционная система

СУС – система управления содержимым

СУБД – система управления базами данных

CSS – каскадные таблицы стилей

HTML – язык гипертекстовой разметки

IDE – среда разработки

AJAX – технология асинхронного JavaScript и XML

ВВЕДЕНИЕ

В настоящее время технологии интеллектуальной обработки данных (ИОД) приобретают все большее распространение: они позволяют извлечь из необработанных данных ранее неизвестные, нетривиальные, практически полезные и доступные интерпретации знания и закономерности. Сфера применения ИОД широка: от биологии и медицины до маркетинга и веб-анализа. В системах, в которых каждый день обрабатывается большое количество данных, выгодно внедрять методы ИОД.

Для интернет-приложений наибольший интерес вызывают две задачи ИОД: кластеризация (выделение групп данных, или кластеров) и ассоциация (поиск закономерностей между связанными событиями). В область применения кластеризации и ассоциации входят задачи сегментации данных, анализа веб-логов, выявления похожих товаров и покупателей, рекомендации товаров, выделения групп пользователей и анализа их поведения. Поэтому актуальным является внедрение методов ИОД для решения этих задач в интернет-приложениях и системах управления веб-сайтами.

Для управления содержимым веб-сайтов широко используются системы управления содержимым (СУС). Система 1С-Битрикс является СУС, предназначенной для создания и поддержки корпоративных сайтов, интернет-магазинов, информационных порталов и других веб-проектов.

Целью данной работы является создание инструмента для применения методов ИОД для анализа, кластеризации и поиска ассоциативных правил в данных для системы 1С-Битрикс. Такой инструмент позволит эффективнее использовать информацию, находящуюся в базах данных системы 1С-Битрикс, и извлечь из неё дополнительную выгоду.

Практическая значимость данной разработки заключается в создании решения, которое позволит улучшить применение технологий ИОД в системе управления содержимым.

Пояснительная записка состоит из введения, исследовательского, конструкторского и технологического разделов, заключения, списка литературы и приложений.

В исследовательском разделе исследуются предметная область и актуальность выбранной темы, проводится обзор существующих программных решений, поставлены

цель и задача разработки и составлены требования к разрабатываемому программному модулю.

В конструкторском разделе выбраны инструменты разработки, разработаны структуры входных и выходных данных, разработана архитектуры и алгоритм работы программного модуля, проведен обзор пользовательского интерфейса, исследованы требования к надежности и программной совместимости ПМ ФИО.

В технологическом разделе описаны технологии программирования и отладки, архитектура взаимодействия ПМ ФИО с системой 1С-Битрикс, проведен обзор методов тестирования и описан процесс тестирования ПМ ФИО.

В приложении 1 приведен программный код ПМ ФИО.

В приложении 2 приведено руководство программиста.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

В данном разделе рассмотрена актуальность выбранной темы, поставлены цель и задача разработки, проведены обзор существующих решений и исследование предметной области, проанализированы потребности потенциальных потребителей и составлены функциональные требования к разрабатываемому программному модулю.

1.1. Актуальность выбранной темы

Интеллектуальная обработка данных (ИОД) – набор технологий и методов для извлечения из необработанных данных ранее неизвестных, неочевидных, нетривиальных знаний и закономерностей, необходимых для принятия решений в различных сферах человеческой деятельности. Найденные знания должны быть применимы и на новых данных с некоторой степенью достоверности. Полезность заключается в том, что эти знания могут приносить определенную выгоду при их применении. [11]

Сфера применения технологий ИОД широка: в нее входят как различные бизнес-задачи, такие как банковское дело, финансы, страхование, электронная коммерция, маркетинг, так и исследовательские задачи (среди основных направлений: медицина, биология, молекулярная генетика и генная инженерия, биоинформатика, астрономия и другие). [15]

Основные задачи, решаемые при помощи методов интеллектуальной обработки данных: [11]

- ассоциация (выявление закономерностей между связанными объектами);
- классификация (разбиение объектов на предопределенные классы);
- кластеризация (разбиение объектов на кластеры, отличается от классификации тем, что кластеры заранее не определены);
- прогнозирование (анализ закономерностей временных рядов и прогноз на их основе значений в будущем);
- последовательность (выявление закономерностей между связанными во времени событиями).

В данной работе решаются задачи кластеризации и ассоциации, как одни из самых распространенных в сфере интернет-приложений. Для систем управления сайтами методы кластеризации и поиска ассоциативных правил применяются для решения задач выявления похожих товаров и покупателей, рекомендации товаров, выделения групп пользователей и анализа их поведения, анализа логов веб-сайтов. Исходя из этого, актуальным является внедрение технологий ИОД в системы СУС.

Среди СУС выделяется система 1С-Битрикс как высокопроизводительная система, предназначенная для разработки и поддержки крупных веб-проектов, таких как корпоративные сайты, интернет-магазины и информационные порталы. [9] Кроме того, система постоянно развивается, в нее добавляются новые возможности, в том числе поиск похожих товаров и сервис рекомендации. Но, поскольку система является закрытой, разработчику нельзя изменить поведение этих функций под конкретный проект, а те возможности модификации, что предоставляет система 1С-Битрикс, являются ограниченными. Поэтому тематика данной работы, состоящая в повышении эффективности работы с данными в системе 1С-Битрикс путем добавления программного модуля для реализации функций ИОД, является актуальной.

1.2. Цель и задачи разработки

Цель разработки: повышение эффективности обработки данных в системе 1С-Битрикс путем внедрения технологии интеллектуальной обработки данных.

Задачи разработки:

- исследование предметной области;
- сравнительный анализ существующих решений;
- выбор инструментальных средств и среды разработки;
- разработка схемы данных ПМ ФИО;
- разработка схем алгоритмов ПМ ФИО;
- программная реализация ПМ ФИО;
- разработка пользовательского интерфейса ПМ ФИО;
- отладка и тестирование ПМ ФИО;
- разработка руководства программиста ПМ ФИО.

1.3. Обзор существующих решений

Среди существующих решений для системы 1С-Битрикс можно выделить сервис рекомендаций RetailRocket, сервис персонализации «1С-Битрикс BigData» и готовое решение от компании Сотбит «Похожие товары – как инструмент увеличения среднего чека».

RetailRocket – это сервис персональных рекомендаций, предназначенный для мультиканальной персонализации пользователей интернет-магазина на основе технологии BigData. Данный сервис выявляет потребности посетителей вашего интернет-магазина и в нужный момент делает интересные именно им предложения на сайте, увеличивая доход интернет-магазина за счет роста конверсии, среднего чека и частоты повторных покупок. К недостаткам сервиса относятся необходимость передачи данных во внешний сервис для получения персональных рекомендаций, а также закрытость программного кода. [33]

«1С-Битрикс BigData» – облачный сервис персонализации, являющийся составной частью платформы «1С-Битрикс». Использование сервиса нацелено на рост интернет-бизнеса путем создания более персонализированных отношений с клиентами. Сервис повышает качество управления, уровень продаж и конверсию в интернет-магазине. К недостаткам сервиса относится закрытость программного кода, невозможность модификации функций сервиса, а также узкая направленность применения сервиса. [2][3]

Готовое решение от компании Сотбит «Похожие товары – как инструмент увеличения среднего чека» представляет собой компонент, позволяющий выводить похожие товары для текущего товара. Компонент работает в двух режимах:

- вывод похожих товаров карточке товара;
- вывод похожих товаров в корзине.

Первый режим является стандартным. В нем указывается ID товара, относительно которого будут фильтроваться похожие товары. Во втором режиме параметр Идентификатор товара отсутствует, так как поиск похожих товаров осуществляется относительно товаров, находящихся в корзине. В корзине могут находиться как торговые предложения, так и просто товары.

Режим вывода похожих товаров в корзине привлечет внимание покупателя к данным товарам, что позволит увеличить средний чек интернет-магазина. К недостаткам данного компонента можно отнести отсутствие рекомендации товаров.

Сравнение существующих решений с ПМ ФИО приведено в таблице 1.

Таблица 1 – Сравнение с существующими решениями

Параметры	RetailRocket[1]	1С-Битрикс BigData[2]	Сотбит Похожие товары3	ПМ ФИО
Наличие кластеризации товаров	-	-	+	+
Рекомендация товаров	+	+	-	+
Необходимость передачи данных во внешние сервисы	+	-	+	-
Открытость кода, возможность модификации/добавления функций	-	-	-	+
Стоимость лицензии, руб.	1700	0	3000	Планируется открытая лицензия

Источники:
[1] https://retailrocket.ru/
[2] http://www.1c-bitrix.ru/
[3] http://www.sotbit.ru/

Условные обозначения:
+ – указанная возможность присутствует
- – указанная возможность отсутствует

В результате сравнения получаем, что ПМ ФИО объединяет в себе встраиваемость решения 1С-Битрикс BigData и возможность использования сторонних библиотек, а также возможность добавления/модификации функций модуля.

1.4. Исследование предметной области

1.4.1. Интеллектуальная обработка данных

Основная особенность ИОД – это сочетание широкого математического инструментария и последних достижений в сфере информационных технологий. В технологии ИОД гармонично объединились строго формализованные методы и методы неформального анализа, т.е. количественный и качественный анализ данных. [15]

К методам и алгоритмам ИОД относятся следующие: [11]

- искусственные нейронные сети;
- деревья решений;

- символьные правила;
- методы ближайшего соседа и k-ближайшего соседа;
- метод опорных векторов;
- байесовские сети;
- линейная регрессия;
- корреляционно-регрессионный анализ;
- иерархические методы кластерного анализа;
- неиерархические методы кластерного анализа (в том числе алгоритмы k-средних, c-средних и k-медианы);
- методы поиска ассоциативных правил (в том числе алгоритм Apriori);
- метод ограниченного перебора;
- эволюционное программирование и генетические алгоритмы.

Большинство аналитических методов, используемые в технологии ИОД – это известные математические алгоритмы и методы. Новым в их применении является возможность их использования при решении тех или иных конкретных проблем, обусловленная появившимися возможностями технических и программных средств. Следует отметить, что большинство методов ИОД были разработаны в рамках теории искусственного интеллекта. [15]

1.4.2. Задачи ИОД

К наиболее распространенным задачам ИОД относятся:

- Классификация – одна из самых простых и распространенных задач ИОД. В результате решения задачи классификации обнаруживаются признаки, которые характеризуют группы объектов исследуемого набора данных – классы, далее по этим признакам новый объект можно отнести к тому или иному классу. [6] Для решения задачи классификации могут использоваться методы:
  - ближайшего соседа;
  - k-ближайшего соседа;

- байесовские сети;
- индукция деревьев решений;
- нейронные сети.

- Кластеризация является логическим продолжением задачи классификации. Особенность кластеризации заключается в том, что классы объектов изначально не предопределены. Результатом кластеризации является разбиение объектов на группы – кластеры. К методам решения задачи кластеризации относятся: [27]
  - иерархические методы кластеризации;
  - неиерархические методы кластеризации.
- Ассоциация. В ходе решения задачи поиска ассоциативных правил отыскиваются закономерности между связанными событиями в наборе данных. Отличие ассоциации от задач классификации и кластеризации состоит в том, что поиск закономерностей осуществляется не на основе свойств анализируемого объекта, а между несколькими событиями, которые происходят одновременно (например, покупка одного товара и покупка другого товара). К наиболее известным алгоритмам решения задачи ассоциации относятся: [26]
  - алгоритм AIS;
  - алгоритм SETM;
  - алгоритм Apriori.
- Последовательность позволяет найти временные закономерности между транзакциями. Задача последовательности подобна ассоциации, но ее целью является установление закономерностей не между одновременно наступающими событиями, а между событиями, связанными во времени (т.е. происходящими с некоторым определенным интервалом во времени). Другими словами, последовательность определяется высокой вероятностью цепочки связанных во времени событий. Методы решения данной задачи аналогичны методам решения задачи ассоциации. [11]
- Прогнозирование – задача, в результате решения которой на основе известных данных оцениваются будущие значения целевых численных показателей. К основным методам решения задачи прогнозирования относятся: [15]

- методы математической статистики;
- линейная регрессия;
- нейронные сети.

В рамках применения методов ИОД для интернет-приложений наиболее распространенными задачами являются задачи кластеризации и ассоциации. Их можно применять для поиска похожих товаров, сегментации клиентов, анализа веб-логов, рекомендации товаров и т.д.

В данной разработке реализованы алгоритмы для решения задач кластеризации и ассоциации.

1.4.3. Задача кластеризации

Кластеризация – это задача разбиения объектов на группы (кластеры). Кластер должен состоять из однотипных объектов с максимальным количеством аналогичных параметров. Основное отличие кластеризации от классификации состоит в том, что при кластеризации заранее не определены классы, на которые разбиваются объекты, и определяются в процессе работы алгоритма. Таким образом, целью кластеризации является поиск существующих групп (кластеров) данных. [8]

Существует две основных классификации методов решения задачи кластеризации:

1. Четкие и нечеткие. Четкой кластеризацией называют такое разбиение наборов на кластеры, при котором каждый набор принадлежит только одному кластеру. При нечеткой кластеризации один и тот же набор может принадлежать с разной вероятностью разным кластерам. [12]
2. Иерархические и неиерархические. Иерархические методы основаны на рекурсивном разбиении множества наборов на кластеры, при этом в результате получается система вложенных разбиений. Неиерархические (или плоские) методы строят одно разбиение исходного множества наборов на кластеры. [12]

В любом методе решения задачи кластеризации необходимо выбрать метрику, т.е. функцию расстояния между наборами данных. Основными метриками являются (далее x, y – наборы данных, x_i, y_i – координаты наборов, $\rho(x, y)$ – расстояние между наборами $x и y$):

- Евклидова метрика – наиболее распространенная метрика. Вычисляется как расстояние между двумя векторами в многомерном пространстве:

$$\rho(x, y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \quad (1)$$

- Квадрат евклидовой метрики – применяется для увеличения расстояния между далекими друг от друга наборами:

$$\rho(x, y)=\sum_{i=1}^{n}(x_i-y_i)^2 \quad (2)$$

- Расстояние городских кварталов (или Манхэттенское расстояние). Применяется для усреднения влияния больших разностей (выбросов) на общее расстояние:

$$\rho(x, y)=\sum_{i=1}^{n}|x_i-y_i| \quad (3)$$

- Расстояние Чебышева применяется в случае, когда требуется наборы считать различными, если они отличаются по какой-то координате:

$$\rho(x, y)=\max_i|x_i-y_i| \quad (4)$$

Выбор метрики может существенно повлиять на качество кластеризации и обычно зависит от выбранного алгоритма кластеризации.

Сравнение алгоритмов кластеризации дано в таблице 2.

Таблица 2 – Сравнение алгоритмов кластеризации

Алгоритм	Вычислительная сложность	Тип кластеризации
Иерархический [1]	$O(n^2)$	Четкая иерархическая
с-средних [2]	$O(nkl)$, где k – число кластеров, l – число итераций	Нечеткая плоская
Минимальное покрывающее дерево [3]	$O(n^2 \log n)$	Четкая иерархическая
Послойная кластеризация [4]	$O(n^2)$	Четкая иерархическая

Источники:

[1] Жамбю М. Иерархический кластер-анализ и соответствия. – М.: Финансы и статистика, 1988. – 345 с.

[2] Воронцов К.В. Алгоритмы кластеризации и многомерного шкалирования, 2007

[3] Мандель И. Д. Кластерный анализ. – М.: Финансы и Статистика, 1988

[4] Граничина Н.О., Шалымов Д.С. Рандомизированный алгоритм устойчивой кластеризации. СПбГУ, 2009.

Для решения задачи кластеризации выбран метод с-средних, являющегося модификацией метода k-средних. [6] Данный алгоритм является нечетким и обладает наибольшей эффективностью по сравнению с остальными алгоритмами кластеризации при оптимальном выборе числа кластеров. Также нечеткость кластеризации данного алгоритма позволяет узнать, с какой вероятностью данный элемент относится к тому или иному кластеру.

1.4.4. Алгоритм кластеризации с-средних

Алгоритм с-средних является неиерархическим алгоритмом нечеткой кластеризации. Основан на использовании нечетких множеств, т.е. множеств, чья функция принадлежности может давать значения не только {0;1}, но и значения из отрезка [0;1]. [31]

Вводится понятие нечеткой матрицы принадлежности *U*, каждый элемент U_{ij} показывает вероятность принадлежности набора x_i кластеру *j*.

Основные шаги алгоритма с-средних: [32]

1. Выбрать начальное нечеткое разбиение *N* объектов на *K* кластеров путем выбора матрицы принадлежности *U* размера $N \times K$.
2. Используя матрицу *U*, найти значение критерия E нечеткой ошибки:

$$E^2(X,U)=\sum_{i=1}^{N}\sum_{j=1}^{K}U_{ij}^{m}\left\|x_i-c_j\right\|^2, \qquad (5)$$

где U_{ij} – элемент матрицы принадлежности *U*, $m>1$ – степень нечеткости целевой функции, c_k – центр нечеткого кластера *k*, вычисляющегося по формуле:

$$c_k = \frac{\sum_{i=1}^{N} U_{ik}^m x_i}{\sum_{i=1}^{N} U_{ik}^m} \quad (6)$$

3. Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки путем пересчета матрицы *U*:

$$U_{ij} = \sum_{t=1}^{K} \left(\frac{\|x_i - c_j\|}{\|x_i - c_t\|} \right)^{\frac{2}{1-m}} \quad (7)$$

4. Возвращаться в п. 2 до тех пор, пока не будет выполнено условие:

$$\min_{ij} \left| U_{ik}^{(s+1)} - U_{ik}^{(s)} \right| < \varepsilon, \quad (8)$$

где $\varepsilon \in (0;1)$ – критерий останова, *s* – номер итерации, $U_{ik}^{(s)}$ и $U_{ik}^{(s+1)}$ – значения матрицы U на шагах *s* и *s*+1 соответственно.

С каждой итерацией целевая функция (5) стремится к локальному минимуму (седловой точке).

Входными данными алгоритма являются наборы данных $x_1, x_2, \ldots, x_n$, количество кластеров *K* и степень нечеткости *m*. Выходными данными – значения матрицы принадлежности *U*.

1.4.5. Задача поиска ассоциативных правил

Ассоциация имеет место в случае, если одно событие влечет за собой другие. Например, «при покупке банки краски в магазине в 50% приобретут также и кисточку для краски». Подобные правила называются правилами ассоциации и обозначаются импликацией ($X \Longrightarrow Y$). Задача ассоциации состоит в поиске таких правил. Вводятся следующие понятия: [7]

- транзакция – набор *T* из элементов исходного множества *X*;
- поддержка правила $X \Longrightarrow Y$ – частота встречаемости набора *{X,Y}* в транзакциях, обозначается $supp(X \Longrightarrow Y)$, $supp(X \bigcup Y) = supp(X \Longrightarrow Y)$;
- достоверность правила $X \Longrightarrow Y$ – вероятность того, что из *X* следует *Y*, вычисляется по формуле

$$conf(X \Longrightarrow Y) = \frac{supp(X \bigcup Y)}{supp(X)} \quad (9)$$

Задача ассоциации состоит в поиске таких ассоциативных правил $X \Longrightarrow Y$, поддержка которых $supp(X \Longrightarrow Y)$ превышает заданную минимальную поддержку *min_supp*, а достоверность $conf(X \Longrightarrow Y)$ больше заданной минимальной достоверности min_{conf}. [26]

Сравнение алгоритмов поиска ассоциативных правил дано в таблице 3.

Таблица 3 – Сравнение алгоритмов поиска ассоциативных правил

Алгоритм	Особенности работы алгоритма
AIS [1]	Кандидаты множества наборов генерируются и подсчитываются "на лету", во время сканирования базы данных.
SETM [2]	Модификация алгоритма AIS с использованием операций языка SQL
Apriori [3]	Модификация алгоритмов AIS и SETM

Источники:

[1] https://basegroup.ru/community/articles/intro

[2] Savasere A., Navathe S. An Efficient Algorithm for Mining Association, 1995

[3] R. Agrawal, R. Srikant. "Fast Discovery of Association Rules", 1994

Для решения задачи поиска ассоциативных правил выбран алгоритм Apriori, поскольку данный алгоритм совмещает в себе достоинства алгоритмов AIS и SETM, при этом решая проблему излишнего генерирования кандидатов. [51]

1.4.6. Алгоритм ассоциации Apriori

Алгоритм Apriori использует поиск в ширину и осуществляет его снизу-вверх. Он основан на свойстве антимонотонности: при увеличении размера набора его поддержка не увеличивается. Это свойство позволяет использовать значения, полученные на предыдущих шагах алгоритма, снизив число обращений к БД на каждой итерации, и не генерировать лишних кандидатов. [49]

Кандидатом называется часто встречающийся набор данных, т.е. набор, поддержка которого превышает заданную минимальную поддержку *min_supp*.

Алгоритм Apriori состоит из двух этапов. На первом этапе формируется список часто встречающихся наборов, состоящих из одного элемента. Для этого надо пройтись по всему набору данных и подсчитать для них поддержку. Следующие шаги будут состоять из двух частей: генерации потенциально часто встречающихся наборов элементов (кандидатов) и подсчета их поддержки. Генерация кандидатов также будет состоять из двух шагов: [50]

1. Объединение. Каждый кандидат C_k будет формироваться путем расширения часто встречающегося набора размера (k-1) добавлением элемента из другого (k-1)-элементного набора.
2. Удаление избыточных правил. На основании свойства антимонотонности следует удалить все наборы $c \in C_k$, если хотя бы одно из его (k-1)-элементных подмножеств не является часто встречающимся.

После того как найдены все часто встречающиеся наборы элементов, можно приступить непосредственно к генерации правил. Извлечение правил – менее трудоемкая задача. Во-первых, для подсчета достоверности правила достаточно знать поддержку самого набора и множества, лежащего в условии правила. Например, имеется часто встречающийся набор {A, B, C} и требуется подсчитать достоверность для правила AB ⇒ C. Поддержка самого набора нам известна, но и его множество {A, B}, лежащее в условии правила, также является часто встречающимся в силу свойства анти-монотонности, и значит, его поддержка нам известна. Тогда мы легко сможем подсчитать достоверность правила AB ⇒ C. Это избавляет нас от нежелательного просмотра базы транзакций, который потребовался в том случае если бы это поддержка была неизвестна. [49]

Чтобы извлечь правило из часто встречающегося набора F, следует найти все его непустые подмножества. И для каждого подмножества s мы сможем сформулировать правило s ⇒ (F – s), если достоверность правила

$$conf(s \Rightarrow (F - s)) = \frac{supp(F)}{supp(s)} \qquad (10)$$

не меньше порога *min_conf*. Заметим, что числитель остается постоянным. Тогда достоверность имеет минимальное значение, если знаменатель имеет максимальное значение, а это происходит в том случае, когда в условии правила имеется набор, состоящий из одного элемента. Все надмножества данного множества имеют меньшую или равную поддержку и, соответственно, большее значение достоверности. Это свойство может быть использовано при извлечении правил. Если мы начнем извлекать правила, рассматривая сначала только один элемент в условии правила, и это правило имеет необходимую поддержку, тогда все правила, где в условии стоят надмножества этого элемента, также имеют значение достоверности выше заданного порога. Например, если правило A ⇒ BCDE удовлетворяет минимальному порогу достоверности minconf, тогда AB ⇒ CDE также удовлетворяет. Для того, чтобы извлечь все правила используется рекурсивная процедура. Сделаем важное замечание: любое правило, составленное из часто

встречающегося набора, должно содержать все элементы набора. Например, если набор состоит из элементов {A, B, C}, то правило A ⇒ B не должно рассматриваться. [50]

1.5. Анализ потребностей потенциальных потребителей

Потенциальными потребителями ПМ ФИО являются разработчики проектов, разрабатываемых с использованием системы 1С-Битрикс. Поэтому данный модуль должен удовлетворять основным концепциям системы 1С-Битрикс, а также быть интуитивно понятным в применении для пользователей данной системы. Основными потребностями потенциальных пользователей ПМ ФИО являются:

- возможность сегментации данных, переданных в ПМ ФИО;
- возможность поиска похожих товаров (кластеризация товаров);
- рекомендованные товары (поиск ассоциативных правил);
- возможность добавление новых и модификация существующих функций ПМ ФИО;

1.6. Состав выполняемых функций

На основании потребностей потенциальных пользователей разрабатываемое ПО должно обеспечить выполнение следующих функций:

- систематизацию данных;
- обработка данных;
- рекомендация товаров (поиск ассоциативных правил);
- кластеризация данных (сегментация данных, поиск похожих товаров).

1.7. Схемы данных и алгоритма ПМ ФИО.

Схемы алгоритма и схема данных ПМ ФИО изображены на рис. 1 и 2.

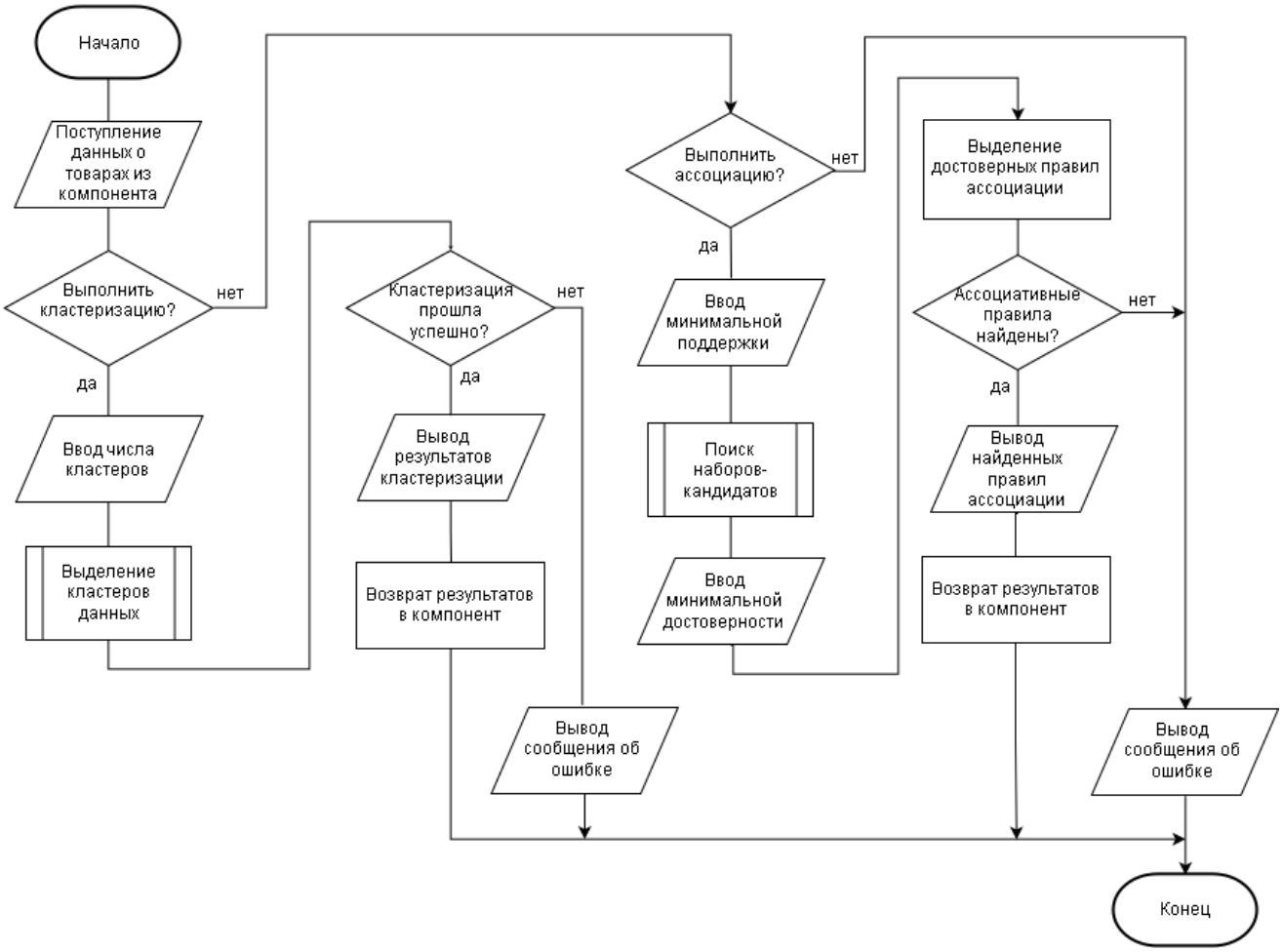

Рисунок 1 – Схема алгоритма ПМ ФИО

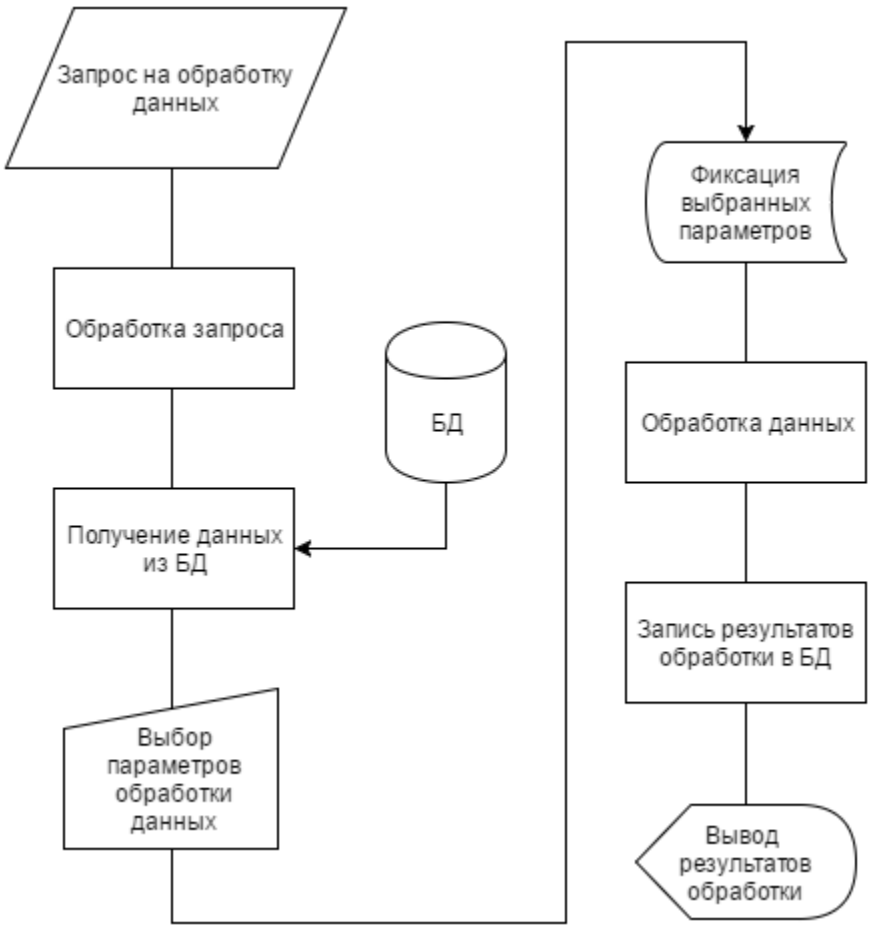

Рисунок 2 – Схема данных ПМ ФИО

Выводы

В исследовательском разделе были исследованы предметная область и потребности потенциальных пользователей, рассмотрены теоретические основы кластеризации и поиска ассоциативных правил, проведен обзор существующих решений, разработаны схема данных и схема алгоритма работы ПМ ФИО.

2. КОНСТРУКТОРСКИЙ РАЗДЕЛ

В данном разделе выбраны инструменты разработки, разработаны структуры входных и выходных данных, разработана архитектуры и алгоритм работы программного модуля, проведен обзор пользовательского интерфейса, исследованы требования к надежности и программной совместимости ПМ ФИО.

2.1. Структура входных и выходных данных

В качестве входных данных выступают данные, поступающие от компонентов системы 1С-Битрикс, вызывающих функции ПМ ФИО, а также параметры обработки этих данных.

Выходные данные для ПМ ФИО заносятся в БД системы 1С-Битрикс и представляют собой результаты обработки входных данных, т.е. описание кластеров, на которые были разбиты эти данные (в случае выбора кластеризации) и список ассоциативных правил (в случае выбора ассоциации).

2.2. Программные технологии решения поставленной задачи

Для разработки программного модуля интеллектуальной обработки данных для системы управления сайтом необходимы следующие элементы:

- язык программирования;
- система управления содержимым веб-сайта (СУС);
- среда разработки и отладчик.

2.2.1. Выбор системы управления содержимым веб-сайта

Выбор производится среди СУС, предназначенных для создания интернет-магазинов: Joomla!, WordPress и 1С-Битрикс.

Joomla! – система управления содержимым, написанная на языках PHP и JavaScript, использующая в качестве хранилища базы данных СУБД MySQL или другие стандартные

промышленные реляционные СУБД. Является свободным программным обеспечением, распространяемым под лицензией GNU GPL. [10][17]

WordPress – система управления содержимым с открытым исходным кодом; написана на PHP; сервер базы данных – MySQL; выпущена под лицензией GNU GPL версии 2. Сфера применения – от блогов до достаточно сложных новостных ресурсов и интернет-магазинов. Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать проекты широкой функциональной сложности. [52]

1С-Битрикс – высокопроизводительная система управления веб-проектами, универсальный программный продукт для создания, поддержки и развития корпоративных сайтов, интернет-магазинов, информационных порталов, сайтов сообществ и других веб-проектов. Для хранения данных сайта используется файловая система сервера и реляционная СУБД. Поддерживаются следующие СУБД: MySQL, Oracle, MS SQL. Продукт работает на Microsoft Windows и UNIX-подобных платформах, включая Linux. [2]

В таблице 4 проведен обзор вышеперечисленных СУС.

Таблица 4 – Обзор систем управления сайтом

Критерии	WordPress[1]	Joomla![2]	1С-Битрикс[3]
Набор инструментов	Минимальный	Минимальный	Более 30 модулей
Требовательность к ресурсам	высокая	низкая	высокая
Поддержка мультимедийного контента	-	-	+
Наличие тех. поддержки	-	-	+
Мультиязычность	+	+	+

Источники:	Условные обозначения:
[1] https://ru.wordpress.org/ [2] https://www.joomla.org/ [3] http://www.1c-bitrix.ru/	+ – указанная возможность присутствует - – указанная возможность отсутствует

Исходя из результатов сравнения, в качестве CMS была выбрана система 1С-Битрикс.

2.2.2. Выбор языка программирования

Выбор проводился среди серверных языков программирования Java, Python, Ruby, PHP,

Java – объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. [29]

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. [20][24]

Ruby – динамический, интерпретируемый высокоуровневый язык программирования для объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора. К минусам можно отнести сложность в обучении, отсутствие документации на русском языке, малая доля потенциальных потребителей. [37]

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. [13]

В таблице 5 приведено сравнение вышеперечисленных языков программирования.

Таблица 5 – Обзор языков программирования

Критерии	Java [1]	Python [2]	Ruby [3]	PHP [4]
Простота изучения	+	+	-	+
Наличие в CMF	-	+	+	+
Динамическая типизация	-	+	+	+

Сборщик мусора	+	+	+	+
Наличие средств для работы с БД	+	+	+	+
Возможность работы с 1С-Битрикс	-	-	-	+
Наличие опыта	+	-	-	+

Источники:
[1] https://www.java.com/
[2] https://www.python.org/
[3] https://www.ruby-lang.org/ru/
[4] https://php.net/

Условные обозначения:
+ – указанная возможность присутствует
- – указанная возможность отсутствует

Исходя из результатов сравнения, в качестве языка программирования выбран язык программирования PHP.

2.2.3. Выбор среды разработки

Выбор проводился среди сред разработки с возможностью разработки на языке программирования PHP: Notepad++, Sublime Text, NetBeans IDE, PhpStorm.

Notepad++ – свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки. Поддерживает открытие более 100 форматов. Распространяется под лицензией GNU GPL. Базовая функциональность программы может быть расширена как за счёт плагинов, так и сторонних модулей, таких как компиляторы и препроцессоры. [28]

Sublime Text – кроссплатформенный проприетарный текстовый редактор. Поддерживает плагины на языке программирования Python. Основные возможности: быстрая навигация, командная палитра, одновременное редактирование, высокая степень настраиваемости, кросс-платформенность (Windows, OS X, Linux). [53]

NetBeans IDE – свободная интегрированная среда разработки приложений на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада и ряда других. Поддерживает рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода. Для разработки программ в среде NetBeans и успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE

SDK подходящей версии. Среда разработки NetBeans по умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 NetBeans поддерживает разработку для мобильных платформ J2ME, C++ (только g++) и PHP без установки дополнительных компонентов. [45]

JetBrains PhpStorm – коммерческая кросс-платформенная интегрированная среда разработки для PHP. PhpStorm представляет собой интеллектуальный редактор для PHP, HTML и JavaScript с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами рефакторинга для PHP и JavaScript. PhpStorm разработан на основе платформы IntelliJ IDEA, написанной на Java. Пользователи могут расширить функциональность среды разработки за счет установки плагинов, разработанных для платформы IntelliJ, или написав собственные плагины. [48]

В таблице 6 проведено сравнение сред разработки.

Таблица 6 – Обзор сред разработки

Критерии	Notepad++ 6.8 [1]	Sublime Text 3 [2]	NetBeans 8.1 [3]	PHPStorm 3 [4]
Создание проектов	-	-	+	+
Возможность отладки	-	-	+	+

Продолжение таблицы 6

Критерии	Notepad++ 6.8 [1]	Sublime Text 3 [2]	NetBeans 8.1 [3]	PHPStorm 3 [4]
Поддержка автодополнения	+	+	+	+
Автоформатирование кода	-	-	+	+
Тип лицензирования	Не требуется	+	Не требуется	от 5 000 руб./год

Источники:
[1] https://notepad-plus-plus.org/
[2] https://www.sublimetext.com/
[3] https://netbeans.org/
[4] https://www.jetbrains.com/phpstorm

Условные обозначения:
+ – указанная возможность присутствует
- – указанная возможность отсутствует

В качестве среды разработки выбрана NetBeans 8.1.

2.3. Программная архитектура и алгоритм работы

По результатам исследовательской части можно сделать вывод, что для реализации поставленной задачи необходимо следовать концепциям ООП.

Для реализации алгоритма Apriori будем использовать следующие классы:

- AprioriAlgorithm – класс, реализующий алгоритм Apriori;
- Candidate – класс кандидатов (часто встречающихся наборов) в алгоритме Apriori;
- AssocRule – класс ассоциативных правил.

Для реализации алгоритма с-средних будут использованы следующие классы:

- FuzzyCMeans – класс, реализующий алгоритм с-средних;
- FuzzyCMeansPoint – класс, соответствующий наборам данных в алгоритме с-средних.

Такая структура классов позволяет сопоставить каждой сущности, участвующих алгоритмах, отдельный класс, а также делает модуль независимым от типа и количества входных данных (поскольку они представлены отдельными классами от остальной реализации алгоритмов).

На рисунке 3 изображена диаграмма классов ПМ ФИО:

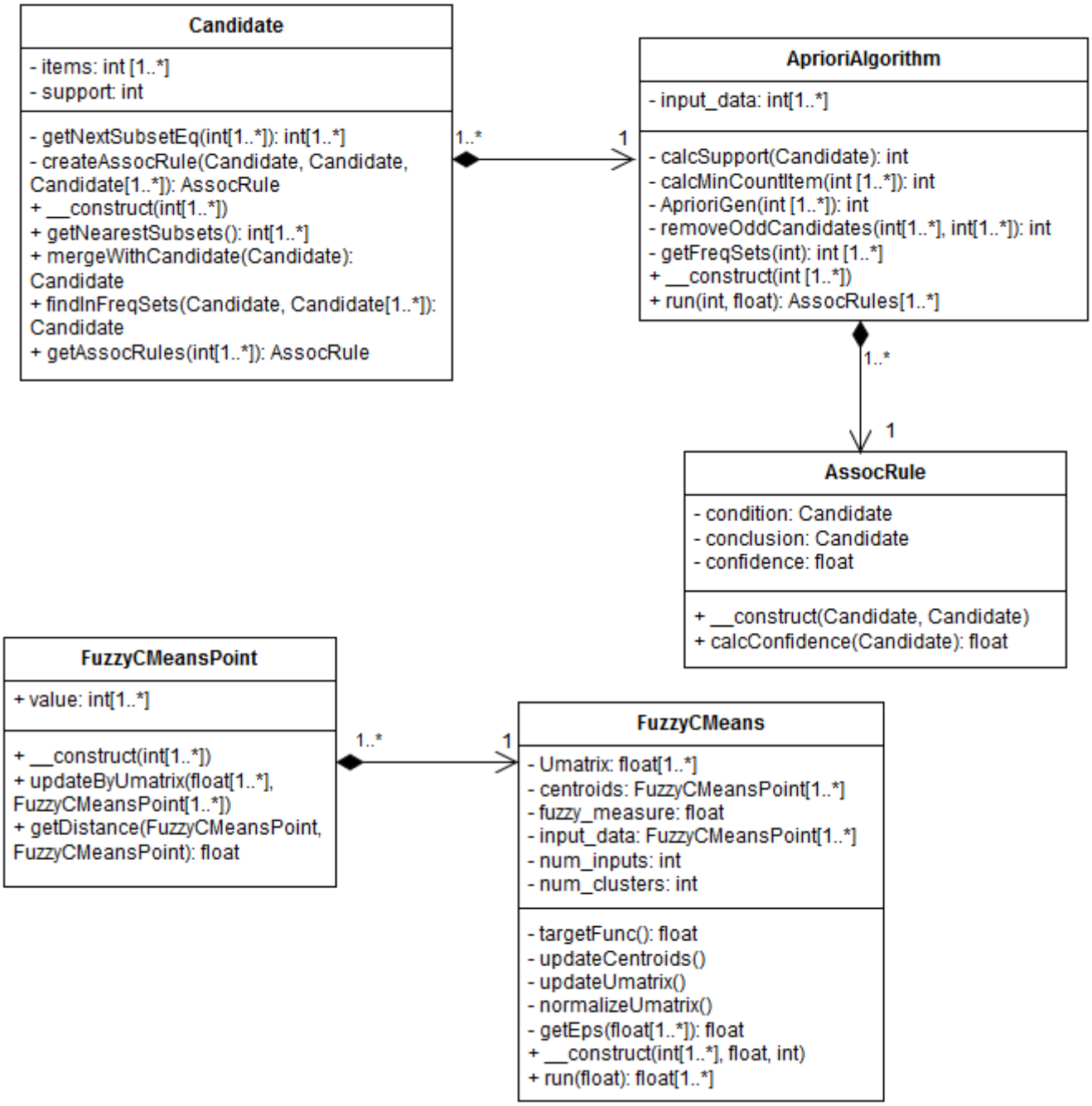

Рисунок 3 – Диаграмма классов ПМ ФИО

Рассмотрим подробнее некоторые детали реализации:

- все наборы данных кластеризации передаются в виде объектов класса FuzzyCMeansPoint, содержимое набора содержится в поле value объектов, функция getDistance – расстояние между наборами данных – может быть переопределена путем наследования класса FuzzyCMeansPoint и определения собственной реализации функции. Это обеспечит гибкость и независимость от типа и количества входных данных – разработчик сам может определить поведение их как объектов класса FuzzyCMeansPoint;

- для запуска кластеризации в классе FuzzyCMeans необходимо использовать функцию run() (параметр которой – критерий останова алгоритма). Далее каждый метод соответствует шагам алгоритма с-средних:
    - updateCentroid() – обновление координат центров кластеров;
    - updateUMatrix() – пересчет матрицы принадлежности;
    - normalizeUMatrix() – нормализация матрицы принадлежности;
    - getEps() – проверка критерия останова алгоритма.
- для запуска поиска ассоциативных правил в классе AprioriAlgorithm следует применить функцию run() (параметры которой – минимальная поддержка и минимальная достоверность). На вход класса AprioriAlgorithm поступают наборы данных, а на выходе – ассоциативные правила с подсчитанной достоверности (реализованные в виде отдельного класса AssocRule). Кандидаты также реализованы в виде отдельного класса Candidate. Аналогично реализации с-средних, каждый метод в AprioriAlgorithm соответствует шагам алгоритма:
    - getFreqSets() – получение списка часто встречающихся наборов (кандидатов);
    - calcSupport() – подсчет поддержки для кандидатов;
    - AprioriGen() – генерация кандидатов;
    - removeOddCandidates() – удаление избыточных кандидатов.

Для обеспечения возможности дальнейшего развития и использования ПМ ФИО в других системах управления содержимым, структуру классов реализуем в виде отдельной библиотеки классов. Для этого в модуле 1С-Битрикс в файловой структуре должна быть определена папка classes, в которую необходимо положить файлы классов.

Для возможности установки модуля необходимо, чтобы он соответствовал структуре модулей 1С-Битрикс [35]:

- admin/ – каталог с административными скриптами модуля;
- menu.php – файл с административным меню модуля;
- classes/ – скрипты с классами модуля;
- general/ – классы модуля, не зависящие от используемой базы данных;
- lang/ID языка/ – каталог с языковыми файлами скриптов модуля;

- install/ – каталог с файлами используемыми для инсталляции и деинсталляции модуля;
- admin/ – каталог со скриптами подключающими административные скрипты модуля (вызывающие скрипты);
- js/ – каталог с js-скриптами модуля. Копируются в /bitrix/js/ID_модуля/;
- images/ – каталог с изображениями используемыми модулем; после инсталляции модуля они должны быть скопированы в каталог /bitrix/images/ID модуля/;
- components/пространство имен/имя компонента/ – каталог с компонентами модуля;
- panel/имя_модуля/ – содержит css и картинки для стилей административной панели, если модуль в таковых нуждается.
- index.php – файл с описанием модуля;
- version.php – файл с номером версии модуля. Версия не может быть равной нулю.
- include.php – данный файл подключается в тот момент, когда речь идет о подключении модуля в коде, в нем должны находиться включения всех файлов с библиотеками функций и классов модуля;
- default_option.php – содержит массив с именем $ID модуля_default_option, в котором заданы значения по умолчанию для параметров модуля.

Такая структура модуля позволит устанавливать его из административной панели.

Для получения функций модуля в 1С-Битрикс необходимо создать компонент. Компонент – это логически завершённый код, предназначенный для извлечения информации из инфоблоков и других источников и преобразования её в HTML-код для отображения в виде фрагментов web-страниц. Состоит из собственно компонента (контроллер) и шаблона (представление). Компонент, с помощью API одного или нескольких модулей, манипулирует данными. Шаблон компонента выводит данные на страницу. Таким образом, компонент в 1С-Битрикс – это реализация паттерна MVC (модель-вид-контроллер). [9]

Для подключения ПМ ФИО в коде компонента необходимо написать следующий код:

```
CModule::includeModule('sdmm_module');
```

После этого в компоненте будут доступны функции модуля.

В модуль ПМ ФИО включены следующие компоненты:

- similar.products – компонент для поиска похожих товаров. Входными данными компонента являются номер инфоблока товаров, номер товара в этом инфоблоке и поля инфоблока, по которым необходимо производить поиск похожих товаров. Выходными данными является список похожих товаров. Для реализации применены методы кластеризации, реализованные в ПМ ФИО.
- recommended.products – компонент для поиска рекомендуемых товаров, т.е. товаров, которые покупают вместе с данным. Входными данными компонента являются номер инфоблока товаров, номер товара в этом инфоблоке и поля инфоблока, по которым необходимо производить поиск товаров, покупаемых вместе с данным товаром. Выходными данными является список рекомендуемых товаров. Для реализации применены методы поиска ассоциативных правил из ПМ ФИО.
- similar.users – компонент для поиска похожих пользователей. Входными данными являются номер пользователя и поля инфоблока пользователей, по которым необходимо производить поиск похожих пользователей. Выходными данными являются список похожих на данного пользователей. Для реализации применены методы кластеризации из ПМ ФИО.

Также имеется возможность редактировать эти компоненты и добавлять свои, основанные на применении методов кластеризации и ассоциации ПМ ФИО.

Опишем на примере реализации компонента similar.products структуру компонентов ПМ ФИО. Как и любой компонент в системе 1С-Битрикс, компонент similar.products имеет следующую структуру файлов:

- .description.php – файл с описанием компонента, используется при установке компонента;
- .parameters.php – файл с описанием входных параметров компонента. Параметры должны описаны в массиве с названием arParameters следующим образом:

```
$arComponentParameters = array(
```

```
    "PARAMETERS" => array(
        "имя_параметра" => array(
            "PARENT" => "группа_параметра",
            "NAME" => "название_параметра",
            "TYPE" => "тип_параметра",
        ),
    ),
);
```

Группа параметра указывает раздел, к которому относится параметр (BASE – основной, DATA SOURCE – источники данных), название параметра будет выводится при установке настроек компонента через панель администрирования 1С-Битрикс, тип параметра – это тип данных данного параметра (L – список, N – числовой, S – строковой).

- component.php – файл, в котором описана логика компонента. В этом файле производятся все операции чтения/записи из инфоблоков, а также вызываются методы используемых модулей. Входные параметры передаются в этот файл через массив arParams. Внутри файла component.php может быть использование кэширование компонента с помощью метода

```
bool
StartResultCache([
 int cacheTime [,
 string additionalCacheID [,
 string cachePath]]]
);
```

Данный метод позволяет кэшировать выходные данные компонента на время cacheTime в файл cachePath. Кэширование данных компонента позволяет не выполнять операции чтения/записи в БД при каждом включении данного компонента, вместо этого напрямую выводя сохраненный при первом вызове компонента вывод для текущих параметров. Дополнительные параметры кэширования могут быть указаны в аргументе additionalCacheID.

- templates/ – папка шаблонов компонента. Каждый компонент может иметь несколько шаблонов вывода, каждый из которых располагается в своей папке внутри templates. В файле template.php располагается HTML-разметка выходных данных компонента, в style.css – подключаемые CSS-стили и в

scripts.js – подключаемые JS-скрипты. Выходные данные передаются в шаблон из файла component.php через массив arResult.

Для подключения компонента similar.products необходимо вызвать метод

```
$APPLICATION->IncludeComponent(
 "similar.products",
 "",
 $arParams
);
```

Первый параметр метода отвечает за имя подключаемого компонента, второй – за подключаемый шаблон компонента, третий – за входные параметры, передаваемые в компонент.

Внутри компонента шаблон подключается в файле component.php при помощи метода

```
$this->IncludeComponentTemplate();
```

при этом будет подключен тот шаблон компонента, который был указан при подключении компонента. Если в качестве шаблона была указана пустая строка, то будет подключен шаблон по умолчанию, расположенный в папке .default внутри templates.

После подключения модуля ПМ ФИО его методы вызываются в компоненте similar.products следующим образом:

1) создается объект класса FuzzyCMeans при помощи конструктора данного класса (inputData – массив входных данных, fuzzy – степень нечеткости кластеризации, num_clusters – число кластеров, на которые будут разделены данные):

```
$fcm = new FuzzyCMeans($inputData, $fuzzy, $num_clusters);
```

2) вызывается метод run() созданного объекта, аргументом которого является eps – критерий останова алгоритма, а на выходе которого получается массив кластеров, на которые были разбиты входные данные:

```
$clusters = $fcm->run($eps);
```

После этого в массиве clusters производится поиск нужного товара и выводится список похожих на него товаров, т.е. находящихся с ним в одном кластере.

Аналогичную структуру имеют компоненты рекомендуемых товаров recommended.products и похожих пользователей similar.users. Для поиска рекомендуемых

товаров в компоненте recommended.products подключается модуль ПМ ФИО и вызываются его методы поиска ассоциативных правил следующим образом:

1) создается объект класса AprioriAlgorithm при помощи конструктора данного класса (inputData – набор транзакций):

```
$apriori = new AprioriAlgorithm($inputData);
```

2) вызывается метод run() созданного объекта, аргументами которого являются минимальная поддержка кандидатов (число часто встречающихся наборов данных в транзакциях) и минимальная достоверность правил, а на выходе получается массив ассоциативных правил с рассчитанными достоверностями:

```
$assocRules = $apriori->run($min_supp, $min_conf);
```

3) Из найденных ассоциативных правил выделяются правила, удовлетворяющие минимальной достоверности, и для данного товара выводятся рекомендуемые для покупки с данным товары. Для этого применяются правила вида $X = ¿ Y1, Y2, \ldots, Yn$, где *X* – входной товар, *Y1*,...,*Yn* – рекомендуемые товары.

2.4. Этапы реализации

Разработка программного модуля состоит в реализации алгоритмов Apriori и c-средних на языке PHP (версия 5.4) и внедрение этих алгоритмов в систему 1С-Битрикс. Выбранная среда разработки – NetBeans IDE 8.1.

Этапы реализации модуля состоят из:

1) Реализация алгоритма кластеризации c-средних;
2) Реализация алгоритма поиска ассоциативных правил Apriori;
3) Создание модуля в системе 1С-Битрикс:
   a. Добавление механизма установки модуля;
   b. Настройка модуля;
   c. Добавление классов реализации алгоритмов;
4) Тестирование и отладка модуля.

2.5. Требования к надежности

Разрабатываемое ПО предназначено для работы с сетью Интернет. В связи с этим, одним из главных требований к надежности является обеспечение сохранности и

защищенность данных пользователя в процессе работы ПО. Для обеспечения надежности в ПО должно быть предусмотрено:

- отображение сообщений об ошибках при неверно заданных исходных данных;
- объем занимаемого пространства на жестком диске не должен превышать 10 Гб;
- объем используемой оперативной памяти не должен превышать 1 Гбайт;
- система не должна использовать и требовать от пользователя ввода персональных данных;
- данные, передающиеся через сеть Интернет, не должны изменять каким-либо образом данные на компьютере пользователя.

2.6. Условия эксплуатации и требования к составу и параметрам технических средств

Пользователи ПО должны иметь навыки работы на ПК и в сети Интернет, располагать общими сведениями по технологии ИОД и уметь работать с системой 1С-Битрикс. Требования к составу и параметрам технических средств представлены в таблицах 7 и 8.

Таблица 7 – Минимальный состав технических средств и их технические характеристики

Процессор	Intel Xeon E3-1225 v3 2.3 Ггц
RAM (оперативная память)	1 Гбайт
OS (операционная система)	Windows XP\Vista, Unix
HDD (объем свободного места на жестком диске)	100 Гб
Видеосистема с поддержкой разрешения экрана	800×600, SVGA

Таблица 8 – Рекомендуемый состав технических средств и их технические характеристики

Процессор	Intel Xeon E5-2650v3 3.2 Ггц
RAM (оперативная память)	8 Гбайт
OS (операционная система)	Windows 8\8.1\10, Unix
HDD (объем свободного места на жестком диске)	500 Гб
Видеосистема с поддержкой разрешения экрана	1024×768, SVGA

2.7. Требования к информационной и программной совместимости

Программное обеспечение должно работать под операционными системами Windows XP\Vista\7\8.1\10, Mac OS X.

На клиентских ПК должен быть установлен доступ к сети Интернет и установлена система «1С-Битрикс: Управление сайтом» версии 14.0 или выше.

Метод решения задачи базируется на алгоритмах кластеризации данных и поиска ассоциативных правил.

В качестве языка программирования выбран PHP 5.4. Средой разработки выбран NetBeans 8.1.

2.8. Разработка пользовательского интерфейса ПМ ФИО

Интерфейс настройки модуля ПМ ФИО находится в административной панели. В этом интерфейсе можно настроить тип и номер инфоблока входных данных, выбрать поля и свойства, участвующие в алгоритмах Apriori и с-средних, а также параметры этих алгоритмов. Интерфейс настройки изображен на рис. 4 и 5:

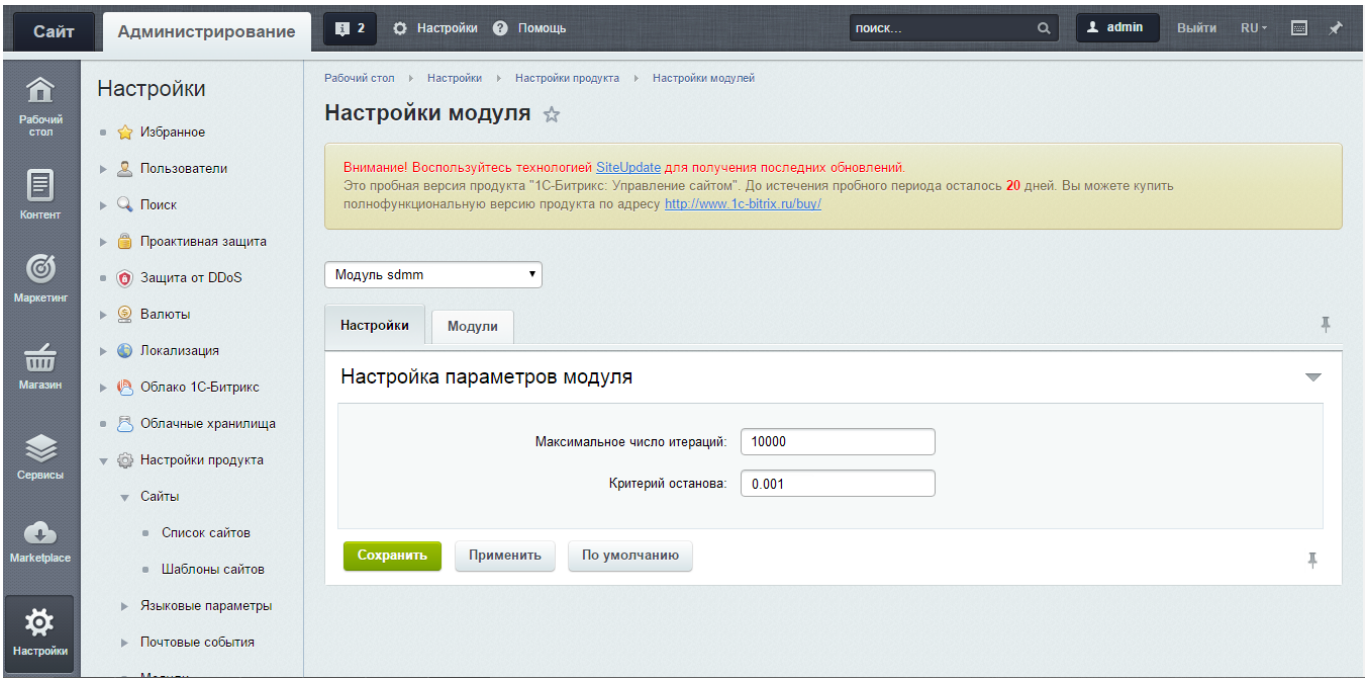

Рисунок 4 – Экранная форма настройки модуля ПМ ФИО

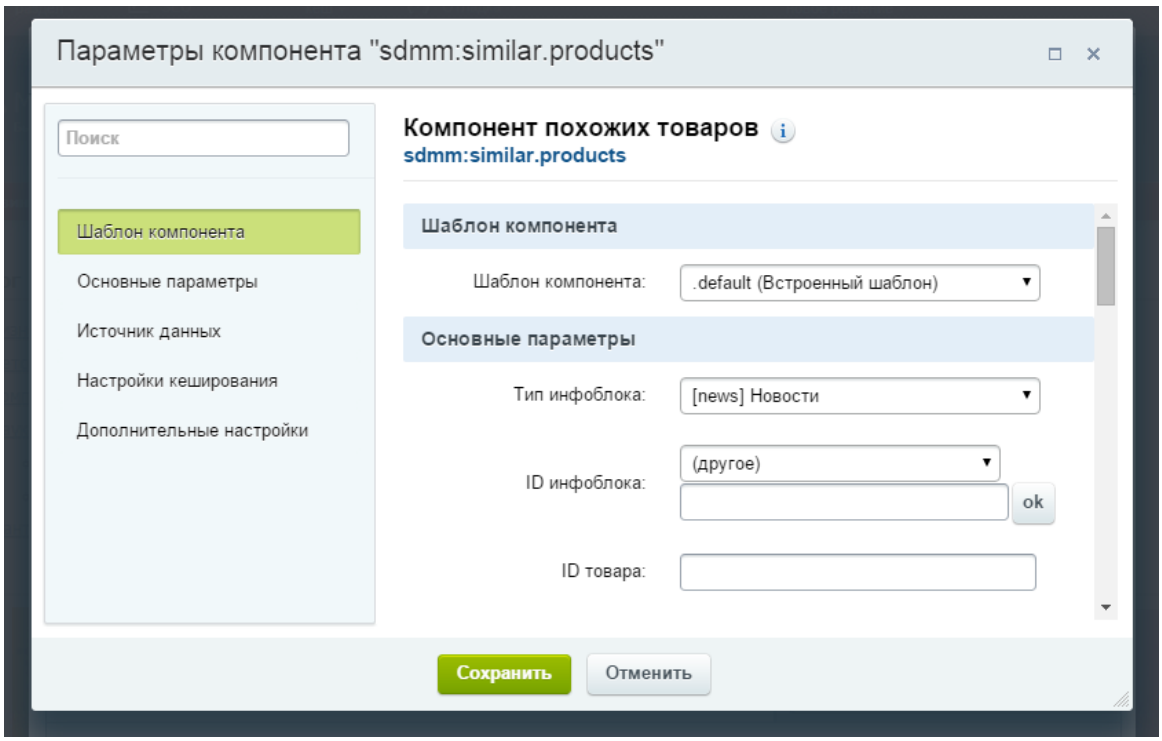

Рисунок 5 – Экранная форма настройки компонента модуля ПМ ФИО

Для примера применения ПМ ФИО возьмем демонстрационный интернет-магазин системы 1С-Битрикс. С помощью функции кластеризации был реализован компонент похожих пользователей sdmm:similar.clients. Результат изображен на рисунке 6.

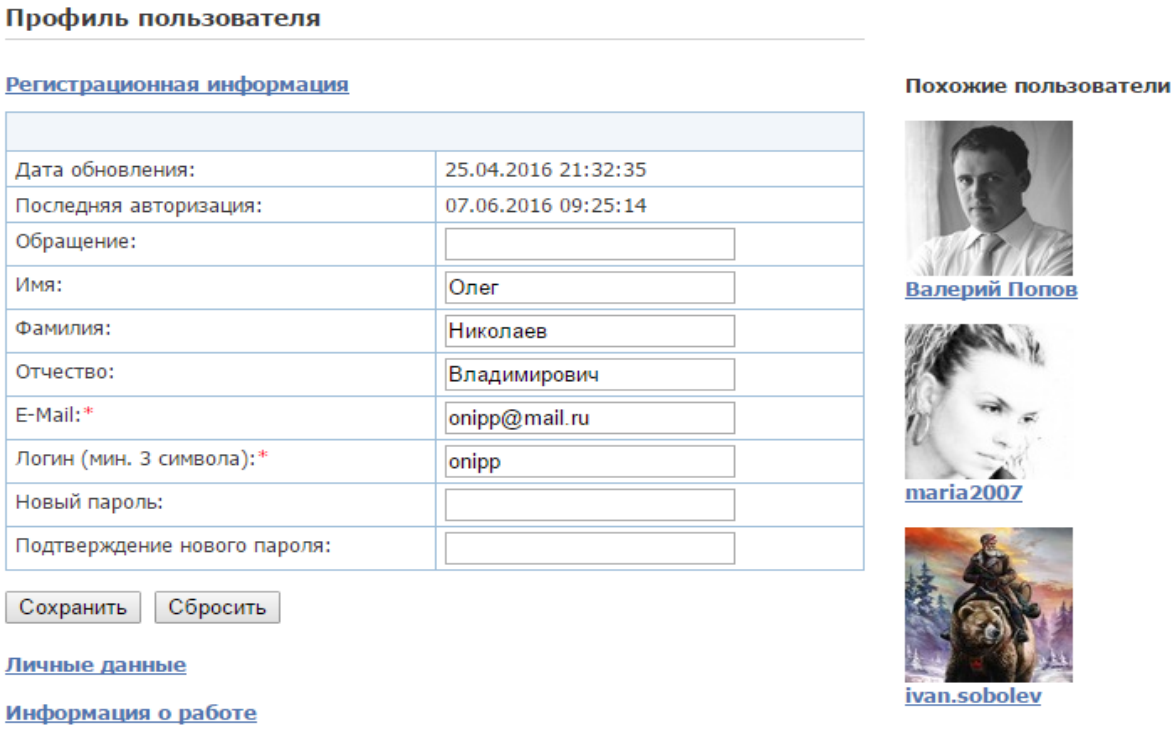

Рисунок 6 – Экранная форма страницы компонента похожих пользователей

Также с помощью функций кластеризации и ассоциации ПМ ФИО были реализованы задачи поиска похожих товаров (блок «Похожие товары») и рекомендация товаров (блок «С этим товаром покупают»). Для этого в компонент Карточка товара были подключены компоненты модуля ПМ ФИО и применены его функции. Результат подключения компонентов на странице товара продемонстрирован на рисунке 7.

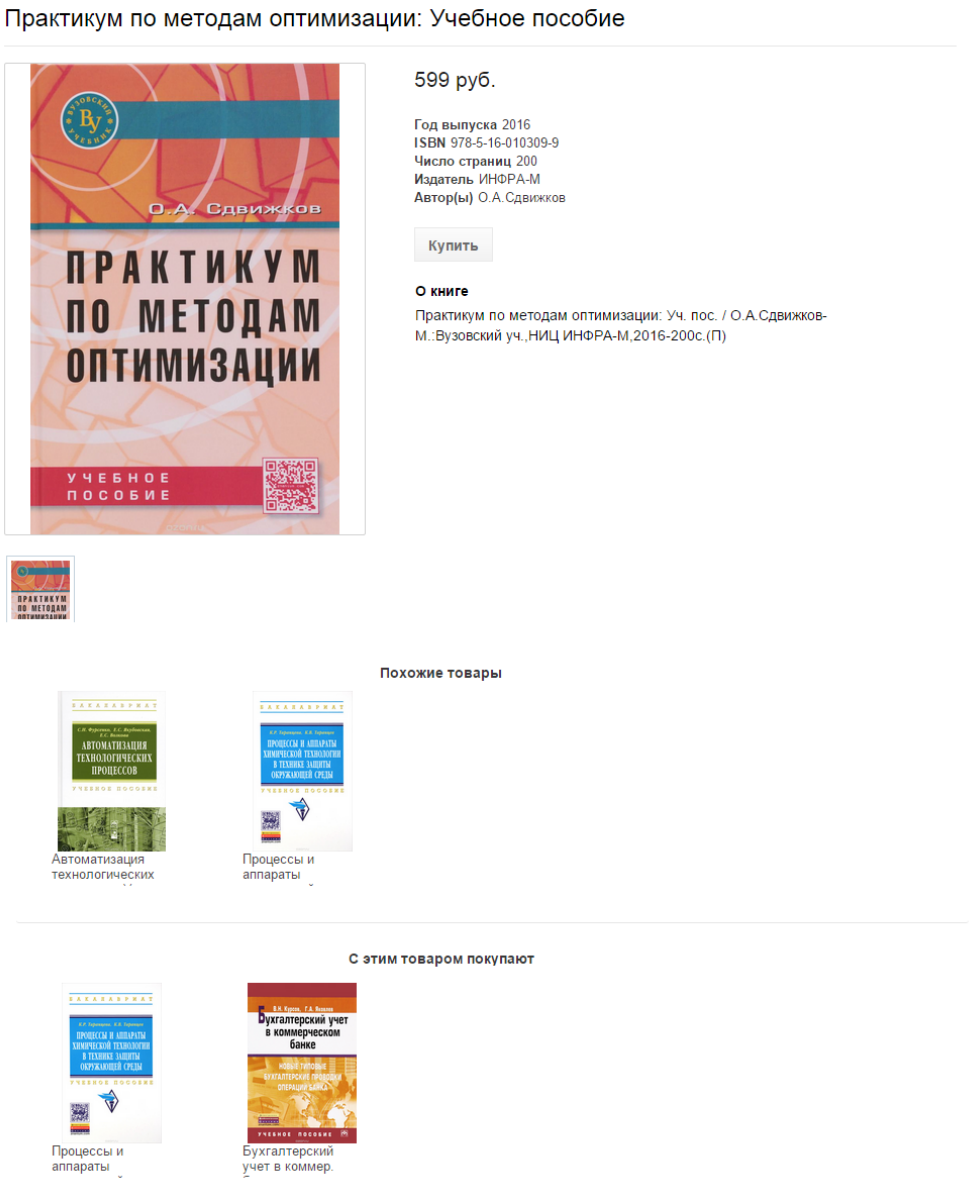

Рисунок 7 – Экранная форма страницы компонента товара

Структура системы 1С-Битрикс позволяет изменить разработчику внешнее представление компонента (с помощью файла template.php в шаблонах компонента) и создать для него визуальное представление, соответствующее остальному шаблону сайта.

Выводы

В конструкторском разделе были выполнены следующие задачи:

- рассмотрены программные технологии реализации ПМ ФИО;
- рассмотрены системы управления содержимым и выбрана в качестве основной система 1С-Битрикс;
- проведен обзор языков программирования и выбран в качестве основного PHP 5.4;
- проведен обзор сред разработки и выбрана в качестве основной NetBeans IDE;
- разработан пользовательский интерфейс ПМ ФИО.

3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

В данном разделе описаны технологии программирования и отладки, архитектура взаимодействия ПМ ФИО с системой 1С-Битрикс, проведен обзор методов тестирования и описан процесс тестирования ПМ ФИО.

3.1. Технологии веб-программирования

3.1.1. Язык разметки HTML

HTML – стандартизированный язык разметки документов в интернете. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства. [40]

Язык HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. [41]

Язык XHTML является более строгим вариантом HTML, он следует всем ограничениям XML и, фактически, XHTML можно воспринимать как приложение языка XML к области разметки гипертекста. [40]

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования. [41]

3.1.2. Каскадные таблицы стилей

CSS (или каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. [42]

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL. [40]

3.1.3. JavaScript

JavaScript – прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262). [38]

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. [46]

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке. [38]

3.1.4. Технология AJAX

Ajax – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. [21]

AJAX – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах: [25]

- использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например с использованием XMLHttpRequest (основной объект):
  - через динамическое создание дочерних фреймов;
  - через динамическое создание тега <script>;
  - через динамическое создание тега <img> (как это реализовано в Google Analytics);
- использование DHTML для динамического изменения содержания страницы;

Действия с интерфейсом преобразуются в операции с элементами DOM (англ. Document Object Model), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш. Каскадные таблицы стилей, или CSS (англ. Cascading Style Sheets), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных. [21]

Три из этих четырёх технологий – CSS, DOM и JavaScript – составляют DHTML. По мнению некоторых специалистов, средства DHTML, появившиеся в 1997 году, подавали большие надежды, но так и не оправдали их.

В качестве формата передачи данных могут использоваться фрагменты простого текста, HTML-кода, JSON или XML. [25]

3.1.4. Библиотека jQuery

jQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. [23]

jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл:

```
<head>
   <script src="jquery-2.2.2.min.js">
   </script>
</head>
```

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — jQuery. [39]

Получение jQuery-объекта происходит с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий селектора и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает

ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса. [23]

Вызов глобальных методов у объекта $ производится, например, для удобных прохождений итератором по массиву. Также, `$.ajax`, `$.post` и другие соответствующие функции позволяют использовать методы AJAX. [39] Например:

```
$.ajax({
  type: "POST",
  url: "script.php",
  data: {
    field1: 'value1',
    field2: 'value2'},
  success: function(data){
    console.log(data);
  }
});
```

3.1.4. Серверный язык PHP

В области веб-программирования, в частности серверной части, PHP – один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET).

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств для разработки веб-приложений. Основные из них: [13]

- автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
- взаимодействие с большим количеством различных систем управления базами данных (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Интерфейс PDO);
- автоматизированная отправка HTTP-заголовков;
- работа с HTTP-авторизацией;
- работа с cookies и сессиями;
- работа с локальными и удалёнными файлами, сокетами;

- обработка файлов, загружаемых на сервер;
- работа с XForms.

В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемся на данных поисковых систем, в мае 2016 года PHP находился на 6 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др. [18]

Входит в LAMP – распространённый набор программного обеспечения для создания и хостинга веб-сайтов (Linux, Apache, MySQL, PHP).

3.1.4. Типы данных PHP

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных. Преобразования между скалярными типами зачастую осуществляются неявно без дополнительных усилий (впрочем, PHP предоставляет широкие возможности и для явного преобразования типов). [19]

К скалярным типам данных относятся: [18]

- целый тип (integer);
- вещественный тип данных (float, double);
- логический тип (boolean);
- строковый тип (string);
- специальный тип NULL.

К нескалярным типам относятся: [18]

- ресурс (resource);
- массив (array);
- объект (object).

К псевдотипам относятся:

- mixed – любой тип;
- number – число (integer либо float);
- callback – string или анонимная функция;
- void отсутствие параметров.

Псевдотипы обычно используются в документации и описаниях функций. [19]

3.1.5. PHP и ООП

Класс в PHP объявляется с помощью ключевого слова class. Методы и поля класса могут быть общедоступными (по умолчанию public), защищёнными (protected) и скрытыми (private). PHP поддерживает все три основных механизма ООП – инкапсуляцию, полиморфизм подтипов и наследование (родительский класс указывается с помощью ключевого слова extends после имени класса). Поддерживаются интерфейсы (ставятся в соответствие с помощью implements). [22]

Разрешается объявление финальных, абстрактных методов и классов. Множественное наследование классов не поддерживается, однако класс может реализовывать несколько интерфейсов. Для обращения к методам родительского класса используется ключевое слово parent. [13]

Начиная с версии 5.4.0 множественное наследование может быть реализовано с помощью механизма особенностей (trait). Особенности похожи на примеси (mixins), за исключением того, что для них нельзя напрямую создать экземпляр. Повторное использование кода заключено в использовании кода особенности в нескольких классах. Допускается использовать в одном классе нескольких особенностей. Механизм особенностей имеет средства разрешения конфликтов имён. При запуске программы код особенности будет скомпилирован в код содержащего его класса.[22]

Классы в PHP имеют ряд «магических» методов (magic methods), начинающихся с двух символов подчёркивания. Особо стоит отметить конструктор (__construct(), в версиях до 5.0 конструктором служил метод, одноимённый с классом) и деструктор (__destruct()), а также методы чтения (__get()) и записи (__set()), свёртывания (__sleep()) и развёртывания (__wakeup()), клонирования (__clone()) и др. Эти методы являются достаточно гибким инструментом: переопределяя их, можно добиться существенного изменения поведения объекта.[22]

Все функции-члены реализованы как виртуальные и потому все они являются методами.[13]

Экземпляры класса создаются с помощью ключевого слова new, обращение к полям и методам объекта производится с использованием оператора ->. Для доступа к членам класса из его методов используется переменная $this. [22]

3.2. Особенности программирования для системы 1С-Битрикс

3.2.1. Bitrix Framework

Bitrix Framework – это созданная на основе PHP платформа для разработки веб-приложений. На этой платформе компанией «1С-Битрикс» созданы два популярных продукта: «1С-Битрикс: Управление сайтом» и «1С-Битрикс: Корпоративный портал». В базовой поставке идёт большой набор компонентов, и именно он обеспечивает быстрое развёртывание и внедрение проектов. [9]

3.2.2. Модули

Bitrix Framework имеет модульную структуру. Каждый модуль отвечает за управление определенными элементами и параметрами сайта: информационным наполнением и структурой сайта, форумами, рекламой, рассылкой, распределением прав между группами пользователей, сбором статистики посещений, оценкой эффективности рекламных кампаний и т. д. [2]

Модуль – это модель данных и API для доступа к этим данным. Статические методы классов модуля могут вызываться в компонентах, шаблонах, других модулях. Также внутри контекста Bitrix Framework могут создаваться экземпляры классов. Модули системы, главным образом, работают независимо друг от друга. Однако в целом ряде случаев функционал одних модулей основан на возможностях других. [35]

3.1.3. Информационные блоки

Информационные блоки представляют собой очередной уровень абстракции над обычными таблицами СУБД, своеобразная "база данных в базе данных". Поэтому к ним частично применимы все те правила, которых придерживаются при проектировании БД. [2]

Инфоблоки – сущность, которая в физической структуре БД создает 4 таблицы, не меняющиеся при изменении структуры данных: типы объектов, экземпляры объектов, свойства объектов и значения свойств объектов. [3]

Плюсы такого подхода: [9]

- удобный контроль над данными такой структуры из своего приложения,
- универсальность методов,
- общая структура данных для любого проекта,
- возможность многократно менять типы данных для полей без уничтожения самих данных.

Минусы такого подхода: [9]

- повышенные требования к производительности,
- непрозрачность при прямом доступе к данным.

3.2.4. Работа с инфоблоками через API

Потребности заказчиков сайтов очень разнообразны. Штатный функционал Bitrix Framework не может решать всех задач, которые могут быть поставлены перед разработчиком при создании интернет-проектов. Для реализации нестандартных задач необходимо использовать API [35]. Они чаще всего используются при программировании.

API модуля состоит из нескольких высокоуровневых функций для выборки данных в публичном разделе сайта и набора классов с низкоуровневыми методами для более специализированной работы. [9]

Перед использованием модуля необходимо проверить, установлен ли он, и подключить его при помощи конструкции:

```
<?
if(CModule::IncludeModule("iblock"))
{
   //здесь можно использовать функции и классы модуля
}
?>
```

Для получения данных при показе в публичном разделе сайта можно пользоваться функциями с простыми параметрами и предустановленными фильтрами. Эти функции выбирают по умолчанию те значения, которые подходят для места выборки, а именно только активные, привязанные к текущему сайту, подходящие по правам доступа и т.п.

Вся работа с датами через API (вставка, выборка, фильтры и т.п.) производится в формате текущего сайта или, если в административной части, в формате текущего языка.

Ряд функций API доступен всегда, т.е. описан в главном модуле, а ряд функций зависит от используемого модуля, и может присутствовать или отсутствовать в различных редакциях продукта. Например, функции для работы с социальной сетью присутствуют в редакциях «1С-Битрикс: Управление сайтом – Бизнес» и выше, а также в «1С-Битрикс: Корпоративный портал». [9]

Для большинства классов Bitrix Framework доступны функции [2]:

- Выборка данных (GetList):

```
CIBlockResult
CIBlockElement::GetList(
 array arOrder = Array("SORT"=>"ASC"),
 array arFilter = Array(),
 mixed arGroupBy = false,
 mixed arNavStartParams = false,
 array arSelectFields = Array()
);
```

Возвращает список элементов по фильтру arFilter. Нестатический метод.

- Занесение нового элемента (Add):

```
int CIBlockElement::Add(
 array arFields,
 bool bWorkFlow = false,
 bool bUpdateSearch = true,
 bool bResizePictures = false
);
```

Метод добавляет новый элемент информационного блока. Перед добавлением элемента вызываются обработчики события OnBeforeIBlockElementAdd, из которых можно изменить значения полей или отменить добавление элемента вернув сообщение об ошибке. После добавления элемента вызывается событие OnAfterIBlockElementAdd. Нестатический метод.

- Обновление и удаление элемента (Update):

```
bool
 CIBlockElement::Update(
 int ID,
```

```
 array arFields,
 bool bWorkFlow = false,
 bool bUpdateSearch = true,
 bool bResizePictures = false,
 bool bCheckDiskQuota = true
);
```

Метод изменяет параметры элемента с кодом ID. Перед изменением элемента вызываются обработчики события OnStartIBlockElementUpdate из которых можно изменить значения полей или отменить изменение элемента вернув сообщение об ошибке. После изменения элемента вызывается само событие OnAfterIBlockElementUpdate. Нестатический метод.

- Удаление элемента (Delete):

```
bool CIBlockElement::Delete(
 int ID
);
```

Метод удаляет элемент информационного блока. Также удаляются значения свойств типа "Привязка к элементу" указывающие на удаляемый. При установленном модуле поиска элемент удаляется из поискового индекса. Перед удалением вызываются обработчики события OnBeforeIBlockElementDelete из которых можно отменить это действие. После удаления вызывается обработчик события OnAfterIBlockElementDelete. Метод статический.

Для большинства модулей предлагается специализированная структура классов, механизм событий и дополнительные функции. В частности, для модуля Информационные блоки приводится описание: [9]

- Всех используемых таблиц в базе данных, в том числе полей таблиц.
- Классов для работы с типами инфоблоков, инфоблоками, элементами, разделами, полями.
- Событий, происходящих при добавлении, изменении и удалении объектов модуля.
- Функций, расширяющих возможности ядра.
- Способов создать пользовательские формы редактирования и свои типы данных.

3.3. Отладка программного модуля

Для успешной отладки приложений PHP в IDE NetBeans для PHP требуется механизм PHP, локальный веб-сервер Apache и отладчик XDebug, установленный и настроенный для разработки PHP. [30]

Когда исполнение программы установлено на паузу, XDebug может извлечь информацию о текущем состоянии программы, такую, как значения переменных программы. Фактически это означает следующую последовательность выполняемых действий [30]:

1. Установить точку останова в каждой строке, на которой исполнение исходного кода PHP должно приостановиться.
2. Начать сеанс отладки.
3. Когда достигнута строка с точкой останова, исполнять сценарий по одной строке, нажимая F7 и F8. Отслеживать состояние приложения нужно в окнах отладчика.
4. Закрыть сеанс отладки.

IDE NetBeans обеспечивает панель инструментов отладки, которая используется для пошагового перехода между файлами (рис. 7).

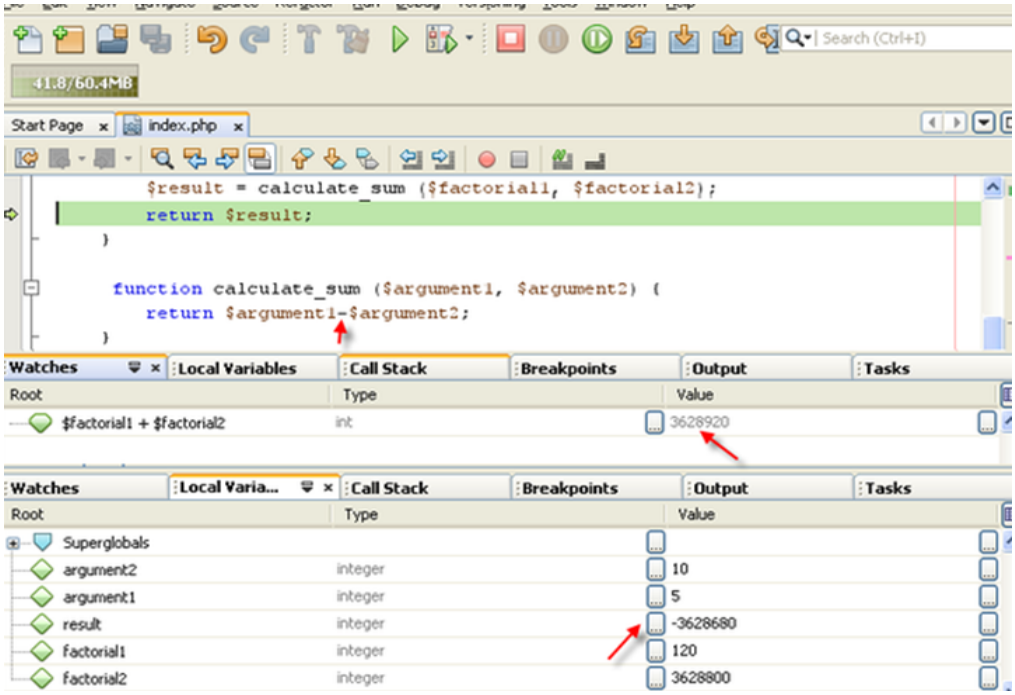

Рисунок 7 – Отладка в NetBeans с помощью XDebug

3.4. Тестирование методами «черного» и «белого» ящика

Разность этих подходов заключается в наличии доступа к исходному коду тестируемого ПО. При использовании «черного ящика» тестировщик использует только внешние рычаги взаимодействия с программой: с помощью пользовательского интерфейса или подключившись к тестируемой системе [4]. При работе с «белым ящиком» тестировщик имеет доступ к коду, тем самым тестируя внутреннюю структуру программы.

Тестирование чёрного ящика или поведенческое тестирование – стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Используя этот метод, не нужно знать внутреннее устройство программы («черный ящик»). Объектами тестирования в этом случае являются потоки входных и выходных данных. Это позволяет определять «правильность» работы ПО в соответствии с функциональными требованиями к продукту. Таким образом, критериями тестирования черным ящиком являются [4]:

- тестирование функций программы;
- тестирование потока входных данных;
- тестирование потока выходных данных;
- тестирование области допустимых значений;
- тестирование длины набора данных;
- тестирование порядка входных данных.

Тестирование по стратегии белого ящика, также называемое техникой тестирования, управляемой логикой программы, позволяет проверить внутреннюю структуру программы. Исходя из этой стратегии, тестирующий получает тестовые данные путём анализа логики работы программы. Техника Белого ящика включает в себя следующие методы тестирования [14]:

- покрытие решений
- покрытие условий
- покрытие решений и условий
- комбинаторное покрытие условий

В результате исследования в качестве основного метода тестирования ПМ ФИО выбрано тестирование методом «белого ящика», поскольку оно позволяет покрыть тест-кейсами функции модуля ПМ ФИО и проверить правильность выполнения кластеризации

и ассоциации. При использовании данного метода тестировщик имеет доступ к исходному коду программы.

3.5. Классификация по объекту тестирования

По объекту тестирования различают:

- функциональное тестирование;
- тестирование производительности;
- тестирование безопасности;
- тестирование интерфейса пользователя.

Производительность ПМ ФИО следует тестировать вместе с системой «1С-Битрикс», для чего в системе присутствуют специальные инструменты. Безопасность также обеспечивает система контроля содержимого. Тестирование пользовательского интерфейса в данном случае не имеет смысла, поскольку интерфейс компонентов модуля внедряется вместе с разработкой интерфейса всего веб-сайта. Поэтому для тестирования ПМ ФИО было выбрано функциональное тестирование.

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО и какие задачи оно решает. [14]

Функциональные требования включают в себя:

- функциональную пригодность;
- точность;
- способность к взаимодействию;
- соответствие стандартам и правилам;
- защищённость.

3.6. Классификация по степени изолированности компонентов

По степени изолированности компонентов различают:

- Модульное тестирование. Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности (модули программ, объекты,

классы, функции и т.д.). Один из наиболее эффективных подходов модульного тестирования – это подготовка автоматизированных тестов до начала основного кодирования (разработки) программного обеспечения. Разработка ведется до тех пор пока все тесты не будут успешно пройдены. [4]

- Интеграционное тестирование – это фаза тестирования ПО, на которой отдельные программные модули комбинируются и тестируются в группе. Основной целью интеграционного тестирования является подтверждение того, что результаты взаимосвязи между двумя и более компонентами отвечают функциональным требованиям. [14]
- Системное тестирование. Системное тестирование предназначено для тестирования готового ПО в том состоянии, в котором оно будет внедряться в опытно-промышленную эксплуатацию. При этом выявляются дефекты, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность. [4]

Поскольку необходимо было проверить работу функций кластеризации и ассоциации ПМ ФИО по отдельности, применялось модульное тестирование и были разработаны тест-кейсы для проверки работоспособности данных функций.

3.7. Тестирование реализации алгоритма с-средних методом «белого ящика»

Протестируем алгоритм кластеризации при помощи метода «белого ящика». Алгоритм тестирования изображен на рисунке 7.

Рисунок 7 – Алгоритм тестирования реализации метода с-средних

Возьмем следующий тестовый набор данных (в [26] данный набор называется Ирис Фишера):

1. Набор данных состоит из 150 записей, в каждой из которых 5 полей;

2. Для кластеризации используются первые 4 поля записей, 5 поле – номер кластера, в которую входит запись, не используется при кластеризации и предназначено для проверки результата.

Результат кластеризации отображен на рисунке 8.

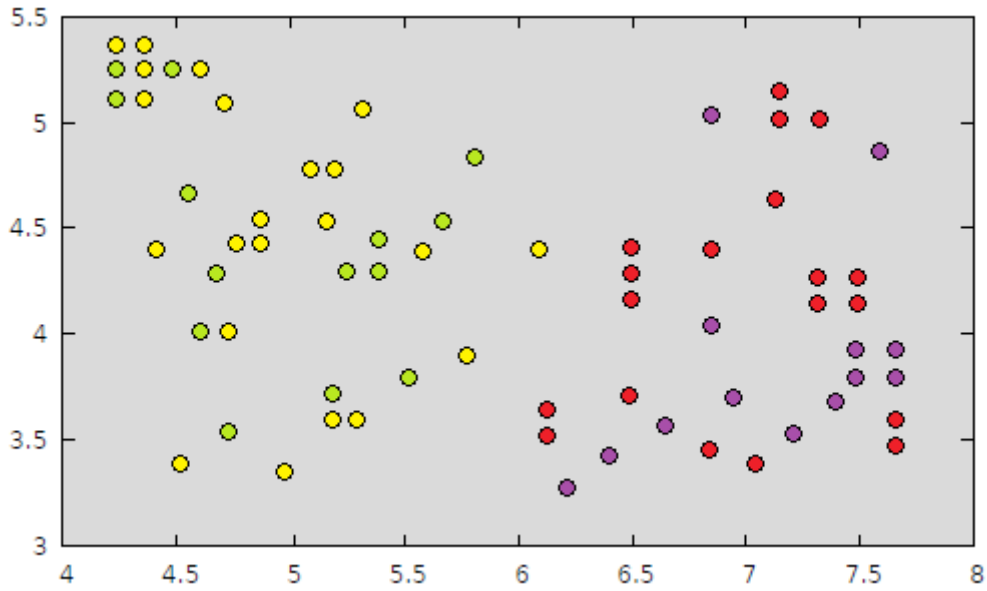

Рисунок 8 – Результаты кластеризации

Точки с одним цветом принадлежат одному кластеру. Значение целевой функции сведено к минимуму (J = 0), т.е. результат совпадает с эталоном.

3.8. Тестирование реализации алгоритма Apriori методом «белого ящика»

Протестируем алгоритм поиска ассоциативных правил при помощи метода «белого ящика». Алгоритм тестирования реализации Apriori изображен на рисунке 9.

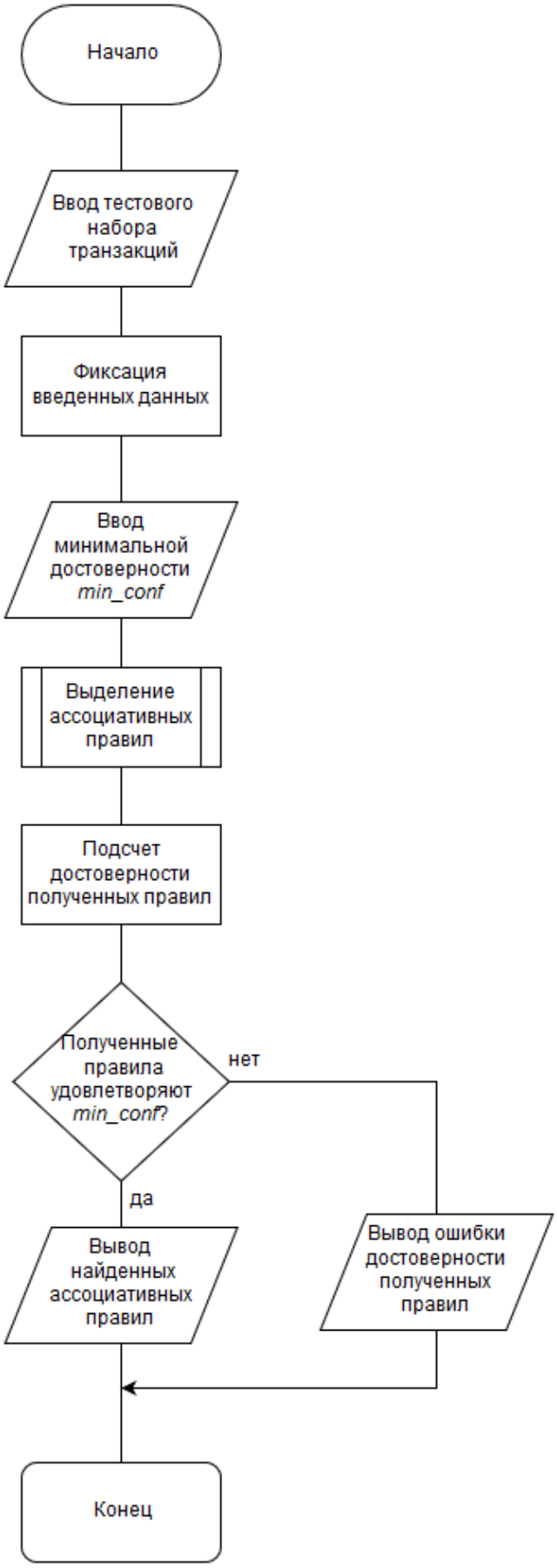

Рисунок 9 – Алгоритм тестирования реализации метода Apriori

Возьмем следующий тестовый набор данных:

1. Дан набор транзакций (1,2,3,4 – товары):

{1,2,3,4},

{1,2,4},

{1,2},

{2,3,4},

{2,3},

{3,4},

{2,4}

2. Выбраны параметры ассоциации: минимальная поддержка – 2, минимальная достоверность – 60%.

3. Получить достоверность следующих правил:

1 => 2,4 – 66,666% – достоверен

2 => 3,4 – 33,333% – нет

1 => 2 – 100% – достоверен

2 => 3 – 50% – нет.

Полученные результаты:

1 => 2,4 – 66,666%

2 => 3,4 – 33,333%

1 => 2 – 100%

2 => 3 – 50%

Среди этих правил достоверными являются:

1 => 2,4

1 => 2.

В результате получаем , что при покупке 1 товара с вероятностью 66,6% купят 2 и 4 товары и с 100% – 2 товар.

Таким образом, результат совпадает с эталоном и удовлетворяет условию минимальной достоверности.

Выводы

В технологическом разделе были выполнены следующие задачи:

- описаны технологии программирования;
- рассмотрены применявшиеся методы отладки;
- проведен анализ методов тестирования и выбран наиболее подходящий для тестирования ПМ ФИО;
- разработаны тест-кейсы для тестирования ПМ ФИО с использованием метода «белого ящика».

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке программного модуля интеллектуальной обработки данных для системы 1С-Битрикс.

Пояснительная записка состоит из исследовательского, конструкторского и технологического разделов и приложений 1 и 2.

В исследовательском разделе были исследованы предметная область и потребности потенциальных пользователей, рассмотрены теоретические основы кластеризации и поиска ассоциативных правил, проведен обзор существующих решений, разработаны схема данных и схема алгоритма работы ПМ ФИО.

В конструкторском разделе рассмотрены программные технологии реализации ПМ ФИО, выбраны язык программирования и среда разработки, разработан пользовательский интерфейс ПМ ФИО.

В технологическом разделе описаны технологии программирования, применявшиеся методы отладки, проведен анализ методов тестирования и выбран наиболее подходящий для тестирования ПМ ФИО, разработаны тест-кейсы для тестирования ПМ ФИО с использованием метода «белого ящика».

В приложении 1 приведен программный код ПМ ФИО.

В приложении 2 приведено руководство программиста.

СПИСОК ЛИТЕРАТУРЫ

1. Гагарина Л.Г., Касимов Р.А., Коваленко Д.Г., Федотова Е.Л, Чжо Зо Е, Черников Б.В. Методические указания по подготовке выпускной квалификационной работы по направлению подготовки бакалавров 09.03.04 «Программная инженерия»/ Под редакцией Б.В. Черникова; М., МИЭТ, 2016 г., 20 с.
2. Басыров Р. И. 1С-Битрикс. Постройте профессиональный сайт сами! – СПб.: Питер, 2008. – С. 304.
3. Басыров Р. И. 1С-Битрикс: Корпоративный портал. Повышение эффективности компании. – СПб.: Питер, 2010. – С. 320.
4. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. – Питер, 2004. – 320 с.
5. Бернерс-Ли Т. Architecture of the World Wide Web, Volume One (W3C), 2004, – С.220.
6. Воронцов К.В. Алгоритмы кластеризации и многомерного шкалирования. Курс лекций. МГУ, 2007, – С.109-119.
7. Введение в анализ ассоциативных правил [Электронный ресурс], – BaseGroup Labs, URL: https://basegroup.ru/community/articles/intro – (Дата обращения: 19.02.2016).
8. Граничина Н.О., Шалымов Д.С. Рандомизированный алгоритм устойчивой кластеризации. СПбГУ, 2009, – С. 55-67.
9. Документация системы 1С-Битрикс [Электронный ресурс] – 1С-Bitrix, URL: https://dev.1c-bitrix.ru/learning/course/ – (Дата обращения: 19.02.2016).
10. Дэн Рамел. Joomla! для профессионалов – М.: «Вильямс», 2014. – 448 с.
11. Дюк В. Data mining - интеллектуальный анализ данных, 1996, – 400 с.
12. Жамбю М. Иерархический кластер-анализ и соответствия. – М.: Финансы и статистика, 1988. – 345 с.
13. Зандстра М. PHP: объекты, шаблоны и методики программирования, 3-е издание. – М.: «Вильямс», 2010. – С. 560.
14. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование. – М.: «Вильямс», 2002. – 374 с.
15. Коваленко О. С. Обзор проблем и перспектив анализа данных // Информатика, вычислительная техника и инженерное образование. ТТИ ЮФУ, 2011 – №5 (7), – С. 13-17.

16. Колдаев, В.Д. Основы алгоритмизации и программирования: учебное пособие / под ред. проф. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2012. – 416 с.
17. Колисниченко Д. Joomla! 3.0. Руководство пользователя. – М.: «Диалектика», 2013. – 256 с.
18. Костарев А. Ф. PHP 5. – Спб.: «БХВ-Петербург», 2008. – С. 1104.
19. Котеров Д., Костарев А. PHP. – Спб.: «БХВ-Петербург», 2005. – С. 1120.
20. Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python. – Перевод с английского. – М.: ДМК Пресс, 2015, – С. 95.
21. Крейн Д., Бибо Б. Ajax на практике. – М.: Вильямс, 2007, – С. 84-91.
22. Кузнецов М., Симдянов И. Объектно-ориентированное программирование на PHP. – Спб.: «БХВ-Петербург», 2007.
23. Ленгсторф Дж. PHP и jQuery для профессионалов. – М.: «Вильямс», 2010. – С. 352.
24. Маккинли У. Python и анализ данных. – Перевод с английского. – М.: ДМК Пресс, 2015. – 482 с.
25. Маклафлин Б. Изучаем Ajax. – СПб.: Питер, 2007, – С. 43-45.
26. Максимова К.И. Задача поиска ассоциативных правил в статистике покупок продуктов, СФУ ИМиФИ, 2009, – С. 77.
27. Мандель И. Д. Кластерный анализ. – М.: Финансы и Статистика, 1988, – С. 32.
28. Меркулов Ю. Путеводитель по текстовым редакторам [Электронный ресурс], – URL: http://www.ixbt.com/soft/texteditors-4.shtml – (Дата обращения: 26.02.2016).
29. Монахов Вадим. Язык программирования Java и среда NetBeans. – 3-е издание. – СПб.: «БХВ-Петербург», 2011. – С. 704.
30. Отладка исходного кода PHP в IDE NetBeans [Электронный ресурс], – NetBeans, – URL: https://netbeans.org/kb/docs/php/debugging_ru.html – (Дата обращения: 26.02.2016).
31. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети. – Винница: УНІВЕРСУМ–Вінниця, 1999. – 320 с.
32. Ротштейн А.П., Кательников Д.И. Идентификация нелинейных зависимостей нечеткими базами знаний // Кибернетика и системный анализ. – 1998. – №5. – С. 53–61.

33. Сервис рекомендаций RetailRocket [Электронный ресурс] – RetailRocket, URL: https://retailrocket.ru/ – (Дата обращения: 19.02.2016).
34. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. – М.: БИНОМ, 2008. – 368 с.
35. Создание собственных модулей [Электронный ресурс] – Bitrix Framework, URL: http://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=43&CHAPTER_ID=1014 – (Дата обращения: 15.04.2016).
36. Суэринг С, Конверс Т., Парк Дж. PHP и MySQL. Библия программиста, 2-е издание. – М.: «Диалектика», 2010. — С. 912.
37. Флэнаган Д., Мацумото Ю. Язык программирования Ruby. – 1-е изд. – СПб.: Питер, 2011. – 496 с.
38. Флэнаган Д. JavaScript: Подробное руководство, 2008, – с. 982.
39. Фримен А. jQuery для профессионалов – М.: «Вильямс», 2012. – 960 с.
40. Фримен Э. Изучаем HTML, XHTML и CSS, «Питер», 2010, – 656 с.
41. Шафер С. HTML, XHTML и CSS. Библия пользователя, 5-е издание, – «Диалектика», 2010, – 656 с.
42. Шмитт К. CSS. Рецепты программирования. – СПб.: БХВ-Петербург, 2007. – 592 с.
43. Штовба С.Д. Введение в теорию нечетких множеств и нечеткой логики, 2009, – 98 с.
44. Binder B. Testing Object-oriented Systems, 2000, – с. 320.
45. Boudreau T., Glick J. NetBeans: The Definitive Guide. O'Reilly Media, 2003, – с. 672.
46. Burns J., Growney A. JavaScript Goodies, 2000, Pearson Education, – с. 372.
47. Fisher, R.A. «The Use of Multiple Measurements in Taxonomic Problems». 179–188.
48. PHPStorm [Электронный ресурс], – PHPStorm IDE JetBrains, – URL: https://www.jetbrains.com/phpstorm/ – (Дата обращения: 26.02.2016).
49. R. Agrawal, R. Srikant. "Fast Discovery of Association Rules", In Proc. of the 20th International Conference on VLDB, Santiago, Chile, September 1994.
50. R. Agrawal, T. Imielinski, A. Swami. 1993. Mining Associations between Sets of Items in Massive Databases. In Proc. of the 1993 ACM-SIGMOD Int’l Conf. on Management of Data, 207-216.
51. Savasere A., Navathe S. An Efficient Algorithm for Mining Association Rules in Large Databases, San Francisco, 1995, 432-444

52. Scott, Adam D. WordPress for Education. – Birmingham: Packt Publishing Ltd, 2012. – 144 c.
53. Sublime Text 3 [Электронный ресурс], – Sublime Text 3, – URL: http://www.sublimetext.com/ – (Дата обращения: 26.02.2016).
54. Williams, Laurie. "White-Box Testing", 2004, – C. 69.

Приложение 1

Программный модуль для реализации функции
интеллектуальной обработки данных для системы 1С-Битрикс
Программный код

ПМ ФИО

ОГЛАВЛЕНИЕ

1. Файлы установки и настройки параметров ПМ ФИО

1.1. Файл include.php

```
<?php

define('MAX_FCM_ITERNUM', 10000);

CModule::includeModule('sdmm_module');

require_once('classes/general/fuzzycmeanspoint.php');
require_once('classes/general/fuzzycmeans.php');
require_once('classes/general/candidate.php');
require_once('classes/general/assocrule.php');
require_once('classes/general/apriori.php');

CModule::AddAutoloadClasses(
    'sdmm_module',
    [
        'FuzzyCMeansPoint' => 'classes/general/fuzzycmeanspoint.php',
        'FuzzyCMeans' => 'classes/general/fuzzycmeans.php',
        'Candidate' => 'classes/general/candidate.php',
        'AssocRule' => 'classes/general/assocrule.php',
        'AprioriAlgorithm' => 'classes/general/apriori.php',
    ]
);
```

1.2. Файл install/index.php

```
<?

/**
 * Класс установки модуля ПМ ФИО
 *
 *
 */
class sdmm_module extends CModule
{
    var $MODULE_ID = "sdmm_module";
    var $MODULE_VERSION;
    var $MODULE_VERSION_DATE;
    var $MODULE_NAME;
    var $MODULE_DESCRIPTION;
    var $MODULE_CSS;

    /**
     * Конструктор класса sdmm_module
     *
     *
     */
    function sdmm_module() {
        $this->MODULE_VERSION = '1.0';
        $this->MODULE_VERSION_DATE = '2016-04-07 15:00:00';
        $this->MODULE_NAME = 'sdmm_module - модуль Data Mining';
        $this->MODULE_DESCRIPTION = 'Модуль для решения задач Data Mining.';
    }
```

```
    /**
     * Функция установки модуля sdmm_module
     *
     *
     */
    function DoInstall() {
        RegisterModule('sdmm_module');
        return true;
    }

    /**
     * Функция деинсталляции модуля sdmm_module
     *
     *
     */
    function DoUninstall() {
        UnregisterModule('sdmm_module');
    }
}

?>
```

1.3. Файл install/version.php

```
<?php

$arModuleVersion = array
(
    "VERSION"      => "1.0",
    "VERSION_DATE" => "2016-04-07 15:00:00"
);

?>
```

2. Классы модуля ПМ ФИО

2.1. Классы реализации алгоритма Apriori

2.1.1. Класс Apriori

```
<?php

/**
 * Класс, реализующий алгоритм Apriori
 * поиска ассоциативных правил
 *
 */
class AprioriAlgorithm
{
    private $input_data; // входные данные в нормализованном виде

    /**
     * Функция, пересчитывающая поддержку кандидатов
     *
     * @param Candidate
     * @return int
     */
    private function calcSupport(Candidate $candidate) {
        if ($candidate->getSupport() > 0)
            return $candidate->getSupport();

        $items = $candidate->getItems();
        $min_count_item = $this->calcMinCountItem($items);
        if (!$min_count_item || $min_count_item <= 0)
            return false;

        $support = 0;
        $transactions = $this->input_data[$min_count_item];
        foreach ($transactions as $id => $value) {
            $supp_flag = true;
            foreach ($items as $item) {
                if (!isset($this->input_data[$item][$id])
                        || !$this->input_data[$item][$id]) {
                    $supp_flag = false;
                    break;
                }
            }
            $support += $supp_flag ? 1 : 0;
        }

        return $support;
    }

    /**
     * Нахождение минимального по количеству элементов кандидата
     *
     * @param array
```

```
     * @return int
     */
    private function calcMinCountItem(array $items) {
        $min_count = -1;
        $min_count_item = -1;
        foreach ($items as $item) {
            $new_count = count($this->input_data[$item]);
            if ($min_count == -1 || $new_count < $min_count) {
                $min_count = $new_count;
                $min_count_item = $item;
            }
        }

        return $min_count_item;
    }

    /**
     * Генерация k-элментных кандидатов
     * путем объединения (k-1)-элементных кандидатов
     *
     * @param array
     * @return array
     */
    private function AprioriGen(array $freqSets) {
        $Ck = [];
        $n = count($freqSets);
        for ($i = 0; $i < $n; $i++) {
            for ($j = $i + 1; $j < $n; $j++) {
                $merge = $freqSets[$i]->mergeWithCandidate($freqSets[$j]);
                if (!empty($merge))
                    $Ck[] = $merge;
            }
        }

        return $Ck;
    }

    /**
     * Удаление избыточных правил
     *
     * @param array, array
     * @return void
     */
    private function removeOddCandidates(array &$Ck, array $freqSets) {
        $indices = [];
        foreach ($Ck as $index => $cand) {
            $subsets = $cand->getNearestSubsets();
            $remove_flag = false;
            foreach ($subsets as $subset) {
                if (!Candidate::findInFreqSets($subset, $freqSets)) {
                    $remove_flag = true;
                }
            }
            if ($remove_flag)
                $indices[] = $index;
        }

        foreach ($indices as $index)
```

```
            unset($Ck[$index]);
        $Ck = array_values($Ck);
    }

    /**
     * Получение часто встречающихся наборов (кандидатов)
     *
     * @param int
     * @return array
     */
    private function getFreqSets($min_support) {
        $Ck = [];
        foreach ($this->input_data as $item => $value) {
            $Ck[] = new Candidate([$item]);
        }

        $freqSets = [];
        $i = 1;
        while (!empty($Ck)) {
            $freqSets[$i] = [];
            foreach ($Ck as &$cand) {
                $cand->setSupport($this->calcSupport($cand));
                if ($cand->getSupport() >= $min_support)
                    $freqSets[$i][] = $cand;
            }
            unset($Ck);
            $Ck = $this->AprioriGen($freqSets[$i]);
            $this->removeOddCandidates($Ck, $freqSets[$i]);
            $i++;
        }

        return $freqSets;
    }

    /**
     * Конструктор класса, преобразует входные данные
     * в нормализованный вид
     *
     * @param array
     */
    public function __construct(array $transactions) {
        $this->input_data = [];
        foreach ($transactions as $id => $items) {
            foreach ($items as $item) {
                $this->input_data[$item][$id] = true;
            }
        }
    }

    /**
     * Запуск алгоритма, нахождение ассоциативных правил
     *
     * @param int, float
     * @return array
     */
    public function run($min_support, $min_conf) {
        $freqSets = $this->getFreqSets($min_support);
        $assocRules = [];
```

```
        foreach ($freqSets as $level => $sets) {
            if ($level == 1)
                continue;
            foreach ($sets as $freqSet) {
                $assocRules = array_merge($assocRules,
                        $freqSet->getAssocRules($freqSets));
            }
        }
        $indices = [];
        foreach ($assocRules as $index => $rule) {
            if ($rule->getConfidence() < $min_conf) {
                $indices[] = $index;
            }
        }

        foreach ($indices as $index)
            unset($assocRules[$index]);
        $assocRules = array_values($assocRules);

        return $assocRules;
    }
}
```

2.1.2. Класс ассоциативных правил AssocRule

```
<?php

/**
 * Класс ассоциативных правил
 *
 */
class AssocRule
{
    private $condition;  // условие (левая часть правила)
    private $consequent; // заключение (правая часть правила)
    private $confidence; // достоверность правила

    /**
     * Конструктор класса
     *
     * @param Candidate, Candidate
     */
    public function __construct(Candidate $condition, Candidate $consequent)
{
        $this->condition = $condition;
        $this->consequent = $consequent;
    }

    /**
     * Получение условия правила
     *
     * @return Candidate
     */
    public function getCondition() {
        return $this->condition;
    }
```

```
    /**
     * Получение заключения правила
     *
     * @return Candidate
     */
    public function getConsequent() {
        return $this->consequent;
    }

    /**
     * Установка значения достоверности правила
     *
     * @param float
     * @return void
     */
    public function setConfidence($conf) {
        $this->confidence = $conf;
    }

    /**
     * Получение достоверности правила
     *
     * @return float
     */
    public function getConfidence() {
        return $this->confidence;
    }

    /**
     * Подсчет достоверности сгенерированного правила
     *
     * @param Candidate
     * @return float
     */
    public function calcConfidence(Candidate $cand) {
        $suppF = (float)$cand->getSupport();
        $suppS = (float)$this->condition->getSupport();
        $this->confidence = $suppF / $suppS;

        return $this->confidence;
    }
}
```

2.2. Классы реализации алгоритма c-means

2.2.1. Класс кандидатов Candidate

```
<?php

/**
 * Класс кандидатов (часто встречающихся наборов)
 *
 */
```

```
class Candidate
{
    private $items;   // Массив элементов кандидата
    private $support; // Поддержка кандидата

    /**
     * Вспомогательная функция
     * Получение следующей двоичной последовательности
     *
     * @param array
     * @return array
     */
    private function getNextSubsetEq(array $subsetEq) {
        $n = count($subsetEq);
        $subsetEq[$n-1]++;
        for ($i = $n - 1; $i > 0; $i--) {
            if ($subsetEq[$i] > 1) {
                $subsetEq[$i] = 0;
                $subsetEq[$i-1]++;
            }
        }
        if ($subsetEq[0] > 1)
            return false;

        return  $subsetEq;
    }

    /**
     * Создание ассоциативного правила
     *
     * @param Candidate, Candidate, array
     * @return AssocRule
     */
    private function createAssocRule($cond, $conseq, $freqSets) {
        if (empty($freqSets[count($cond)])
                || empty($freqSets[count($conseq)]))
            return false;

        $freqCond = Candidate::findInFreqSets($cond,
                $freqSets[count($cond)]);
        $freqConseq = Candidate::findInFreqSets($conseq,
                $freqSets[count($conseq)]);
        $assocRule = new AssocRule($freqCond, $freqConseq);
        $assocRule->calcConfidence($this);

        return $assocRule;
    }

    /**
     * Конструктор класса
     *
     * @param array
     */
    public function __construct($items) {
        $this->items = $items;
    }

    /**
```

```
     * Установка значения поддержки кандидата
     *
     * @param int
     * @return void
     */
    public function setSupport($support) {
        $this->support = $support;
    }

    /**
     * Получение элементов кандидата
     *
     * @return array
     */
    public function getItems() {
        return $this->items;
    }

    /**
     * Получение поддержки кандидата
     *
     * @return int
     */
    public function getSupport() {
        return $this->support;
    }

    /**
     * Получение (k-1)-элементных подмножеств кандидата
     *
     * @return array
     */
    public function getNearestSubsets() {
        $subsets = [];
        foreach ($this->items as $i => $item) {
            $subset = $this->items;
            array_splice($subset, $i, 1);
            $subsets[] = $subset;
        }

        return $subsets;
    }

    /**
     * Получение нового кандидата
     * путем объединения элементов с другим кандидатом
     *
     * @param Candidate
     * @return Candidate
     */
    public function mergeWithCandidate(Candidate $b) {
        if (count($this->items) != count($b->items))
            return false;

        $k = count($this->items);
        $merge_flag = true;
        for ($i = 0; $i < $k - 1; $i++) {
            if ($this->items[$i] != $b->items[$i]) {
```

```
                $merge_flag = false;
                break;
            }
        }

        $merge_flag = $merge_flag && $this->items[$k - 1] < $b->items[$k -
1];

        if ($merge_flag) {
            $res = $this->items;
            $res[] = $b->items[$k - 1];
        }

        return $res ? new Candidate($res) : false;
    }

    /**
     * Поиск в списке кандидатов
     *
     * @param Candidate, array
     * @return Candidate
     */
    public static function findInFreqSets($needle, array $freqSets) {
        foreach ($freqSets as $freqSet) {
            $cand = $freqSet->getItems();
            if ($cand == $needle) {
                return $freqSet;
            }
        }

        return false;
    }

    /**
     * Получение ассоциативного правила
     * при известном надмножестве данного кандидата
     *
     * @param array
     * @return AssocRule
     */
    public function getAssocRules(array $freqSets) {
        $assocRules = [];
        $n = count($this->items);
        $subsetEq = array_fill(0, $n, 0);
        while ($subsetEq = $this->getNextSubsetEq($subsetEq)) {
            $cond = [];
            $conseq = [];
            for ($i = 0; $i < $n; $i++) {
                if ($subsetEq[$n-1-$i]) {
                    $cond[] = $this->items[$i];
                } else {
                    $conseq[] = $this->items[$i];
                }
            }
            if (!empty($cond) && !empty($conseq)) {
                $assocRules[] = $this->createAssocRule($cond,$conseq,
$freqSets);
            }
```

```
        }

        return $assocRules;
    }
}
```

2.2.2. Класс FuzzyCMeans

```
<?php

/**
 * Класс алгоритма нечеткой кластеризации c-means
 *
 */
class FuzzyCMeans
{
    private $Umatrix;       // Матрица нечеткой принадлежности
    private $centroids;     // Центры кластеров
    private $fuzzy_measure; // Степень нечеткости
    private $input_data;    // Входные данные
    private $num_inputs;    // Число входных наборов
    private $num_clusters;  // Число кластеров

    /**
     * Подсчет значения целевой функции
     *
     * @return float
     */
    private function targetFunc() {
        $sum = 0;
        $m = $this->fuzzy_measure;
        for ($i = 0; $i < $this->num_inputs; $i++) {
            for ($j = 0; $j < $this->num_clusters; $j++) {
                $r = FuzzyCMeansPoint::getDistance($this->input_data[$i],
                        $this->centroids[$j]);
                $sum += pow($this->Umatrix[$i][$j], $m) * pow($r, 2);
            }
        }

        return $sum;
    }

    /**
     * Обновление координат центров кластеров
     *
     * @return void
     */
    private function updateCentroids() {
        $m = $this->fuzzy_measure;
        for ($j = 0; $j < $this->num_clusters; $j++) {
            $Ucol = [];
            for ($i = 0; $i < $this->num_inputs; $i++) {
                $Ucol[$i] = pow($this->Umatrix[$i][$j], $m);
            }
            $this->centroids[$j]->updateByUmatrix($Ucol, $this->input_data);
        }
```

```
}

/**
 * Пересчет матрицы нечеткой принадлежности
 *
 * @return void
 */
public function updateUmatrix() {
    $m = $this->fuzzy_measure;
    for ($i = 0; $i < $this->num_inputs; $i++) {
        $Xi = $this->input_data[$i];
        $Rsum = 0;
        for ($k = 0; $k < count($Xi->getValue()); $k++) {
            $Rik = FuzzyCMeansPoint::getDistance($this->input_data[$i],
                    $this->centroids[$k]);
            $Rsum += pow($Rik, 2/((float)$m-1));
        }
        for ($j = 0; $j < $this->num_clusters; $j++) {
            $Rij = FuzzyCMeansPoint::getDistance($this->input_data[$i],
                    $this->centroids[$j]);
            $this->Umatrix[$i][$j] = pow($Rij, 2/(1-(float)$m)) * $Rsum;
        }
    }
}

/**
 * Нормализация матрицы нечеткой принадлежности
 *
 * @return void
 */
private function normalizeUmatrix() {
    for ($i = 0; $i < $this->num_inputs; $i++) {
        $Uimax = max($this->Umatrix[$i]);
        for ($j = 0; $j < $this->num_clusters; $j++) {
            $this->Umatrix[$i][$j] /= $Uimax;
        }
    }
}

/**
 * Подсчет значения критерия останова
 *
 * @return float
 */
private function getEps(array $oldUmatrix) {
    $eps = -1;
    for ($i = 0; $i < $this->num_inputs; $i++) {
        $Urow = [];
        for ($j = 0; $j < $this->num_clusters; $j++) {
            $Urow[] = abs($this->Umatrix[$i][$j] - $oldUmatrix[$i][$j]);
        }
        $eps = max($eps, max($Urow));
    }

    return $eps > 0 ? $eps : false;
}

/**
```

```
     * Конструктор класса
     *
     * @param array, float, int
     */
    public function __construct(array $input_data, $fuzzy_measure,
$num_clusters) {
        $this->num_inputs = count($input_data);
        $this->input_data = [];
        for ($i = 0; $i < $this->num_inputs; $i++) {
            foreach ($input_data[$i] as $j => $value) {
                $input_data[$i][$j] = (float)$value /
(float)max($input_data[$i]);
            }
            $this->input_data[$i] = new FuzzyCMeansPoint($input_data[$i]);
            $input_size = count($input_data[$i]);
        }
        $this->fuzzy_measure = $fuzzy_measure;
        $this->num_clusters = $num_clusters;

        $this->Umatrix = [];
        for ($i = 0; $i < $this->num_inputs; $i++) {
            for ($j = 0; $j < $this->num_clusters; $j++) {
                $this->Umatrix[$i][$j] = (float)mt_rand() /
(float)mt_getrandmax();
            }
        }

        $this->centroids = [];
        for ($i = 0; $i < $this->num_clusters; $i++) {
            $init_values = [];
            for ($j = 0; $j < $input_size; $j++) {
                $init_values[$j] = (float)mt_rand() / (float)mt_getrandmax();
            }
            $this->centroids[$i] = new FuzzyCMeansPoint($init_values);
        }
    }

    /**
     * Запуск алгоритма c-means
     *
     * @param float
     * @return array
     */
    public function run($targetEps) {
        $iter = 0;
        do {
            $this->updateCentroids();
            $oldUMatrix = $this->Umatrix;
            $this->updateUmatrix();
            $this->normalizeUmatrix();

            $iter++;
            $eps = $this->getEps($oldUMatrix);
            if (!$eps)
                return false;
        } while($eps > $targetEps && $iter < MAX_FCM_ITERNUM);

        return $this->Umatrix;
```

```
    }
}
```

2.2.3. Класс FuzzyCMeansPoint

```
<?php

/**
 * Класс вектора входных данных
 * для алгоритма нечеткой кластеризации c-means
 *
 */
class FuzzyCMeansPoint
{
    private $value; // Значение вектора

    /**
     * Конструктор класса
     *
     * @param array
     */
    public function __construct(array $value) {
        $this->value = $value;
    }

    /**
     * Получение значения вектора
     *
     * @param array
     */
    public function getValue() {
        return $this->value;
    }

    /**
     * Обновление значения центра векторов
     *
     * @param array, array
     * @return void
     */
    public function updateByUmatrix(array $Ucol, array $X) {
        foreach ($this->value as $k => &$Cjk) {
            $sum = 0;
            foreach ($Ucol as $i => $Uij) {
                $sum += $Uij * $X[$i]->value[$k];
            }
            $Cjk = $sum / array_sum($Ucol);
        }
    }

    /**
     * Расстояние между двумя векторами
     *
     * @param FuzzyCMeansPoint, FuzzyCMeansPoint
     * @return float
     */
```

```
    public static function getDistance(FuzzyCMeansPoint $a,
            FuzzyCMeansPoint $b) {
        if (count($a->value) != count($b->value))
            return false;

        $n = count($a->value);
        $sum = 0;
        for ($i = 0; $i < $n; $i++) {
            $sum += pow($a->value[$i] - $b->value[$i], 2);
        }

        return sqrt($sum);
    }
}
```

3. Компоненты ПМ ФИО

3.1. Компонент similar.products

3.1.1. Файл component.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

if (!isset($arParams["CACHE_TIME"]))
    $arParams["CACHE_TIME"] = 36000;

$arParams["IBLOCK_TYPE"] = trim($arParams["IBLOCK_TYPE"]);
$arParams["IBLOCK_ID"] = intval($arParams["IBLOCK_ID"]);

if (!is_array($arParams["FIELD_CODE"]))
    $arParams["FIELD_CODE"] = array();
foreach ($arParams["FIELD_CODE"] as $key => $val) {
    if (!$val)
        unset($arParams["FIELD_CODE"][$key]);
}
if (!is_array($arParams["PROPERTY_CODE"]))
    $arParams["PROPERTY_CODE"] = array();
foreach ($arParams["PROPERTY_CODE"] as $key => $val) {
    if ($val==="")
        unset($arParams["PROPERTY_CODE"][$key]);
}

if (!empty($arParams["FUZZY"]) && ($arParams["FUZZY"] ==
floatval($arParams["FUZZY"]))) {
    $arParams["FUZZY"] = floatval($arParams["FUZZY"]);
} else {
    $arParams["FUZZY"] = 2;
}

if (!empty($arParams["NUM_CLUSTERS"]) && ($arParams["NUM_CLUSTERS"] ==
intval($arParams["NUM_CLUSTERS"]))) {
    $arParams["NUM_CLUSTERS"] = intval($arParams["NUM_CLUSTERS"]);
```

```
} else {
    $arParams["NUM_CLUSTERS"] = 3;
}

if (!empty($arParams["EPS"]) && ($arParams["EPS"] ==
floatval($arParams["EPS"]))) {
    $arParams["EPS"] = floatval($arParams["EPS"]);
} else {
    $arParams["EPS"] = 0.01;
}

if (empty($arParams["PRODUCT_ID"])) {
    $this->IncludeComponentTemplate();
    return;
}

if ($this->StartResultCache()) {
    if (!CModule::includeModule("iblock") || !
CModule::includeModule("sdmm_module")) {
        echo "Модуль sdmm не установлен";
        return;
    }

    //SELECT
    $arSelect = [
        "ID",
        "IBLOCK_ID"
    ];
    foreach ($arParams["FIELD_CODE"] as $field) {
        if ($field && !in_array($field, $arSelect))
            $arSelect[] = $field;
    }
    foreach ($arParams["PROPERTY_CODE"] as $prop) {
        if ($prop && !in_array($prop, $arSelect))
            $arSelect[] = "PROPERTY_".$prop;
    }

    $arFilter = array(
        "ACTIVE" => "Y",
        "IBLOCK_ID" => $arParams["IBLOCK_ID"],
        "IBLOCK_ACTIVE" => "Y",
    );

    $inputData = [];
    $inputIds = [];
    $res = CIBlockElement::GetList(array(), $arFilter, false, false,
$arSelect);
    while($ob = $res->GetNext()) {
        $inputArr = [];
        foreach ($arParams["PROPERTY_CODE"] as $prop) {
            $inputArr[] = $ob['PROPERTY_'.$prop.'_VALUE'];
        }
        $inputIds[] = $ob["ID"];
        $inputData[] = new FuzzyCMeansPoint($inputArr);
    }

    $fuzzy = $arParams["FUZZY"];
    $num_clusters = $arParams["NUM_CLUSTERS"];
```

```
    $eps = $arParams["EPS"];
    $fcm = new FuzzyCMeans($inputData, $fuzzy, $num_clusters);
    $clusters = $fcm->run($eps);

    $searchId = $arParams["PRODUCT_ID"];
    $arResult['SIMILAR'] = $clusters[$inputIds[$searchId]];

    $this->IncludeComponentTemplate();
}
```

3.1.2. Файл parameters.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

if(!CModule::IncludeModule("iblock")){
    echo "Модуль sdmm не установлен";
    return;
}

$arIBlockType = array();
$rsIBlockType = CIBlockType::GetList(
    array("sort"=>"asc"),
    array("ACTIVE"=>"Y")
);
while ($arr = $rsIBlockType->Fetch()) {
    if($ar = CIBlockType::GetByIDLang($arr["ID"], LANGUAGE_ID)) {
        $arIBlockType[$arr["ID"]] = "[".$arr["ID"]."] ".$ar["~NAME"];
    }
}

$arIBlock=array();
$rsIBlock = CIBlock::GetList(
    array("sort" => "asc"),
    array("TYPE" => $arCurrentValues["IBLOCK_TYPE"], "ACTIVE"=>"Y")
);
while($arr = $rsIBlock->Fetch()) {
    $arIBlock[$arr["ID"]] = "[".$arr["ID"]."] ".$arr["NAME"];
}

$arProperty_LNS = array();
$rsProp = CIBlockProperty::GetList(
    array("sort"=>"asc"),
    array(
        "ACTIVE"=>"Y",
        "IBLOCK_ID" => $arCurrentValues["IBLOCK_ID"],
    )
);
while ($arr = $rsProp->Fetch()) {
    if (in_array($arr["PROPERTY_TYPE"], array("L", "N", "S"))) {
            $arProperty_LNS[$arr["CODE"]] = "[".$arr["CODE"]."] ".
$arr["NAME"];
    }
}

$arComponentParameters = array(
    "PARAMETERS" => array(
```

```
        "IBLOCK_TYPE" => array(
            "PARENT" => "BASE",
            "NAME" => "Тип инфоблока",
            "TYPE" => "LIST",
            "VALUES" => $arIBlockType,
            "REFRESH" => "Y"
        ),
        "IBLOCK_ID" => array(
            "PARENT" => "BASE",
            "NAME" => "ID инфоблока",
            "TYPE" => "LIST",
            "ADDITIONAL_VALUES" => "Y",
            "VALUES" => $arIBlock,
            "REFRESH" => "Y",
        ),
        "PRODUCT_ID" => array(
            "PARENT" => "BASE",
            "NAME" => "ID товара",
            "TYPE" => "STRING",
        ),
        "FUZZY" => array(
            "PARENT" => "BASE",
            "NAME" => "Нечеткость кластеризации",
            "TYPE" => "STRING",
        ),
        "NUM_CLUSTERS" => array(
            "PARENT" => "BASE",
            "NAME" => "Число кластеров",
            "TYPE" => "STRING",
        ),
        "EPS" => array(
            "PARENT" => "BASE",
            "NAME" => "Точность кластеризации",
            "TYPE" => "STRING",
        ),
        "FIELD_CODE" => CIBlockParameters::GetFieldCode(
            "Поля",
            "DATA_SOURCE"
        ),
        "PROPERTY_CODE" => array(
            "PARENT" => "DATA_SOURCE",
            "NAME" => "Свойства",
            "TYPE" => "LIST",
            "MULTIPLE" => "Y",
            "ADDITIONAL_VALUES" => "Y",
            "VALUES" => $arProperty_LNS,
        ),
        "SET_TITLE" => array(),
        "CACHE_TIME" => array("DEFAULT" => 3600),
    ),
);
```

3.1.3. Файл description.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentDescription = array(
    "NAME" => "Компонент похожих товаров",
    "DESCRIPTION" => "Компонент похожих товаров модуля SDMM",
    "CACHE_PATH" => "Y",
    "PATH" => array(
        "ID" => "content",
    ),
);
```

3.1.4. Файл template.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();?>

<? foreach ($arResult['SIMILAR'] as $arItem) { ?>
    <div class="product_info">
        <? foreach ($arItem as $prop) { ?>
            <span class="product_prop"><?=$prop?></span>
        <? } ?>
    </div>
<? } ?>
```

3.2. Компонент similar.clients

3.2.1. Файл component.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

if (!isset($arParams["CACHE_TIME"]))
    $arParams["CACHE_TIME"] = 36000;

if (!is_array($arParams["FIELD_CODE"]))
    $arParams["FIELD_CODE"] = array();
foreach ($arParams["FIELD_CODE"] as $key => $val) {
    if (!$val)
        unset($arParams["FIELD_CODE"][$key]);
}

if (!empty($arParams["FUZZY"]) && ($arParams["FUZZY"] ==
floatval($arParams["FUZZY"]))) {
    $arParams["FUZZY"] = floatval($arParams["FUZZY"]);
} else {
    $arParams["FUZZY"] = 2;
}

if (!empty($arParams["NUM_CLUSTERS"]) && ($arParams["NUM_CLUSTERS"] ==
intval($arParams["NUM_CLUSTERS"]))) {
```

```
    $arParams["NUM_CLUSTERS"] = intval($arParams["NUM_CLUSTERS"]);
} else {
    $arParams["NUM_CLUSTERS"] = 3;
}

if (!empty($arParams["EPS"]) && ($arParams["EPS"] ==
floatval($arParams["EPS"]))) {
    $arParams["EPS"] = floatval($arParams["EPS"]);
} else {
    $arParams["EPS"] = 0.01;
}

if (empty($arParams["CLIENT_ID"])) {
    $this->IncludeComponentTemplate();
    return;
}

if ($this->StartResultCache()) {
    if (!CModule::includeModule("sdmm_module")){
        echo "Модуль sdmm не установлен";
        return;
    }

    //SELECT
    $arSelect = [
        "FIELDS" => array(
            "ID",
        ),
    ];
    foreach ($arParams["FIELD_CODE"] as $field) {
        if ($field && !in_array($field, $arSelect))
            $arSelect["SELECT"][] = $field;
    }

    $arFilter = array(
        "ACTIVE" => "Y",
    );

    $inputData = [];
    $inputIds = [];
    $res = CUser::GetList($by = 'id', $order = 'desc', $arFilter, $arSelect);
    while($ob = $res->GetNext()) {
        $inputIds[] = $ob["ID"];
        $inputData[] = new FuzzyCMeansPoint($ob);
    }

    $fuzzy = $arParams["FUZZY"];
    $num_clusters = $arParams["NUM_CLUSTERS"];
    $eps = $arParams["EPS"];
    $fcm = new FuzzyCMeans($inputData, $fuzzy, $num_clusters);
    $clusters = $fcm->run($eps);

    $searchId = $arParams["PRODUCT_ID"];
    $arResult['SIMILAR'] = $clusters[$inputIds[$searchId]];

    $this->IncludeComponentTemplate();
}
```

3.2.2. Файл parameters.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentParameters = array(
    "PARAMETERS" => array(
        "CLIENT_ID" => array(
            "PARENT" => "BASE",
            "NAME" => "ID клиента",
            "TYPE" => "STRING",
        ),
        "FUZZY" => array(
            "PARENT" => "BASE",
            "NAME" => "Нечеткость кластеризации",
            "TYPE" => "STRING",
        ),
        "NUM_CLUSTERS" => array(
            "PARENT" => "BASE",
            "NAME" => "Число кластеров",
            "TYPE" => "STRING",
        ),
        "EPS" => array(
            "PARENT" => "BASE",
            "NAME" => "Точность кластеризации",
            "TYPE" => "STRING",
        ),
        "FIELD_CODE" => CUser::GetFieldCode(
            "Поля",
            "DATA_SOURCE"
        ),
        "SET_TITLE" => array(),
        "CACHE_TIME" => array("DEFAULT" => 3600),
    ),
);
```

3.2.3. Файл description.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentDescription = array(
    "NAME" => "Компонент похожих клиентов",
    "DESCRIPTION" => "Компонент похожих клиентов модуля SDMM",
    "CACHE_PATH" => "Y",
    "PATH" => array(
        "ID" => "content",
    ),
);
```

3.2.4. Файл template.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();?>

<? foreach ($arResult['SIMILAR'] as $arUser) { ?>
    <div class="user_info">
        <? foreach ($arUser as $field) { ?>
            <span class="user_field"><?=$field?></span>
        <? } ?>
    </div>
<? } ?>
```

3.3. Компонент recommended.products

3.3.1. Файл component.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

if (!isset($arParams["CACHE_TIME"]))
    $arParams["CACHE_TIME"] = 36000;

$arParams["IBLOCK_TYPE"] = trim($arParams["IBLOCK_TYPE"]);
$arParams["IBLOCK_ID"] = intval($arParams["IBLOCK_ID"]);

if (!is_array($arParams["FIELD_CODE"]))
    $arParams["FIELD_CODE"] = array();
foreach ($arParams["FIELD_CODE"] as $key => $val) {
    if (!$val)
        unset($arParams["FIELD_CODE"][$key]);
}
if (!is_array($arParams["PROPERTY_CODE"]))
    $arParams["PROPERTY_CODE"] = array();
foreach ($arParams["PROPERTY_CODE"] as $key => $val) {
    if ($val==="")
        unset($arParams["PROPERTY_CODE"][$key]);
}

if (!empty($arParams["MIN_SUPP"]) && ($arParams["MIN_SUPP"] ==
intval($arParams["MIN_SUPP"]))) {
    $arParams["MIN_SUPP"] = intval($arParams["MIN_SUPP"]);
} else {
    $arParams["MIN_SUPP"] = 3;
}

if (!empty($arParams["MIN_CONF"]) && ($arParams["MIN_CONF"] ==
floatval($arParams["MIN_CONF"]))) {
    $arParams["MIN_CONF"] = floatval($arParams["MIN_CONF"]);
} else {
    $arParams["MIN_CONF"] = 0.6; // по умолчанию 60%
}

if (empty($arParams["PRODUCT_ID"])) {
```

```
    $this->IncludeComponentTemplate();
    return;
}

if ($this->StartResultCache()) {
    if (!CModule::includeModule("iblock") || !
CModule::includeModule("sdmm_module")) {
        echo "Модуль sdmm не установлен";
        return;
    }

    //SELECT
    $arSelect = [
        "ID",
        "IBLOCK_ID"
    ];
    foreach ($arParams["FIELD_CODE"] as $field) {
        if ($field && !in_array($field, $arSelect))
            $arSelect[] = $field;
    }
    foreach ($arParams["PROPERTY_CODE"] as $prop) {
        if ($prop && !in_array($prop, $arSelect))
            $arSelect[] = "PROPERTY_".$prop;
    }

    $arFilter = array(
        "ACTIVE" => "Y",
        "IBLOCK_ID" => $arParams["IBLOCK_ID"],
        "IBLOCK_ACTIVE" => "Y",
    );

    $inputData = [];
    $inputIds = [];
    $res = CIBlockElement::GetList(array(), $arFilter, false, false,
$arSelect);
    while($ob = $res->GetNext()) {
        $inputArr = [];
        foreach ($arParams["PROPERTY_CODE"] as $prop) {
            $inputArr[] = $ob['PROPERTY_'.$prop.'_VALUE'];
        }
        $inputIds[] = $ob["ID"];
        $inputData[] = new FuzzyCMeansPoint($inputArr);
    }

    $min_supp = $arParams["MIN_SUPP"];
    $min_conf = $arParams["MIN_CONF"];
    $apriori = new AprioriAlgorithm($inputData);
    $assocRules = $apriori->run($min_supp, $min_conf);

    $searchId = $arParams["PRODUCT_ID"];
    $arResult['RECOMMENDED'] = [];
    foreach ($assocRules as $assocRule) {
        if (in_array($searchId, $assocRule->getCondition() && $assocRule-
>getConfidence() > $arParams["MIN_CONF"])
            $arResult['RECOMMENDED'] = array_merge($arResult['RECOMMENDED'],
$assocRule->getConsequent());
    }
```

```
    $this->IncludeComponentTemplate();
}
```

3.3.2. Файл parameters.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

if(!CModule::IncludeModule("iblock")) {
    echo "Модуль sdmm не установлен";
    return;
}

$arIBlockType = array();
$rsIBlockType = CIBlockType::GetList(
    array("sort"=>"asc"),
    array("ACTIVE"=>"Y")
);
while ($arr = $rsIBlockType->Fetch()) {
    if($ar = CIBlockType::GetByIDLang($arr["ID"], LANGUAGE_ID)) {
        $arIBlockType[$arr["ID"]] = "[".$arr["ID"]."] ".$ar["~NAME"];
    }
}

$arIBlock=array();
$rsIBlock = CIBlock::GetList(
    array("sort" => "asc"),
    array("TYPE" => $arCurrentValues["IBLOCK_TYPE"], "ACTIVE"=>"Y")
);
while($arr = $rsIBlock->Fetch()) {
    $arIBlock[$arr["ID"]] = "[".$arr["ID"]."] ".$arr["NAME"];
}

$arProperty_LNS = array();
$rsProp = CIBlockProperty::GetList(
    array("sort"=>"asc"),
    array(
        "ACTIVE"=>"Y",
        "IBLOCK_ID" => $arCurrentValues["IBLOCK_ID"],
    )
);
while ($arr = $rsProp->Fetch()) {
    if (in_array($arr["PROPERTY_TYPE"], array("L", "N", "S"))) {
            $arProperty_LNS[$arr["CODE"]] = "[".$arr["CODE"]."] ".
$arr["NAME"];
    }
}

$arComponentParameters = array(
    "PARAMETERS" => array(
        "IBLOCK_TYPE" => array(
            "PARENT" => "BASE",
            "NAME" => "Тип инфоблока",
            "TYPE" => "LIST",
            "VALUES" => $arIBlockType,
            "REFRESH" => "Y"
        ),
```

```
        "IBLOCK_ID" => array(
            "PARENT" => "BASE",
            "NAME" => "ID инфоблока",
            "TYPE" => "LIST",
            "ADDITIONAL_VALUES" => "Y",
            "VALUES" => $arIBlock,
            "REFRESH" => "Y",
        ),
        "PRODUCT_ID" => array(
            "PARENT" => "BASE",
            "NAME" => "ID товара",
            "TYPE" => "STRING",
        ),
        "MIN_SUPP" => array(
            "PARENT" => "BASE",
            "NAME" => "Минимальная поддержка",
            "TYPE" => "STRING",
        ),
        "MIN_CONF" => array(
            "PARENT" => "BASE",
            "NAME" => "Минимальная достоверность",
            "TYPE" => "STRING",
        ),
        "FIELD_CODE" => CIBlockParameters::GetFieldCode(
            "Поля",
            "DATA_SOURCE"
        ),
        "PROPERTY_CODE" => array(
            "PARENT" => "DATA_SOURCE",
            "NAME" => "Свойства",
            "TYPE" => "LIST",
            "MULTIPLE" => "Y",
            "ADDITIONAL_VALUES" => "Y",
            "VALUES" => $arProperty_LNS,
        ),
        "SET_TITLE" => array(),
        "CACHE_TIME" => array("DEFAULT" => 3600),
    ),
);
```

3.3.3. Файл description.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentDescription = array(
    "NAME" => "Компонент рекомендуемых товаров",
    "DESCRIPTION" => "Компонент рекомендуемых товаров модуля SDMM",
    "CACHE_PATH" => "Y",
    "PATH" => array(
        "ID" => "content",
    ),
);
```

3.3.4. Файл template.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();?>

<? foreach ($arResult['RECOMMENDED'] as $arItem) { ?>
    <div class="product_info">
        <? foreach ($arItem as $prop) { ?>
            <span class="product_prop"><?=$prop?></span>
        <? } ?>
    </div>
<? } ?>
```

Приложение 2

Программный модуль для реализации функции интеллектуальной обработки данных для системы 1С-Битрикс (ПМ ФИО)

Руководство программиста

Москва, 2016

АННОТАЦИЯ

В данном программном документе приведено руководство программиста по использованию ПМ ФИО, предназначенного для интеллектуальной обработки данных методами кластеризации и ассоциации.
В разделе «Назначение и условия применения программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.
В разделе «Условия выполнения программы» указаны назначение и функции , выполняемые программой, условия, необходимые для выполнения программы (объем оперативной памяти, требования к составу и параметрам периферийных устройств, требования к программному обеспечению и т .п.).
В разделе «Характеристика программы» приведено описание основных характеристик и особенностей программы (режим работы, средства контроля правильности выполнения и т.п.).
В разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.
В разделе «Сообщения » указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы , описание их содержания и действий, которые необходимо предпринять по этим сообщениям.
Оформление программного документа «Руководство программиста» произведено по требованиям ЕСПД (ГОСТ 19.504-79).

ОГЛАВЛЕНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Назначение программы

В настоящее время технологии интеллектуальной обработки данных (ИОД) приобретают все большее распространение: они позволяют извлечь из необработанных данных ранее неизвестные, нетривиальные, практически полезные и доступные интерпретации знания и закономерности. Сфера применения ИОД широка: от биологии и медицины до маркетинга и веб-анализа. Для интернет-приложений наибольший интерес вызывают две задачи ИОД: кластеризация (выделение групп данных, или кластеров) и ассоциация (поиск закономерностей между связанными событиями). В область применения кластеризации и ассоциации входят задачи сегментации данных, анализа веб-логов, выявления похожих товаров и покупателей, рекомендации товаров, выделения групп пользователей и анализа их поведения. Поэтому актуальным является внедрение методов ИОД для решения этих задач в интернет-приложениях и системах управления веб-сайтами.

Для управления содержимым веб-сайтов широко используются системы управления содержимым (СУС). Система 1С-Битрикс является СУС, предназначенной для создания и поддержки корпоративных сайтов, интернет-магазинов, информационных порталов и других веб-проектов.

Данный программный модуль предназначен для реализации методов ИОД для анализа, кластеризации и поиска ассоциативных правил в данных для системы 1С-Битрикс.

1.2. Функции, выполняемые программы

ПМ ФИО обеспечивает выполнение следующих функций:

- систематизация данных;
- обработка данных;
- поиск ассоциативных правил в данных;
- кластеризация данных.

Помимо реализации данных функций, в ПМ ФИО включены следующие компоненты:

- sdmm:similar.products – компонент похожих товаров;

- sdmm:recommended.products – компонент рекомендуемых товаров;
- sdmm:similar.clients – компонент сегментации клиентов.

1.3. Условия, необходимые для выполнения программы

1.3.1. Объем оперативной памяти

Для выполнения функций ПМ ФИО на устройстве должно быть не меньше 1 Гбайт оперативной памяти. Рекомендуемым объемом является 8 Гбайт оперативной памяти.

1.3.2. Требования к составу периферийных устройств

Особых требований к составу периферийных устройств ПМ ФИО не предъявляет.

1.3.3. Требования к параметрам периферийных устройств

Никаких требований к параметрам периферийных устройств модуль не предъявляет.

1.3.4. Требования к программному обеспечению

Для выполнения функций ПМ ФИО в качестве операционной системы должна быть установлена одна из следующих: Windows 7 / 8 / 8.1 / 10, Unix.

ПМ ФИО предназначен для работы в системе 1С-Битрикс, поэтому для корректной работы модуля на устройстве должна быть установлена «1С-Битрикс: Управление сайтом» версии 14.0 или выше.

1.3.5. Требования к персоналу

Программист должен пройти курс 1С-Битрикс «Разработчик Bitrix Framework».

В перечень задач, выполняемых программистом, должны входить:

1) задача поддержания работоспособности системных программных средств – операционной системы и системы управления содержимым 1С-Битрикс;
2) задача поддержания работоспособности модуля ПМ ФИО.

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

2.1. Описание основных характеристик

2.1.1. Режим работы программы

Режим работы модуля ПМ ФИО круглосуточный и непрерывный.

2.1.2. Контроль правильности выполнения программы

Работоспособность модуля ПМ ФИО можно проверить с помощью сценария тестирования, описанного в таблице 1:

Таблица 1 – Сценарий тестирования работоспособности ПМ ФИО

Этап тестирования	Последовательность действий	Ожидаемый результат
1	Установка модуля через панель администрирования 1С-Битрикс (Настройки - Настройки модулей - Установить модуль sdmm)	Сообщение об успешной установке модуля
2	Подключение модуля в программном коде: `CModule::includeModule('sdmm_module');`	Успешное продолжение работы программы

Продолжение таблицы 1

Этап тестирова	Последовательность действий	Ожидаемый результат

ния		

3	Выполнение функции кластеризации для тестового набора данных и следующих параметров: степень нечеткости кластеризации *m* = 1, число кластеров *K* = 3, критерий останова алгоритма *eps* = 0.01. Для этого в коде необходимо написать:
	`$fuzzy = 1;`
	`$num_clusters = 3;`
	`$eps = 0.01;`
	`$fcm = new FuzzyCMeans($inputData, $fuzzy, $num_clusters);`
	`$clusters = $fcm->run($eps);`
		Кластеры данных, на которые разбивается исходное множество данных, записаны в массиве `$clusters`

4	Выполнение функции ассоциации для тестового набора данных и следующих параметров: минимальная поддержка *min_supp* = 3, минимальная достоверность *min_conf* = 0.6. Для этого в коде необходимо написать:
	`$min_supp = 3;`
	`$min_conf = 0.6;`
	`$apriori = new AprioriAlgorithm($inputData);`
	`$assocRules = $apriori->run($min_supp, $min_conf);`
		Ассоциативные правила, удовлетворяющие минимальной поддержке и достоверности, записаны в массиве `$assocRules`

Если полученный результат совпадет с ожидаемым и не получено сообщений об ошибках, значит, модуль является работоспособным.

3. Обращение к программе

3.1. Установка ПМ ФИО

Установка модуля производится через панель администрирования 1С-Битрикс (Настройки - Настройки модулей - Установить модуль sdmm).

3.2. Подключение модуля ПМ ФИО

Для установки ПМ ФИО в программном коде необходимо написать следующую команду:

```
CModule::includeModule('sdmm_module');
```

После этого в коде будут доступны функции модуля

3.3. Установка компонентов ПМ ФИО

3.3.1. Установка компонента similar.products

Для подключения компонента similar.products необходимо вызвать метод

```
$APPLICATION->IncludeComponent(
 "similar.products",
 "",
 $arParams = array(
        "PRODUCT_ID" => $product_id,     // ID товара
           "FUZZY" => $fuzzy,                    // степень нечеткости кластеризации
        "NUM_CLUSTERS" => $num_clusters, // число кластеров
        "EPS" => $eps,                   // критерий останова
             "FIELD_CODE" => $field_code,        // поле, по которому производится кластеризация
 ),
);
```

Первый параметр метода отвечает за имя подключаемого компонента, второй – за подключаемый шаблон компонента, третий – за входные параметры, передаваемые в

компонент: ID товара, степень нечеткости кластеризации, число кластеров, критерий останова, поле, по которому производится кластеризация.

3.3.2. Установка компонента similar.clients

Для подключения компонента similar.clients необходимо вызвать метод

```
$APPLICATION->IncludeComponent(
 "similar.clients",
 "",
 $arParams = array(
        "CLIENT_ID" => $client_id,       // ID клиента
           "FUZZY" => $fuzzy,                    // степень нечеткости кластеризации
        "NUM_CLUSTERS" => $num_clusters, // число кластеров
        "EPS" => $eps,                    // критерий останова
            "FIELD_CODE" => $field_code,        // поле, по которому производится кластеризация
 ),
);
```

Первый параметр метода отвечает за имя подключаемого компонента, второй – за подключаемый шаблон компонента, третий – за входные параметры, передаваемые в компонент: ID клиента, степень нечеткости кластеризации, число кластеров, критерий останова, поле, по которому производится кластеризация.

3.3.3. Установка компонента recommended.products

Для подключения компонента recommended.products необходимо вызвать метод

```
$APPLICATION->IncludeComponent(
 "recommended.products",
 "",
 $arParams = array(
        "PRODUCT_ID" => $product_id, // ID товара
        "MIN_SUPP" => $min_supp,     // минимальная поддержка
        "MIN_CONF" => $min_conf,     // минимальная достоверность
         "FIELD_CODE" => $field_code, // поле, по которому производится ассоциация
```

```
    ),
);
```

Первый параметр метода отвечает за имя подключаемого компонента, второй – за подключаемый шаблон компонента, третий – за входные параметры, передаваемые в компонент: ID товара, минимальная поддержка, минимальная достоверность, поле, по которому производится ассоциация.

3.3. Выполнение программы

3.3.1. Запуск кластеризации

После подключения модуля ПМ ФИО его методы кластеризации вызываются следующим образом:

3) создается объект класса FuzzyCMeans при помощи конструктора данного класса (inputData - массив входных данных, fuzzy - степень нечеткости кластеризации, num_clusters - число кластеров, на которые будут разделены данные):

```
$fcm = new FuzzyCMeans($inputData, $fuzzy, $num_clusters);
```

4) вызывается метод run() созданного объекта, аргументом которого является eps – критерий останова алгоритма, а на выходе которого получается массив кластеров, на которые были разбиты входные данные:

```
$clusters = $fcm->run($eps);
```

После этого в массиве clusters производится поиск нужного товара и выводится список похожих на него товаров, т.е. находящихся с ним в одном кластере:

```
$searchId = $arParams["PRODUCT_ID"]; // исходный товар
 $result = $clusters[$inputIds[$searchId]]; // похожие на исходный товары
```

3.3.2. Запуск поиска ассоциативных правил

Для поиска рекомендуемых товаров в компоненте recommended.products подключается модуль ПМ ФИО и вызываются его методы поиска ассоциативных правил следующим образом:

4) создается объект класса AprioriAlgorithm при помощи конструктора данного класса (inputData – набор транзакций):

```
$apriori = new AprioriAlgorithm($inputData);
```

5) вызывается метод run() созданного объекта, аргументами которого являются минимальная поддержка кандидатов (число часто встречающихся наборов данных в транзакциях) и минимальная достоверность правил, а на выходе получается массив ассоциативных правил с рассчитанными достоверностями:

```
$assocRules = $apriori->run($min_supp, $min_conf);
```

6) Из найденных ассоциативных правил выделяются правила, удовлетворяющие минимальной достоверности, и для данного товара выводятся рекомендуемые для покупки с данным товары:

```
$min_supp = $arParams["MIN_SUPP"];
$min_conf = $arParams["MIN_CONF"];
$apriori = new AprioriAlgorithm($inputData);
$assocRules = $apriori->run($min_supp, $min_conf);
```

3.4. Завершение программы

Для завершения программы необходимо закрыть веб-страницу, на которой применяются методы ПМ ФИО.

Для удаления ПМ ФИО из системы 1С-Битрикс необходимо перейти в панель администрирования 1С-Битрикс и удалить модуль sdmm (Настройки - Настройки модулей - Удалить модуль sdmm).

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1. Организация входной информации

Входная информация представляется в виде наборов элементов информационных блоков системы «1С-Битрикс», передаваемых в методы ПМ ФИО.

4.2. Организация выходной информации

Выходная информация представляется в виде вывода кластеров данных для задачи кластеризации и ассоциативных правил для задачи ассоциации.

5. СООБЩЕНИЯ ПРОГРАММИСТУ

В таблице 2 указан перечень сообщений, которые появляются в случае неисправности.

Таблица 2 – Перечень возможных неисправностей

Сообщение	Причина	Действия программы	Действия программиста
Модуль sdmm не установлен	Попытка вызвать методы ПМ ФИО до установки модуля	Выдается сообщение об ошибке, дальнейшая загрузка страницы не производится	Установить модуль sdmm через панель администрирования 1С-Битрикс
DB query error	Ошибка базы данных		Необходимо воспользоваться инструментом восстановления БД из панели администрирования 1С-Битрикс (Настройки - Инструменты - Диагностика - Проверка БД)
Error connecting to database	Некорректные настройки подключения к БД		Проверить параметры подключения к базе данных
500 Internal Server Error	Нарушение конфигурации сервера		Необходимо обратиться к администраторам веб-сервера