Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Системный анализ и проектирование информационных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПРИНЯТИЯ РЕШЕНИЙ МЕТОДОМ ВЗВЕШЕННЫХ ЭКСПЕРТНЫХ ОЦЕНОК (НА ПРИМЕРЕ ПРОИЗВОДСТВА АВТОЗАПЧАСТЕЙ)**

Студент гр. 672303 Ставер И.В

Руководитель ассистент кафедры ЭИ,

Хомяков П.В.

Минск 2018

СОДЕРЖАНИЕ

Введение 3

1 Постановка задачи 5

2 Описание работы производства автомобильных запчастей 6

3 Построение базовой аналитической модели 12

4 Описание архитектуры разрабатываемого срества 14

5 Алгоритм работы программы 16

6 Программная реализация 19

7 Руководство пользователю 21

8 Контрольный пример 29

Заключение 30

Список используемых источников 31

Приложение А (обязательное) 32

Приложение Б (обязательное) листинг кода 37

**ВВЕДЕНИЕ**

Исходя из сегодняшнего дня, в мире присутствует огромное количество информации, которую желательно правильно систематизировать. Предприятие, каждый раз обрабатывая и систематизируя информацию, имеет риск столкнуться с ошибками и проблемами, которые могут привести к убыткам. Исходя из этого можно сделать вывод, что даже самая мелкая организация требует использования автоматических систем расчета данных. Каждое предприятие так или иначе осуществляет работу с финансами. Если представить, что ошибки с анализом и обработкой информации возникли в финансовой сфере предприятия, то предприятие понесет убытки, а иногда может и полностью обанкротится и выйти с рынка. Например, производство автомобильных запчастей. В данной сфере предприятия зачастую напрямую контактируют с клиентами, за счет чего и получает доход.

Анализ и систематизация информации о внутренних и внешних проблемах предприятия, производства продукции, которую изготавливает предприятие, а также вычисления благоприятного исхода тех или иных событий нередко становится проблемой.

Использование электронных вычислительных средств может помочь решить эту проблему. Создавая продукт, который сможет осуществлять оценку вышесказанных проблем, может стать очень полезным для предприятия. Приложение поможет хранить и выбирать наилучший исход выбора по тому или иному вопросу.

Целью работы является улучшение качества работы предприятия по производству автомобильных запчастей путем создания качественной и автоматизированной программы, с помощью которой руководство, а также персонал предприятия смогут анализировать и систематизировать, а также с легкостью управлять информацией о предприятии.

Задачи, которые необходимо решить для достижения поставленной цели:

* исследование и изучение механизма работы предприятия по производству автомобильных запчастей;
* изучения метода взвешенных экспертных оценок для решения задачи;
* построение IDEF0 и UML-диаграмм;
* разработка структур данных;
* разработка пользовательских функций приложения;
* разработка алгоритма, реализующего бизнес-логику;
* разработка пользовательского интерфейса;
* организация работы клиент-серверного приложения;
* тестирование приложения;
* отладка ошибок.

Объектом исследования является предприятие по производству автомобильных запчастей.

# 

# **1 ПОСТАНОВКА ЗАДАЧИ**

Производством автомобильных запчастей занимается предприятие или компания. И не редко предприятие сталкивается с проблемами при производстве автозапчастей.

Предприятие взаимодействует с различными поставщиками ресурсов, заказчиками, клиентами и порой оно сталкивается с непредвиденными проблемами взаимодействующих сторон. Обычно, у проблем есть несколько решений и порой действительно сложно выбрать правильное.

Согласно поставленного условия, следует разработать программное приложение, которое поможет осуществлять выбор. А именно, отвечать на подобного типа вопросы: «Какое решение принять при отсутствии поставок материала от поставщика?». Для этого может использоваться метод взвешенных экспертных оценок. У каждого из экспертов имеется свой уровень компетентности в решении той или иной проблемы. Каждый эксперт выставляет свою оценку по каждому из решений проблемы, исходя из своих предпочтений, опыта, интуиции и т.д.

Приложение должно облегчить работу компании, помочь в получении наибольшей выручки, дальнейшему развитию и процветанию.

# **2 ОПИСАНИЕ РАБОТЫ ПРОИЗВОДСТВА**

# **АВТОМОБИЛЬНЫХ ЗАПЧАСТЕЙ**

Производство, в экономическом смысле – процесс создания какого-либо продукта.

Понятие производства характеризует специфически человеческий тип обмена веществами с природой, или, более точно — процесс активного преобразования людьми природных ресурсов в какой-либо продукт.

Предприятие, производящие автомобильные запчасти, в основном производит один тип продукции и таким образом могут возникать вопросы: «Какие запчасти выгоднее производить?».

Для производства автомобильных запчастей была создана стратегическая карта производства (рисунок 2.1), а также были определены цели к которым должно стремится предприятие.

Любая организация стремится максимизировать свою прибыль, для этого ей необходимо сократить издержки производства, а также привлечь новых клиентов. Дабы не упасть в глазах старых клиентов, предприятие должно повышать качество работы. Для этого следует изготовлять продукцию из качественных материалов, своевременно выполнять работу, а также оптимизировать запасы оборудования. От высоких издержек производства может также помочь своевременная наладка оборудования. Планирование и контроль проекта нужно чтобы своевременно выполнять работы. Основу всего предприятия составляют его работники, следовательно, нужно постоянно повышать их квалификацию, а также следить за их профессионализмом.

Также была разработана диаграмма IDEF0. IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. Данная диаграмма способна обеспечить полное представление, как о функционировании процесса, так и об информации и материалах, имеющихся в нем. Далее будет рассмотрен процесс, который является одним из ключевых в исследуемой теме, а именно «Продать автомобильную запчасть».

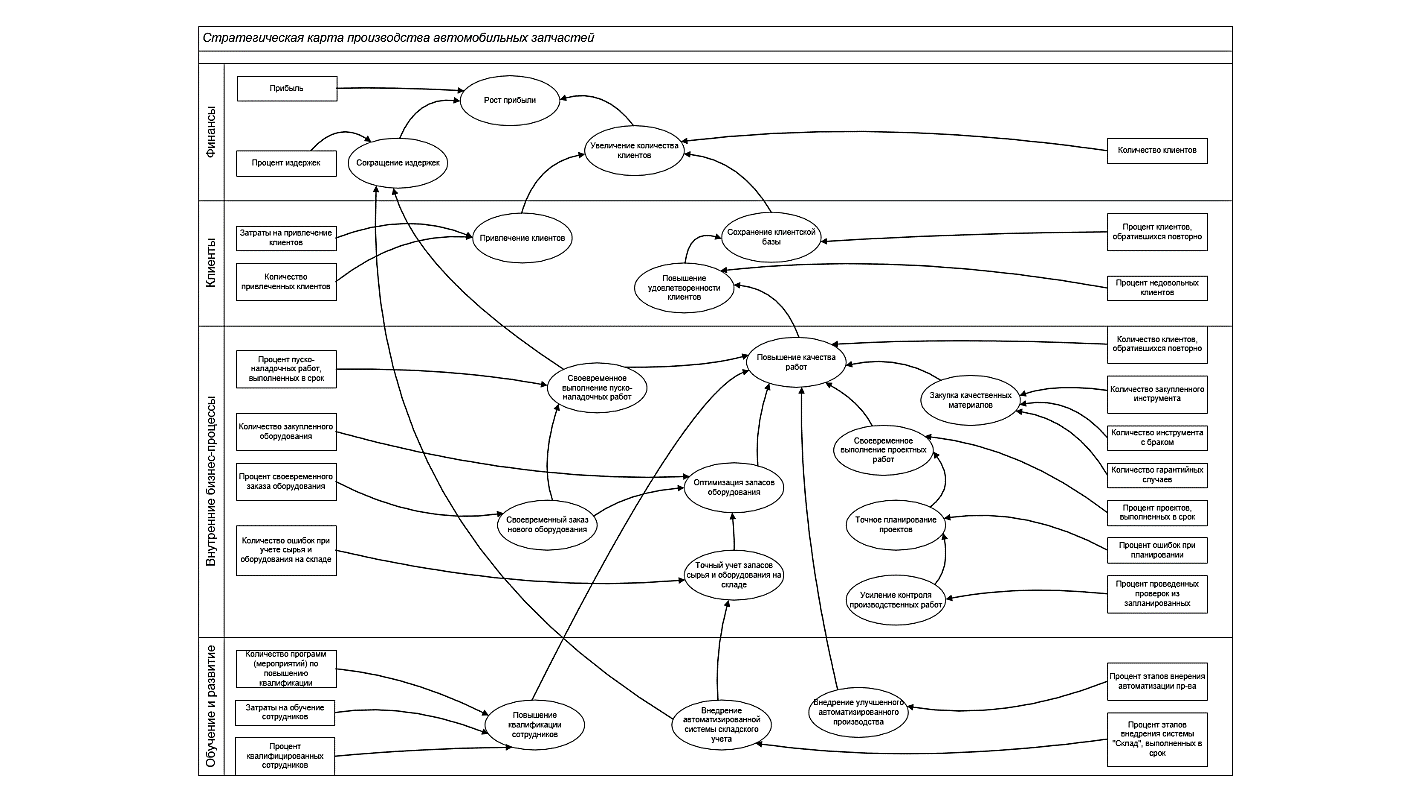


Рисунок 2.1 – Стратегическая карта производства автомобильных запчастей

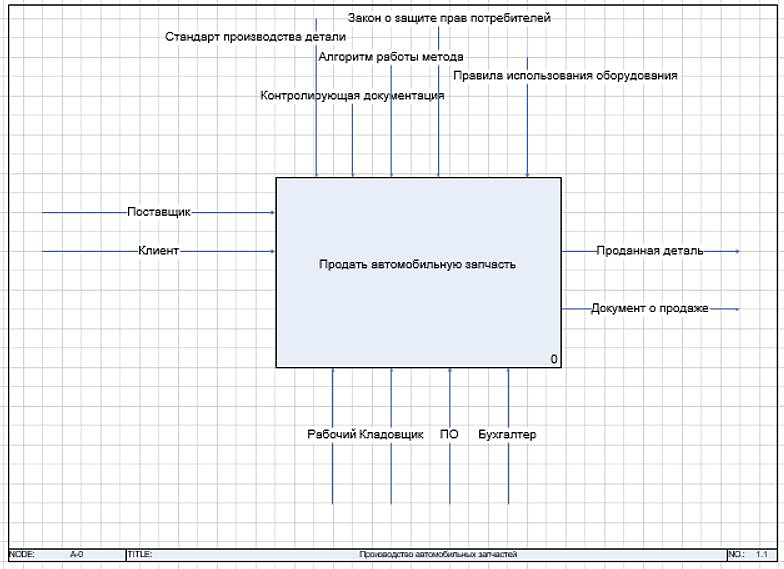
****Рисунок 2.2 представляет первый уровень диаграммы IDEF0, а именно уровень А0. Что нужно чтобы продать автомобильные запчасти? В первую очередь нужны материалы, оборудование и персонал. Для материалов нам нужен поставщик, а собственно для продажи нам нужен клиент. Также все наше предприятие должно регламентироваться правовыми и административными документами, а также нашим алгоритмом взвешенных экспертных оценок. На выходе мы получаем проданную деталь, а также подтверждающий документ.

Рисунок 2.2 – Уровень A0 IDEF0

Декомпозиция данного блока представляется 4-мя блоками предсталенные на рисунке 2.3: «Принять новые материалы и оборудование», «Получить материалы со склада», «Произвести детали» и «Продать деталь». Для продажи детали нам сначало необходимо организовать работу кладовщика, который должен будет принять новые материалы и оборудование, а также их зарегистрировать. После этого мы можем получить нужные материалы для изготовления конкретной модели детали, а далее уже её продать.

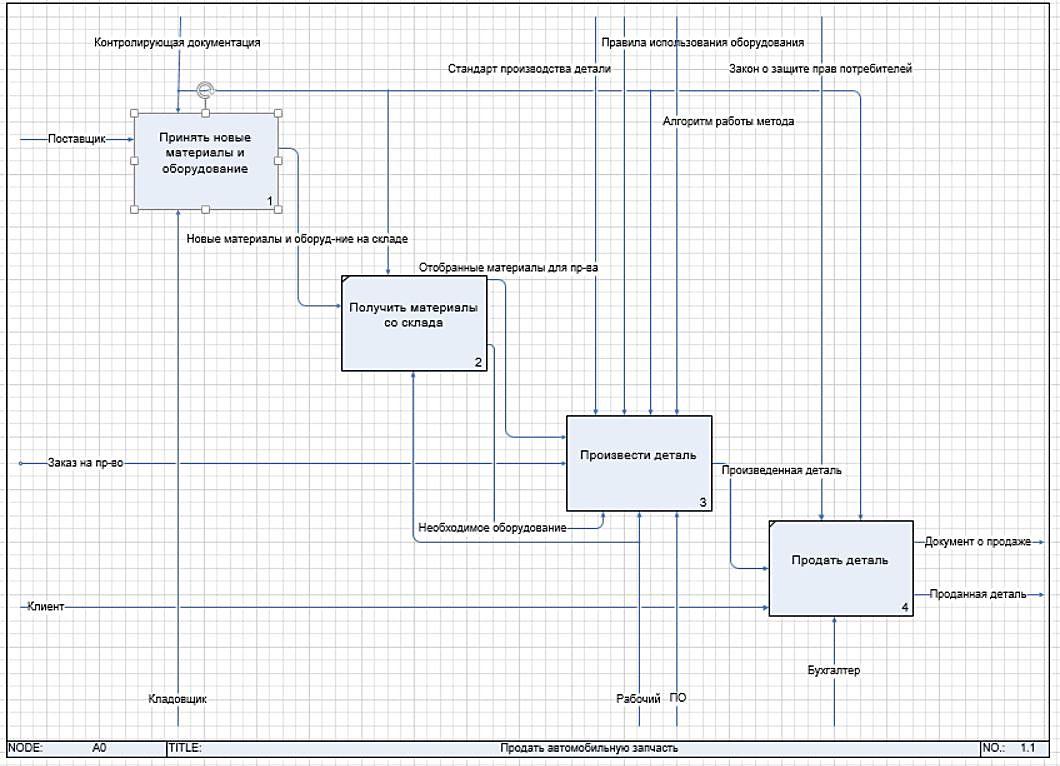


Рисунок 2.3 – Декомпозиция блока «Продать автомобильную запчасть»

Рассмотрим декомпозицию блока «Принять новые материалы и оборудование» (рисунок 2.4). Данный блок разбивается на следующие совокупности блоков: «Встретить поставщика», «Проверить правильность доставки», «Зарегистрировать новые материалы и оборудование», «Разместить на складе». Для выполнения нашей операции следует для начала встретить поставщика, мы получаем от него наш заказ. Далее нам необходимо проверить правильность доставки, если же возникла ошибка, то придется перезаказывать заказ. В случае успеха, мы получаем проверенную доставку. После регистрации новых материалов и оборудования, мы получаем зарегистрированный заказ. Далее нам необходимо разместить их на складе для последующего использования в производстве.

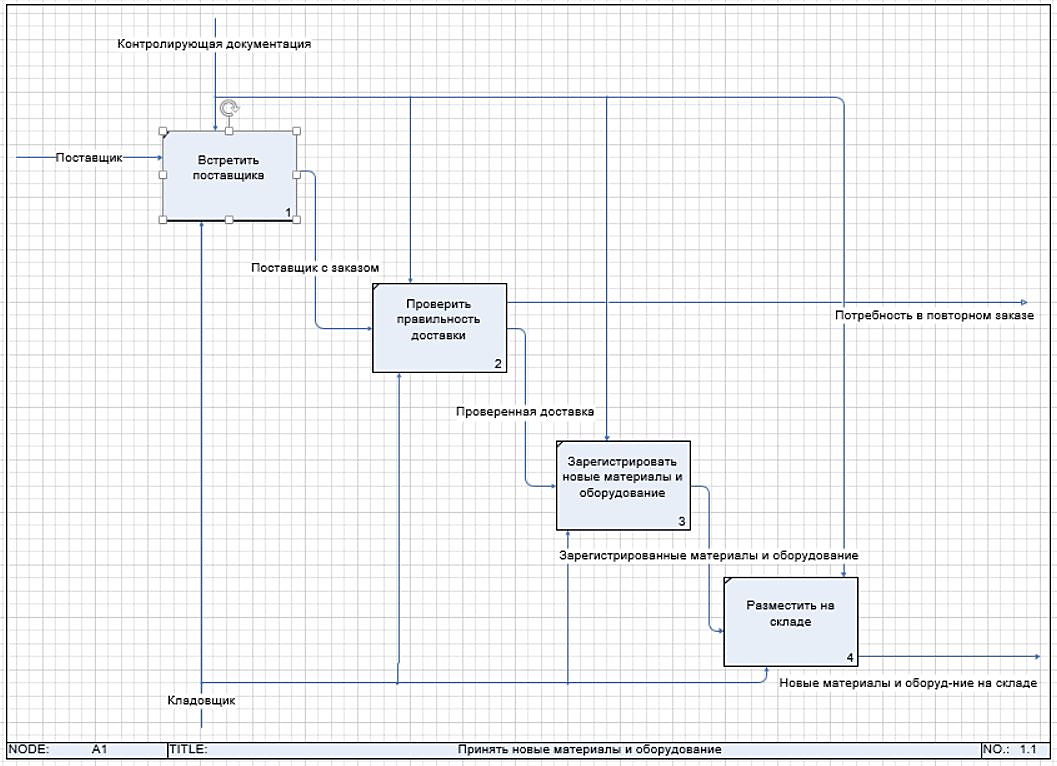
****

Рисунок 2.4 – Декомпозиция блока «Принять новые материалы и оборудование»

На рисунке 2.5 представлена декомпозиция блока «Произвести деталь».

Данный блок можно разбить на «Получить заказ на производство после выбора методом экспертных оценок», «Ознакомиться с правилами работы и производства выбранной детали» и «Изготовить деталь». После того, как получен результат выбора и производства, получим ознакомленного рабочего и, в итоге, - произведенную деталь.

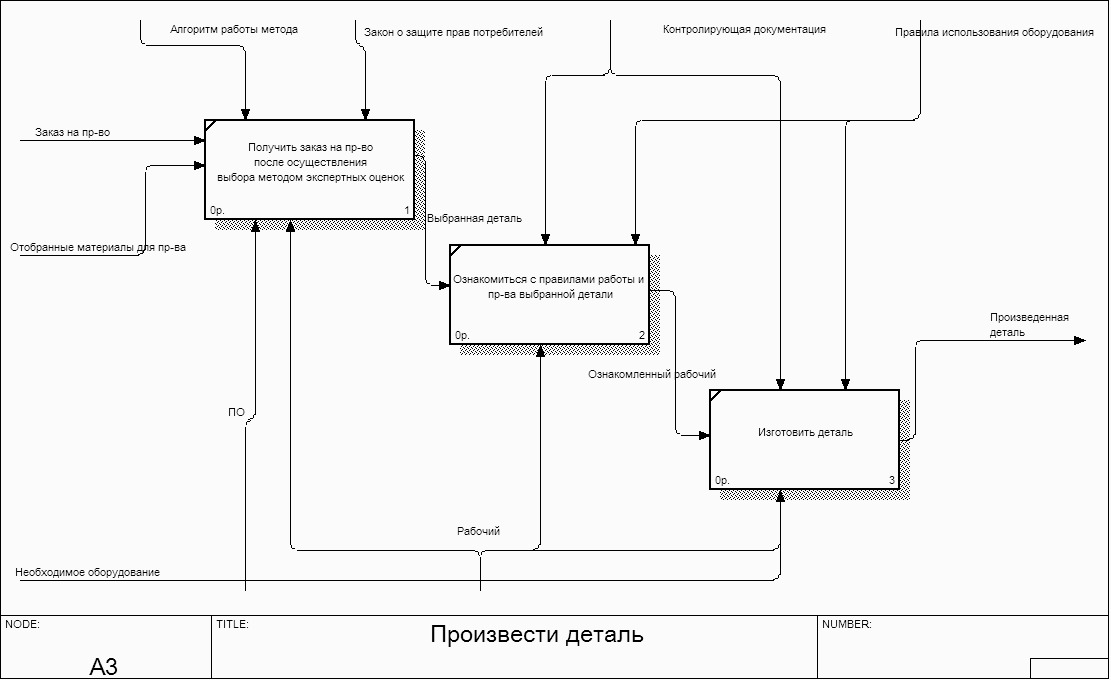


Рисунок 2.5 – Декомпозиция блока «Произвести деталь»

В свою очередь (рисунок 2.6), блок «Получить заказ на производство после осуществления выбора методом экспертных оценок» можно разбить на следующие блоки: «Cоставить матрицу оценок», «Вычислить относительные рейтинги экспертов», «Вычислить веса целей», «Сделать вывод». Каждый этап соответствует алгоритму работы метода взвешенных экспертных оценок.

Диаграммы UML представлены в Приложении А. [2]

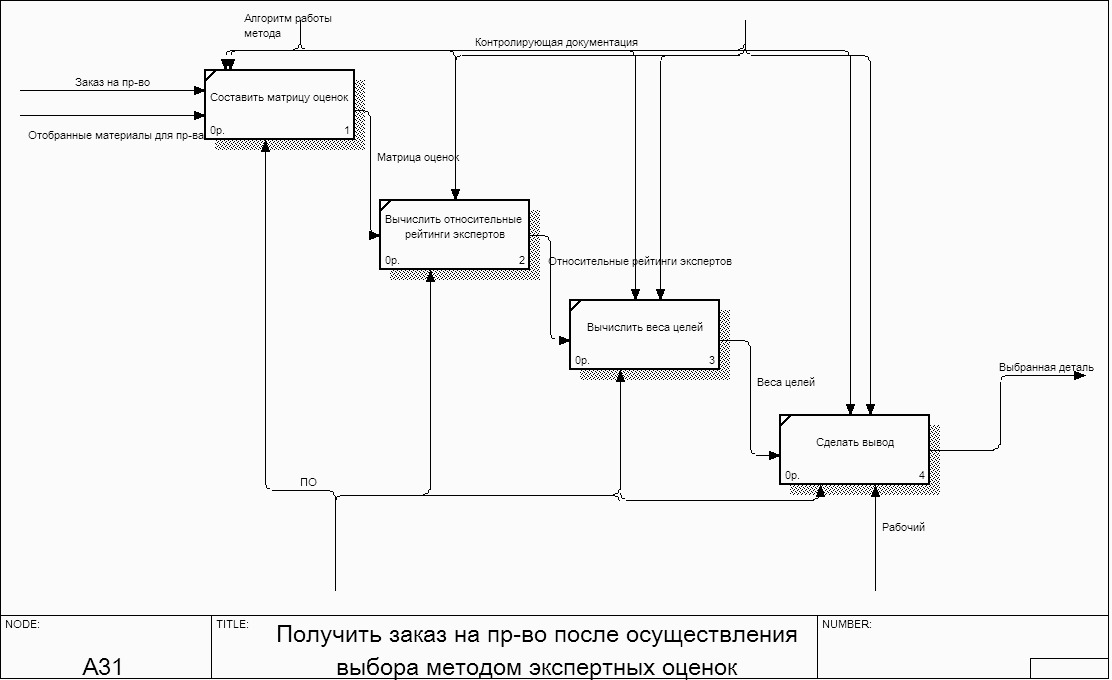


Рисунок 2.7 – Декомпозиция блока «Получить заказ на производство после осуществления выбора методом экспертных оценок»

# **3 ПОСТРОЕНИЕ БАЗОВОЙ АНАЛИТИЧЕСКОЙ**

# **МОДЕЛИ**

Согласно заданию, решить задачу надо с помощью метода взвешенных экспертных оценок. Данный метод является частью обширной области теории принятия решений.

Существует две группы экспертных оценок:

* индивидуальные оценки основанные на использовании мнения отдельных экспертов, независимых друг от друга;
* коллективные оценки, основанные на использовании коллективного мнения экспертов.

Рассмотрим метод взвешенных экспертных оценок подробнее.

Пусть имеется m экспертов: Э1, Э2, …, Эm, которые характеризуются оценками компетентности: R1, R2, … Rm.

Каждый эксперт независимо от других проводит оценку n целей: Z1, Z2, …, Zn.

В результате m независимых экспертиз получена матрица весов целей Vij:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Эj/Zi | Z1 | Z2 | … | Zn |
| Э1 | S11 | S12 | … | S1n |
| Э2 | S21 | S22 | … | S2n |
| … | … | … | … | … |
| Эm | Sm1 | Sm2 | … | Smn |

В этих условиях веса целей определяются формулой:

Относительный коэффициент компетентности:

Компетентность экспертов зависит от множества факторов:

* занимаемой должности;
* ученой степени;
* ученого звания;
* опыта практической работы;
* числа научных трудов;
* знания достижений науки и техники;
* понимания проблем и перспектив развития и др.

Если учитывать только 2 первых фактора, то можно предложить матрицу оценок компетентности экспертов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Rj |  |  |
| Занимаемая должность | специалист без степени | кандидат наук | доктор  наук | академик |
| Ведущий инженер | 1 | - | - | - |
| С.Н.С., Н.С., М.Н.С | 1 | 1,5 | - | - |
| Зав. лабораторией, сектора | 2 | 3 | 4 | 6 |
| Зав. отдела, заместитель | 2,5 | 3,75 | 5 | 7,5 |
| Руководитель комплекса | 3 | 4,5 | 6 | 9 |
| Директор, заместитель | 4 | 6 | 8 | 12 |

Рассмотрим пример.

Два эксперта Э1 и Э2 заводят оценку 4-х целей: Z1, Z2, Z3, Z4.

В результате 2-х независимых экспертиз получена матрица весов целей:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Эj/ Zi | Z1 | Z2 | Z3 | Z4 |
| Э1 | 0,5 | 0 | 0,33 | 0,17 |
| Э2 | 0,54 | 0,04 | 0,2 | 0,17 |

Определим оценки компетентности экспертов, используя таблицу:

* Э1 (руководитель комплекса, кандидат наук) → R1 = 4,5;
* Э2 (директор, доктор наук) → R2 = 8.

Вычислим относительные оценки компетентности экспертов:

;

.

Найдем искомые веса целей:

;

Где сумма Wi должна равняться 1.

Получаем, следовательно, предпочтения целей: Z1, Z3, Z4, Z2

# **4 ОПИСАНИЕ АРХИТЕКТУРЫ РАЗРАБАТЫВАЕМОГО**

**СРЕДСТВА**

«Клиент — сервер» — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, работа с базами данных и др.) или в виде сервисных функций. Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

Серверная часть курсовой работы представляет собой среду выполнения JavaScript кода – Node.js. Технология Node.js позволяет интерпретировать JavaScript текст в исполняемый код. Для создания именно сервера для работы с клиентом используется сторонняя библиотека Express, которая предоставляет методы для легкого создания и конфигурации сервера на языке JavaScript.

Node.js использует цикл событий вместо потоков и способен поддерживать миллионы одновременных соединений. Каждое действие с вводом/выводом - асинхронное, это означает, что сервер может продолжать обрабатывать входящие запросы, во время операций по вводу/выводу. JavaScript отлично подходит для программирования, основанного на событиях, он обладает анонимными функциями и замыканиями, что позволяет с легкостью создавать каллбеки и JavaScript разработчики уже знакомы с подобным способом программирования. Данная событийная модель делает Node.js невероятно быстрым и позволяет без проблем масштабировать приложения в реальном времени.

Клиентская часть приложения разработана в виде HTML-страниц, CSS-стилей, а также JavaScript логики.

Язык HTML – стандартизированный язык разметки во Всемирной паутине. HTML интерпретируется браузерами, полученный в результате интерпретации форматированный текст отображается на экране монитора.

Каскадная таблица стилей (CSS) - формальный язык описания внешнего вида документа, написанного с использованием языка разметки, в данном случае HTML.

JavaScript – мультипарадигменный язык программирования. Он обычно используется как встраиваемый язык для программного доступа к объектам приложений.

Когда пользователь осуществляет свою работу с программой, то он контактирует непосредственно с клиентской частью. А она, в свою очередь, обращается к серверу.

Данные хранятся в базе данных MongoDB. MongoDB – документно-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Эта база данных классифицирована как NoSQL и использует JSON – подобные документы и схему базы данных.

Работа с MongoDB производится с помощью сторонней библиотеки Mongoose. Эта библиотека предоставляет асинхронные методы для доступа к коллекциям, сущностям базы данных.

# **5 АЛГОРИТМ РАБОТЫ ПРОГРАММЫ**

Работа программы начинается с ввода данных для входа. Данные вводятся в форму входа на странице /form или же на странице /registration. После нажатия на кнопку «Вход», с полей формы входа считывается информация. Формируется POST запрос к серверу, куда в теле запроса передается информация с логином и паролем пользователя.

Как только сервер получает POST запрос от страницы /form или /registration, он применяет к телу запроса стороннюю библиотеку BodyParser, которая расшифровывает данные полученные из тела запроса. Данные начинают представлять собой JSON-объект, где под ключами login и password находятся значения формы, которую отправил пользователь. Библиотека Moongoose с помощью запроса к базе данных пытается найти указанные данные в коллекции users. Если данные были найдены, то запрос возвращает объект с введенными данными. Далее проверяется роль пользователя и в зависимости от этого происходит переадресация на соответствующею роли HTML-страницу. Блок-схема данного процесса представлена на рисунке 5.1.

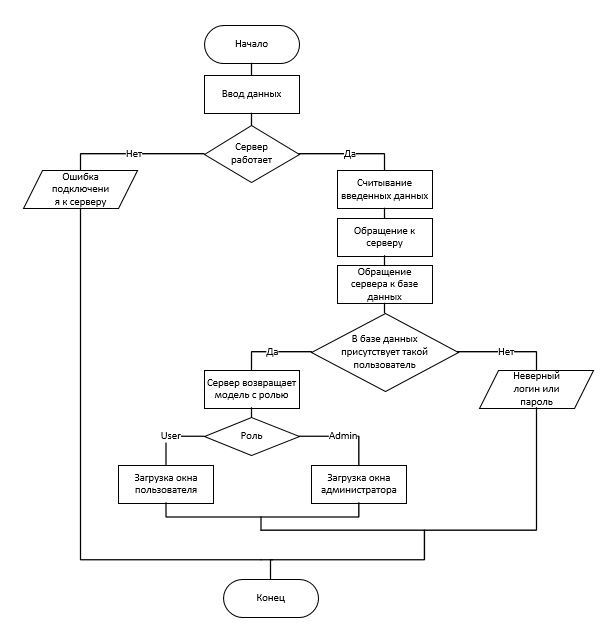


Рисунок 5.1 – Блок-схема взаимодействия клиент-сервер при авторизации

Пускай возвращается значение роли «User». Следовательно, происходит переадресация на HTML-страницу /user. Пользователю предоставляется возможность решить задачу. Но для начала ему следует выбрать проблему, решение которой он хочет получить. После того как пользователь выбрал проблему(продукт) и нажал на кнопку «Ввод», пользователю предоставляется выбор из доступных экспертов, которые уже оценили данную проблему(продукт). Путем нажимания на имена экспертов, пользователь таким случаем выбирает их для решения поставленной задачи. После выборки экспертов и нажатия кнопки «Ввод», пользователь получает: название проблемы, таблицу с данными экспертов, наиболее выгодное решение проблемы, а также кнопки «На главную» и «Показать график проблемы».

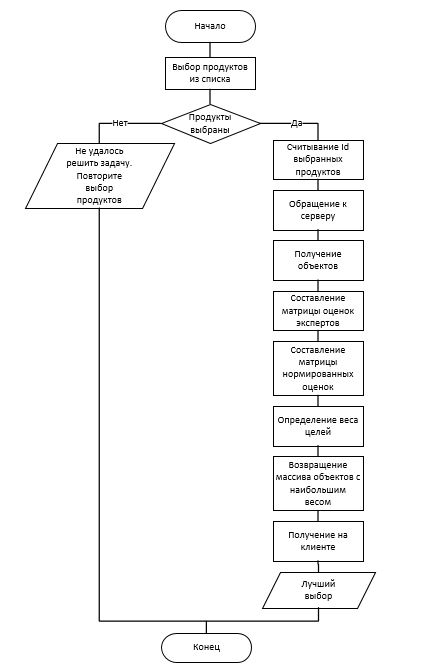
Алгоритм решения задачи с учетом выбранных экспертов следующий: на сервер, после выбора проблемы(продуктов) пользователем, отправляется запрос, который просит сервер найти в базе данных проблему с таким id, а также все её доступные решения экспертами. Все данные после этого отправляются обратно на страницу /user в виде JSON-объекта. Данные анализируются и выводятся в доступной форме пользователю. Блок-схема алгоритма приведена на рисунке 5.2.

Рисунок 5.2 – Блок-схема работы метода взвешенных экспертных оценок

# **6 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

В основе работы Node.js заложен механизм работы с пакетами и отдельными модулями. Отдельные модули Node.js могут подключаться к главному модулю программы с помощью методов Node.js. Отдельные модули используются в качестве классов, отдельных функций и т.д.

В серверной части созданы и подключены следующие модули:

* app.js;
* config.js;
* database.js;
* index.js;
* problem.js;
* user.js.

Остановимся на каждом из модулей подробней.

App.js представляет собой главный файл серверного приложения, в который подключается все остальные модули. Именно в App.js происходит разбор запросов со стороны клиента, а также последующая переадресация на нужные html-страницы.

Config.js представляет собой конфигурационный файл в котором прописан порт на котором следует открыть сервер, а также URL к базе данных.

В database.js происходит подключение к базе данных MongoDB, а также последующая отладка ошибок встроенными методами Express.

Index.js представляет собой своеобразную точку входа серверного приложения. В него подключается app.js, config.js, database.js. И с помощью библиотек Express и Mongoose включается сервер.

Problem.js определяет записи коллекции problems в базе данных MongoDB.

User.js определяет записи коллекции users в базе данных MongoDB.

Для удобной передачи данных на HTML-страницы используется шаблонизатор EJS. Он предоставляет возможность писать JavaScript код внутри HTML-разметки. Для взаимодействия с программой были созданы следующие EJS файлы:

* 403.ejs;
* 404.ejs;
* admin.ejs;
* form.ejs;
* index.ejs;
* registration.ejs;
* user.ejs.
* expert.ejs

Рассмотрим EJS файлы подробнее.

403.ejs представляет собой страницу, которая будет отображаться при ошибке 403(Доступ запрещен).

404.ejs аналогично 403.ejs представляет страницу, которая будет отображаться при ошибке сервера 404(Страница не найдена).

Admin.ejs – это страница администратора, в которую подключаются admin.css, а также клиентский файл JavaScript кода, admin.js.

Form.ejs – страница с формой входа на сайт, в которую подключаются

form.css, а также файл JS кода, form.js

Index.ejs представляет главную страницу, на которой расположены кнопки для перехода на form.ejs и registration.ejs

Registration.ejs аналогично form.ejs, но она предоставляет форму регистрации.

User.ejs – страница пользователя.

Expert.ejs – страница эксперта.

Для приятного визуального отображения данных на страницах используются следующие CSS файлы:

* admin.css
* expert.css
* form.css
* index.css
* registration.css
* user.css

За логику клиентской части отвечают следующие JS файлы:

* admin.js
* expert.js
* user.js

# **7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Данное приложение является сайтом в интернет сети и поэтому доступ к нему осуществляется с помощью перехода по ссылке: <https://blooming-tundra-24684.herokuapp.com/>

При переходе по ссылке пользователь попадает на главную страницу сайта, где он может зарегистрироваться или авторизоваться. Главная страница представлена на рисунке 7.1.

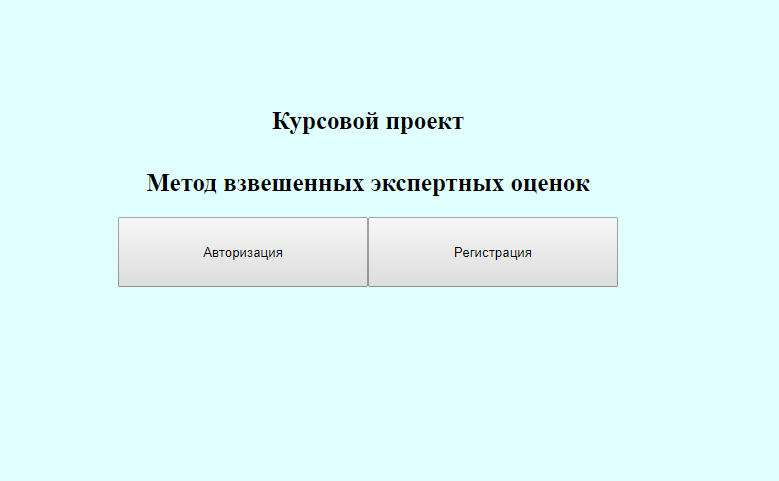


Рисунок 7.1 – Главная страница

При нажатии на кнопку «Авторизация» происходит переход на страницу авторизации, где пользователь вводит свои данные для входа в систему (рисунок 7.2).



Рисунок 7.2 – Страница авторизации

При авторизации администратора, приложение переходит по ссылке /admin, где появляется панель администратора. Администратор может просматривать данные о пользователях, а также удалять их (рисунок 7.3). Добавление и редактирование осуществляется напрямую с базой данных с помощью сторонней программы Robo3T, которая позволяет гибко настраивать пользователей, а также проблемы, решаемые ими (рисунок 7.4).



Рисунок 7.3 – Панель администратора

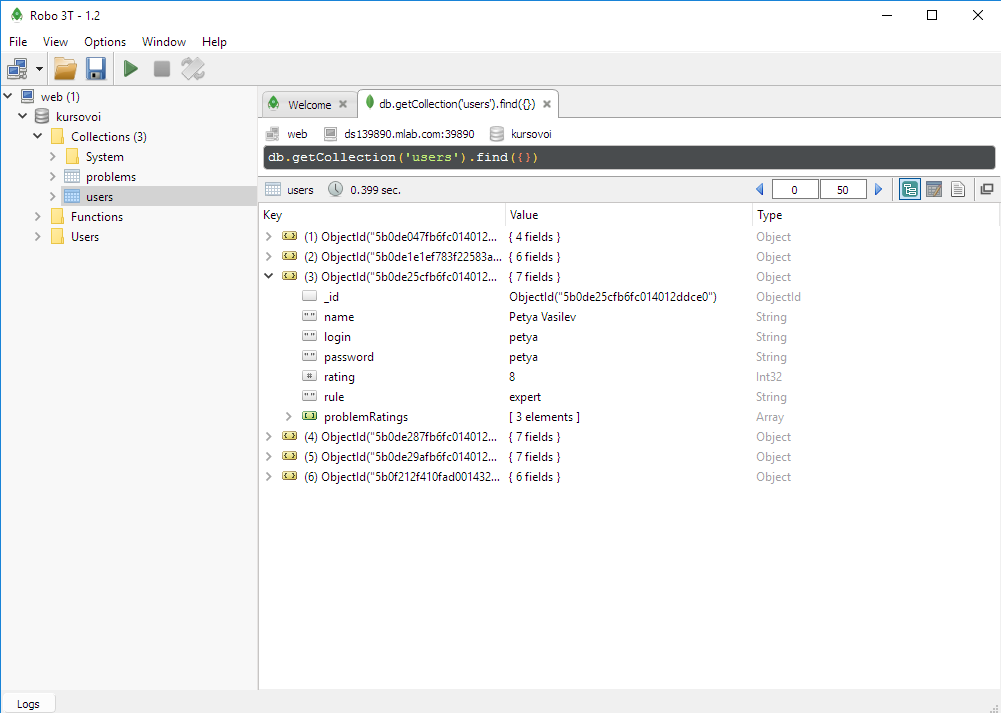


Рисунок 7.4 – Окно программы Robo3T

В приложении реализованы две пользовательские роли: эксперт, пользователь. При авторизации с данными эксперта, пользователь попадает на страницу /expert, где предложен список проблем и требуется выбрать одну для оценивания (рисунок 7.5).

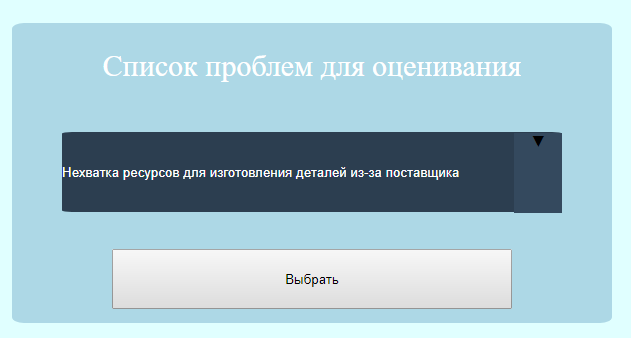


Рисунок 7.5 – Выбор проблемы для оценивания

После выбора проблемы, эксперту предлагается оценить возможные решения проблемы, путем выставления оценок от 0 до 10 включительно. Рисунок 7.6 демонстрирует окно выставления оценок для проблемы.

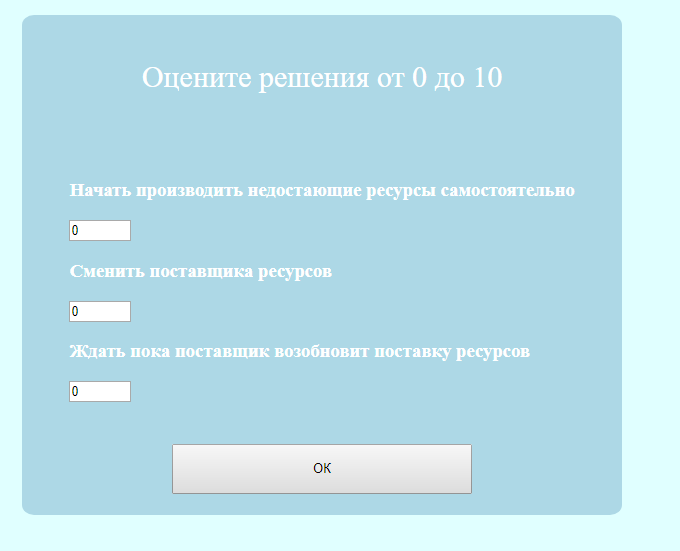


Рисунок 7.6 – Окно оценивания решений

После оценки решений, эксперту выводится окно с кнопкой перехода на главную страницу. Все данные введенные экспертом заносятся в базу данных.

Рисунок 7.7 демонстрирует окно с кнопкой перехода на главную страницу сайта.

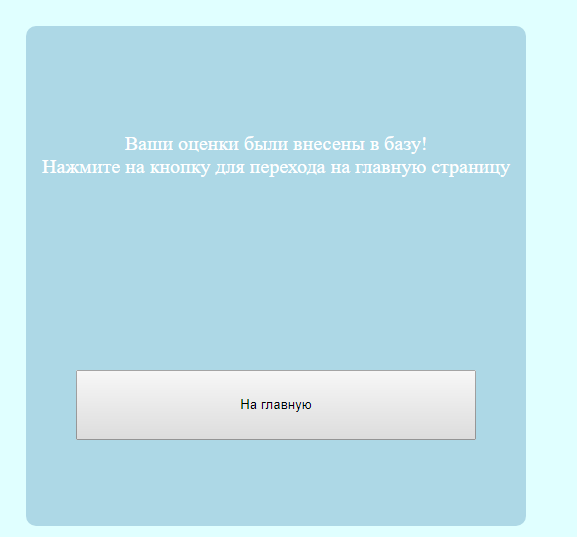


Рисунок 7.7 – Окно завершения работы с экспертом

Если же пользователь авторизовался с ролью «user», то сайт перенаправляет его на страницу /user. Перед пользователем появляется окно с выбором проблемы, решение которой он хочет увидеть (рисунок 7.8).

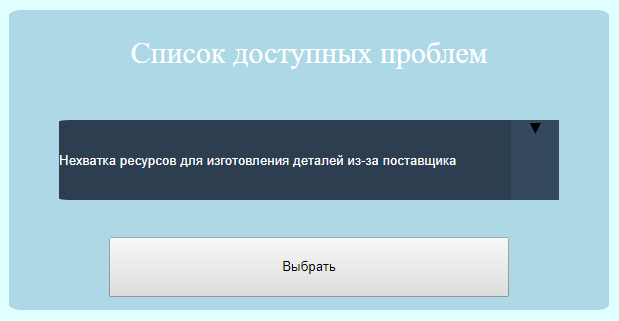


Рисунок 7.8 – Выбор проблемы пользователем

После выбора проблемы, пользователю предоставляется выбор из экспертов уже оценивших эту проблему. Путем нажатия на кнопки с именами экспертов, пользователь выбирает их для дальнейшего решения задачи. Рисунок 7.9 демонстрирует выбор доступных экспертов.



Рисунок 7.9 – Выбор доступных экспертов

После выбора экспертов, пользователю составляется таблица из решений экспертов, а также выводится наилучшее решение из результатов оценивания (рисунок 7.10). Также пользователю доступна кнопка «Результаты оценки проблемы», где после нажатия её нажатия строится круговая диаграмма, для отображения итоговых баллов набранные решениями в процентах (рисунок 7.11).

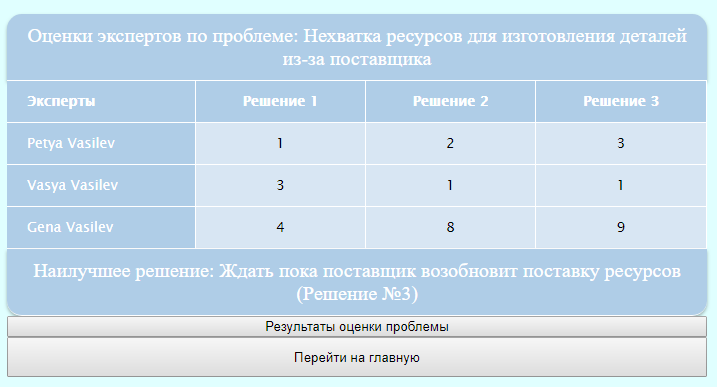


Рисунок 7.10 – Таблица решений проблемы

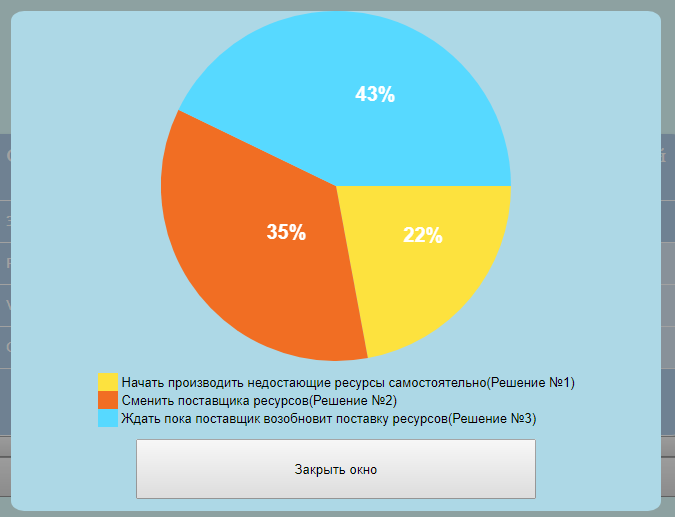


Рисунок 7.11 – Круговая диаграмма решений проблемы

# **8 КОНТРОЛЬНЫЙ ПРИМЕР**

В данном разделе мы рассмотрим решения задачи методом взвешенных экспертных оценок.

Исходные данные введены следующим образом: на предприятии работает 3 эксперта: Э1, Э2, Э3, которые оценивают продукцию и проблемы. Были созданы 3 решения одной проблемы: Z1, Z2, Z3. Рейтинги экспертов соответственно равны: 8, 5, 6.

Рассмотрим решение задачи в Excel.

Матрица оценок экспертов представлена на рисунке 8.1

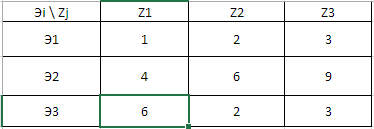


Рисунок 8.1 – Матрица оценок экспертов

Вычислим относительные рейтинги экспертов: R1, R2, R3 (рисунок 8.2).

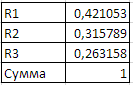


Рисунок 8.2 – Относительные рейтинги экспертов

Вычислим наилучшее решение проблемы (рисунок 8.3).

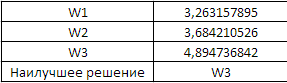


Рисунок 8.3 – Решение проблемы

Результатом решения проблемы является решение Z3.

# **ЗАКЛЮЧЕНИЕ**

При разработке программного приложения мною были исследованы и изучены различные подходы к производству деталей моей предметной области, а именно, производства автомобильных запчастей. Также был изучен и применен один из экспертных методов - метод взвешенных экспертных оценок.

В результате мною был сделан вывод о необходимости использования автоматизированной системы при поиске наилучшего варианта, который бы позволил произвести деталь, решить проблему с наименьшими затратами для предприятия и в будущем получить большую выручку. Данный метод позволяет решать достаточно сложные задачи, которые серьезно могут повлиять на будущее того или иного производства.

Результатом работы является программа, написанная на языке JavaScript. Данная программа обеспечивает надежность и защищенность данных, т.к. все сведения как о пользователях, так и о продукции, изготавливаемой предприятием, хранятся в базе данных, пароли зашифрованы, что не допускает несанкционированного доступа в систему. Бизнес-логика, которая применена, находится на сервере. Также были построены диаграмма IDEF0 и UML-диаграммы.

К достоинствам разработанного мною приложения можно отнести простой и понятный интерфейс, также наличие подсказок, которые будут полезны как умелым пользователям ПК, так и неопытным.

Поставленные задачи решены и цели достигнуты.

В процессе использования есть возможность установления «слабых мест», которые могут быть усовершенствованы в последующих обновлениях программного приложения.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

[1] Wikipedia [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/>.

[2] Яковлев А.О., Сторожев Д.А. Методическое пособие. Язык UML. Диаграммы UML. Минск – 2012.

[3] Е. Н. Живицкая, О. П. Едемская. «Системный анализ и проектирование информационных систем» уч. пособие. Минск БГУИР – 2005.

[4] С. С. Смородинский, Н. В. Батин. Системный анализ и исследование операций. Лабораторный практирум. Минск БГУИР – 2009.

[5] Habr [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habr.com

[6] stackoverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.stackoverflow.com/>

[7] METANIT [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/>

[8] Доманов, А.Т. Предварительный стандарт предприятия. Дипломные проекты(работы): общие требования / А.Т. Доманов, Н.И. Сороко. – Мн.: БГУИР, 2009.

# **ПРИЛОЖЕНИЕ A**

# **(обязательное)**

# **Диаграммы UML (к разделу 2)**

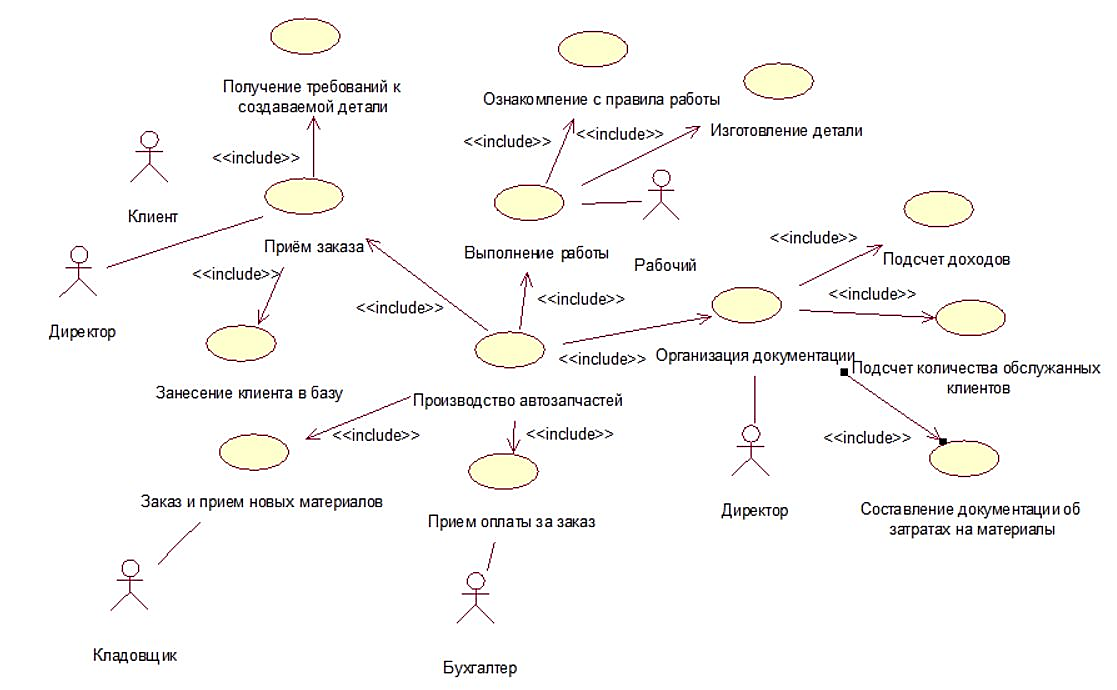
****

Рисунок А.1 – Диаграмма вариантов использования

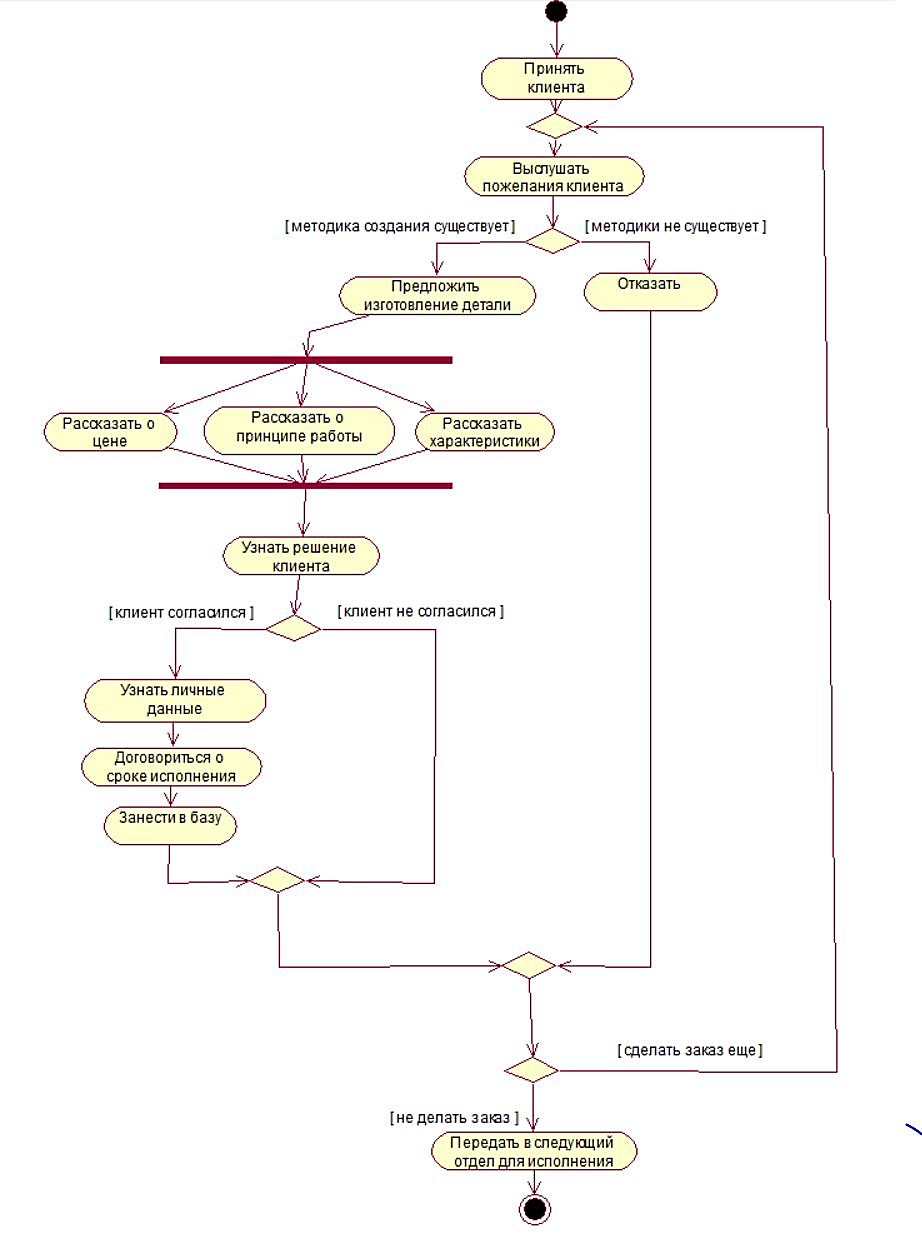


Рисунок А.2 – Диаграмма деятельности

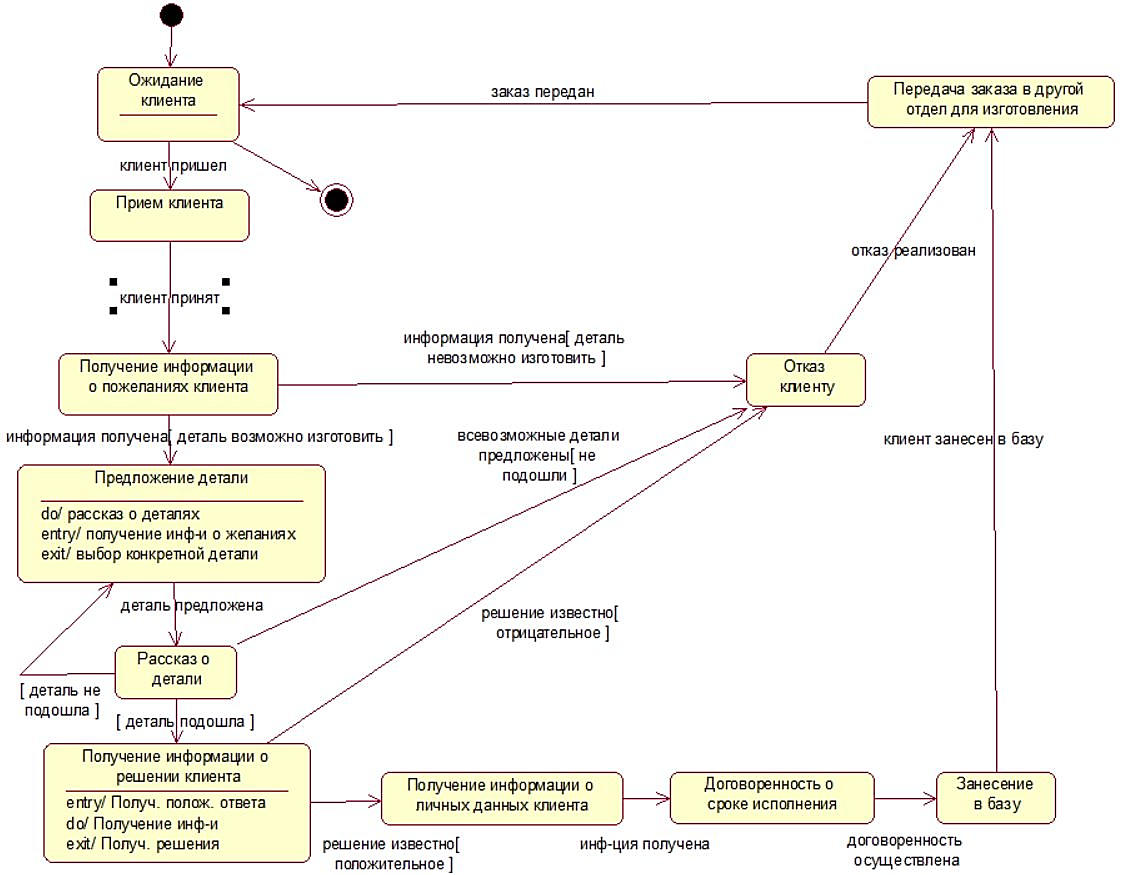
****

Рисунок А.3 – Диаграмма состояний

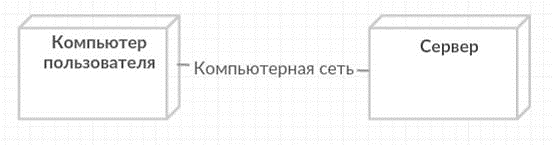


Рисунок А.4 – Диаграмма развертывания

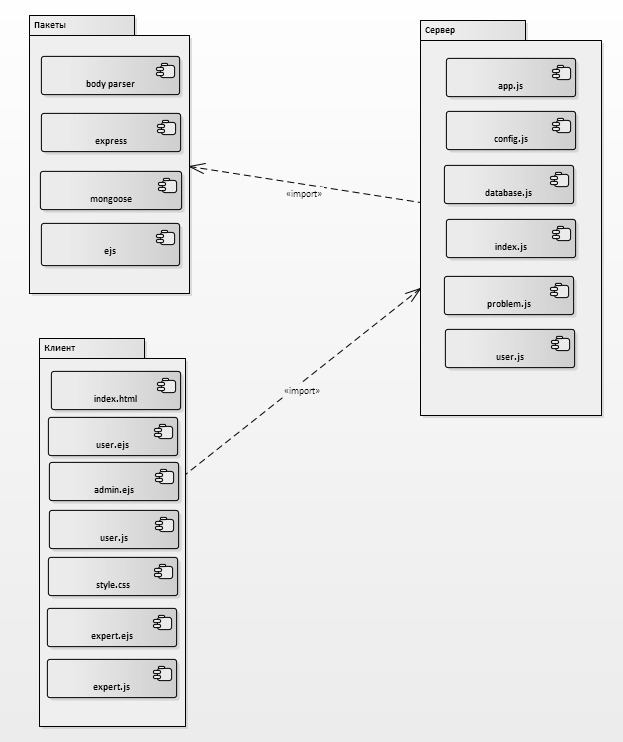


Рисунок А.5 – Диаграмма компонентов

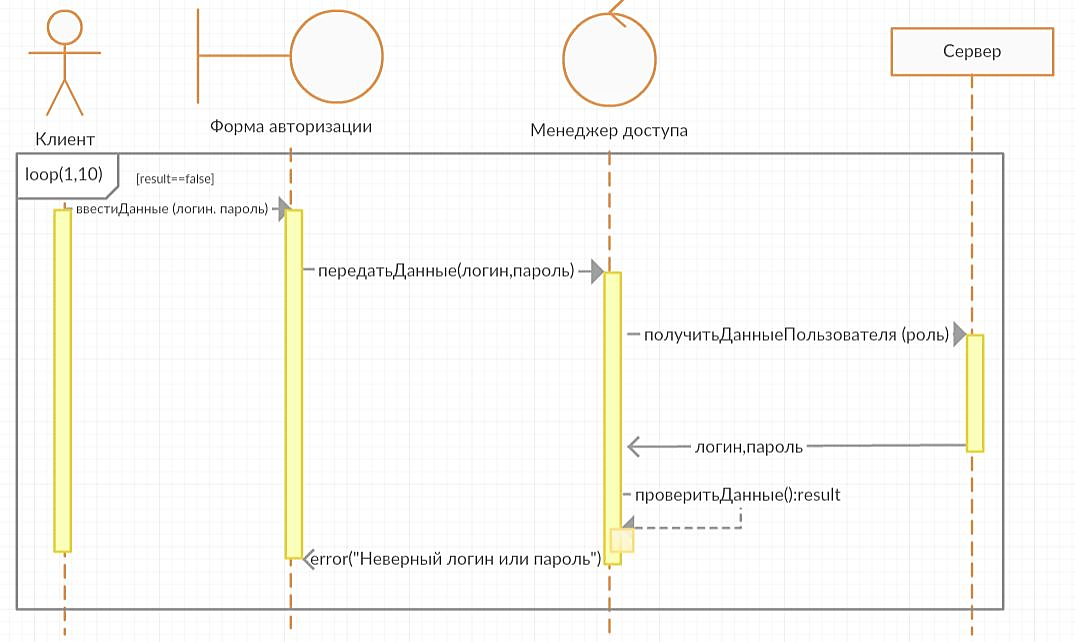


Рисунок А.6 – Диаграмма последовательности

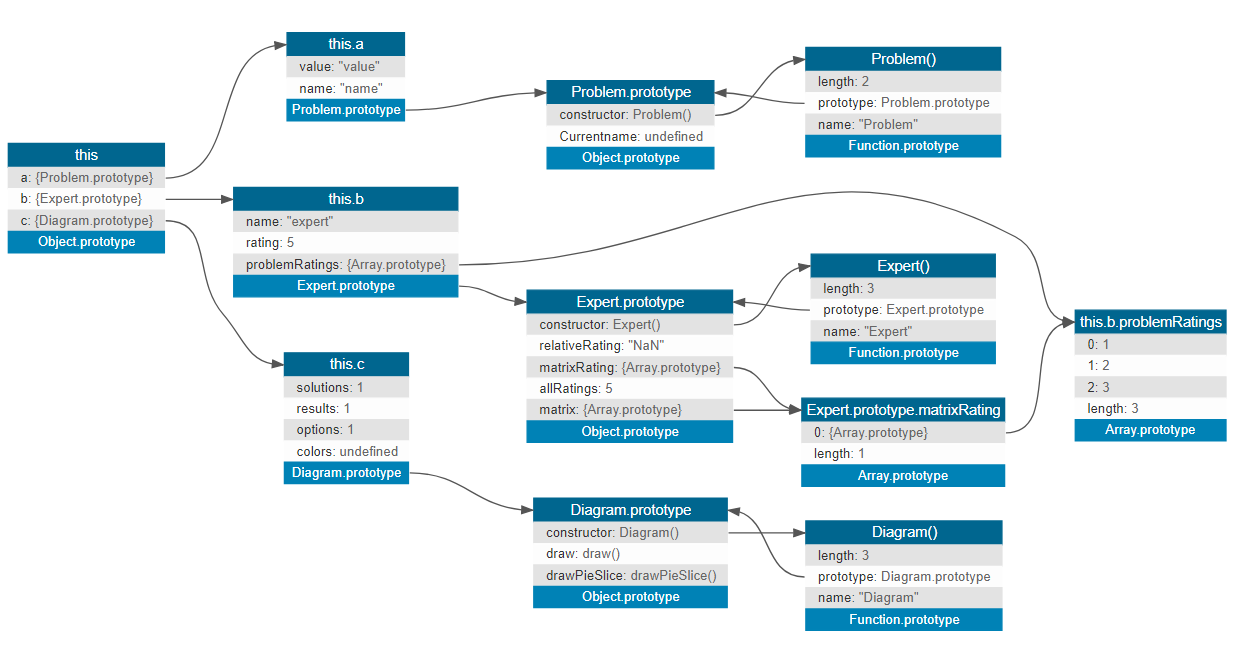


Рисунок А.7 – Диаграмма классов

# **ПРИЛОЖЕНИЕ Б**

# **(обязательное)**

# **Листинг кода**

// user.js

const selectProblem = document.getElementById("selectProblem");

const chooseExperts = document.getElementById("chooseExperts");

let selectedIndex;

let usersArr;

let problemsArr;

function getArrays(problems, users) {

problemsArr = problems;

usersArr = users;

}

selectProblem.addEventListener("click", event=>{

const target = event.target;

const select = document.getElementById("select");

if(target.id === "submit"){

selectedIndex = select.options.selectedIndex;

selectProblem.style.display = "none";

chooseExperts.style.display = "grid";

addExperts();

}

});

function addExperts() {

const expertsList = document.getElementById("expertsList");

usersArr.forEach(user => {

if(user.problemRatings.some(problem => {

return problem.name === problemsArr[selectedIndex].name;

})){

const li = document.createElement("li");

const p = document.createElement("p");

p.innerHTML = user.name;

li.appendChild(p);

li.classList = "expert";

expertsList.appendChild(li);

}

});

}

const choosenExperts = document.getElementsByClassName("choose");

const tableDiv = document.getElementById("tableDiv");

let results;

chooseExperts.addEventListener("click", event => {

const target = event.target;

const mainSolution = document.getElementById("mainSolution");

const caption = document.getElementById("problemName");

console.log(target);

if(target.classList.contains("expert")){

target.classList.toggle("choose");

}

if(target.parentNode.classList.contains("expert")){

target.parentNode.classList.toggle("choose");

}

if(target.id === "submit" && choosenExperts.length) {

chooseExperts.style.display = "none";

caption.innerHTML = 'Оценки экспертов по проблеме: ' + problemsArr[selectedIndex].name;

tableDiv.style.display = "grid";

generateTable();

results = getResults();

mainSolution.innerHTML = `

Наилучшее решение: ${problemsArr[selectedIndex].solutions[results.indexOf(Math.max(...results))].name} (Решение №${results.indexOf(Math.max(...results))+1})

`

}

});

class Expert {

constructor(name, rating, problemRatings) {

this.name = name;

this.rating = rating;

this.problemRatings = problemRatings;

Expert.prototype.allRatings += rating;

Expert.prototype.matrix.push(problemRatings);

}

get relativeRating () {

return (this.rating / Expert.prototype.allRatings).toFixed(2);

}

get matrixRating() {

return Expert.prototype.matrix;

}

}

Expert.prototype.allRatings = 0;

Expert.prototype.matrix = [];

const expertsArr = [];

function generateTable() {

const table = document.getElementById("table");

const solutionsHeader = document.createElement("tr");

const th = document.createElement("th");

th.innerHTML = "Эксперты";

solutionsHeader.appendChild(th);

for(let i = 0; i < problemsArr[selectedIndex].solutions.length; i++){

const th = document.createElement("th");

th.innerHTML = `Решение ${i+1}`;

solutionsHeader.appendChild(th);

}

table.appendChild(solutionsHeader);

for(let i = 0; i < choosenExperts.length; i++) {

const tr = document.createElement("tr");

const nameExpert = document.createElement("td");

nameExpert.innerHTML = choosenExperts[i].firstElementChild.innerHTML;

tr.appendChild(nameExpert);

let expertIndex;

let problemIndex;

for(let j = 0; j < problemsArr[selectedIndex].solutions.length; j++) {

const rate = document.createElement("td");

for(let x = 0; x < usersArr.length; x++){

if(usersArr[x].name === nameExpert.innerHTML) {

expertIndex = x;

break;

}

}

for(let x = 0; x < usersArr[expertIndex].problemRatings.length; x++){

if(usersArr[expertIndex].problemRatings[x].name === problemsArr[selectedIndex].name){

problemIndex = x;

break;

}

}

rate.innerHTML = usersArr[expertIndex].problemRatings[problemIndex].rating[j];

tr.appendChild(rate);

}

expertsArr.push(new Expert(usersArr[expertIndex].name, usersArr[expertIndex].rating, usersArr[expertIndex].problemRatings[problemIndex].rating));

table.appendChild(tr);

}

}

tableDiv.addEventListener("click", event => {

const target = event.target;

const resultsDiv = document.getElementById("results");

const diagramCanvas = document.getElementById("diagram");

diagramCanvas.width = 350;

diagramCanvas.height = 350;

const legend = document.getElementById("legend");

if(target.id === "resultsButton") {

resultsDiv.style.display = "grid";

const diagram = new Diagram(

problemsArr[selectedIndex].solutions,

results,

{

canvas:diagramCanvas,

colors:["#fde23e","#f16e23", "#57d9ff","#937e88"],

legend: legend,

}

)

diagram.draw();

resultsDiv.addEventListener("click", event => {

if(event.target.id === "submit"){

resultsDiv.style.display = "none";

}

})

}

})

function getResults() {

const localResults = new Array(problemsArr[selectedIndex].solutions.length).fill(0);

for(let i = 0; i< problemsArr[selectedIndex].solutions.length; i++) {

for(let j = 0; j < expertsArr[0].matrixRating.length; j++) {

localResults[i] += parseInt(expertsArr[0].matrixRating[j][i]) \* parseFloat(expertsArr[j].relativeRating);

}

}

return localResults;

}

class Diagram {

constructor(solutions, results,options){

this.solutions = solutions;

this.results = results;

this.options = options;

this.canvas = options.canvas;

this.ctx = this.canvas.getContext("2d");

this.colors = options.colors;

}

draw() {

let totalValue = results.reduce((a,b)=>{return a+b}, 0);

let colorIndex = 0;

let startAngle = 0;

this.results.forEach(item =>{

let sliceAngle = 2 \* Math.PI \* item / totalValue;

let pieRadius = Math.min(this.canvas.width/2,this.canvas.height/2);

let labelX = this.canvas.width/2 + (pieRadius / 2) \* Math.cos(startAngle + sliceAngle/2);

let labelY = this.canvas.height/2 + (pieRadius / 2) \* Math.sin(startAngle + sliceAngle/2);

this.drawPieSlice(

this.ctx,

this.canvas.width/2,

this.canvas.height/2,

Math.min(this.canvas.width/2, this.canvas.height/2),

startAngle,

startAngle+sliceAngle,

this.colors[colorIndex%this.colors.length]

);

colorIndex++;

let labelText = Math.round(100 \* item / totalValue);

this.ctx.fillStyle = "white";

this.ctx.font = "bold 20px Arial";

this.ctx.fillText(labelText+"%", labelX,labelY);

startAngle +=sliceAngle;

})

colorIndex = 0;

let legendHTML = "";

this.solutions.forEach(item =>{

legendHTML += "<div><span style='display:inline-block;width:20px;background-color:"+this.colors[colorIndex++]+";'>&nbsp;</span> "+"<p>"+item.name+ "(Решение №"+ colorIndex+ ")"+"</p>"+"</div>";

});

this.options.legend.innerHTML = legendHTML;

}

drawPieSlice(ctx, centerX, centerY, radius, startAngle, endAngle, color) {

ctx.fillStyle = color;

ctx.beginPath();

ctx.moveTo(centerX, centerY);

ctx.arc(centerX, centerY, radius, startAngle, endAngle);

ctx.closePath();

ctx.fill();

}

}

class Problem {

constructor(name, value){

this.value = value;

this.name = name;

}

get name() {

return this.name;

}

}

// expert.js

const selectProblem = document.getElementById("selectProblem");

let selectedIndex = 0;

let problemsArr;

let currentExpert;

function getProblems(problems, expert){

console.log(problems, expert);

currentExpert = expert;

problemsArr = problems;

}

const solutions = document.getElementById("solutions");

const rateSolution = document.getElementById("rateSolution");

selectProblem.addEventListener("click", (event)=>{

const target = event.target;

const select = document.getElementById("select");

if(target.id === "submit"){

selectedIndex = select.options.selectedIndex;

selectProblem.style.display = "none";

rateSolution.style.display = "grid";

addSolutions();

}

});

function addSolutions(){

for(let i = 0; i< problemsArr[selectedIndex].solutions.length; i++){

const solution = document.createElement("div");

solution.classList = "solution";

const inputRating = document.createElement("input");

inputRating.type = "number";

inputRating.min = "0";

inputRating.max = "10";

inputRating.value = "0";

const title = document.createElement("h3");

title.innerHTML = problemsArr[selectedIndex].solutions[i].name;

solution.appendChild(title);

solution.appendChild(inputRating);

solutions.appendChild(solution);

}

}

const endWork = document.getElementById("endWork");

const expertRating = [];

rateSolution.addEventListener("click", (event)=>{

const inputRating = Array.from(document.querySelectorAll("input"));

const target = event.target;

if(target.id === "submit"){

if(!inputRating.some(elem =>{

return elem.value < 0 || elem.value > 10;

})

){

inputRating.forEach(item =>{

expertRating.push(item.value);

});

rateSolution.style.display = "none";

endWork.style.display = "grid";

let problem = new Problem(problemsArr[selectedIndex].name, expertRating);

for(let i = 0; i< currentExpert.problemRatings.length; i++) {

if(currentExpert.problemRatings[i].name === problem.name){

currentExpert.problemRatings.splice(i,1);

break;

}

}

currentExpert.problemRatings.push(problem);

const xhr = new XMLHttpRequest();

currentExpert = JSON.stringify(currentExpert);

xhr.open("POST", "expert");

xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

xhr.send(currentExpert);

}

}

})

class Problem{

constructor(nameProblem, rating){

this.name = nameProblem;

this.rating = rating;

}

}

//admin.js

const userList = document.getElementById("userList");

let usersArr;

function getUsers(users){

usersArr = users;

}

userList.addEventListener("click", (event) => {

const target = event.target;

if(target.classList.contains("delete")){

const login = target.nextElementSibling.textContent;

const xhr = new XMLHttpRequest();

xhr.open("POST", "admin");

xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

xhr.send(JSON.stringify({login: login}));

target.parentNode.style.display = "none";

}

})

//app.js

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const user = require('./models/user');

const problem = require('./models/problem');

const app = express();

app.set("view engine","ejs");

app.use(bodyParser.urlencoded({extended: true}));

app.use(express.static(path.join(\_\_dirname, 'public')));

app.get('/form', (req, res) => res.render('form'));

let expert; // идентификационная переменная (вместо cookie)

let admin;

app.get('/registration', (req, res) => res.render('registration'));

app.get('/admin', (req, res) => {

if(!admin) res.render('403');

user.find({})

.then(data=>{

res.render('admin', {users: data});

})

});

app.get('/user', (req, res) => {

problem.find({})

.then(problems =>{

user.find({rule:"expert"})

.then(users =>{

res.render('user', {users:users, problems: problems});

})

})

});

app.get('/expert', (req, res) => {

if(!expert) res.render('403');

problem.find({})

.then(problems=>{

res.render('expert', {problems: problems, expert:expert});

});

});

app.post('/registration', (req,res) =>{

const {login, password} = req.body;

user.findOne({login: new RegExp('^'+login+'$', "i")}, function(err,User){

if(!User) {

user.create({

login: login,

password: password,

rule: "user"

});

res.redirect('/form');

}

else{

res.sendStatus(400);

}

});

})

app.post('/form', (req,res) =>{

const {login, password} = req.body;

user.findOne({

login: new RegExp('^'+login+'$', "i"),

password: new RegExp('^'+password+'$', "i")

}, function(err, User){

if(User && User.rule === "admin"){

admin = JSON.stringify(User);

res.redirect('/admin');

} else if(User && User.rule === "user"){

res.redirect('/user');

} else if(User && User.rule === "expert"){

expert = JSON.stringify(User);

res.redirect("/expert");

}

else {

res.sendStatus(400);

}

})

});

app.post('/expert', bodyParser.json(), function(req, res){

user.findOneAndUpdate(JSON.parse(expert), req.body, function(err){

if(err) res.sendStatus(500);

return res.sendStatus(200);

});

})

app.get('/', function (req, res) {

res.render('index');

})

app.post('/admin', bodyParser.json(), function(req, res){

user.findOneAndRemove(req.body, function(err){

if(err) res.sendStatus(500);

});

})

app.use(function(req, res,next) {

res.status(404).render("404");

});

module.exports = app;