МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка алгоритма распределения задач между сотрудниками

Обучающийся А.В. Чепаксина
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема данной выпускной квалификационной работы: «Разработка алгоритма распределения задач между сотрудниками».

Целью работы является разработка и реализация модели эффективного распределения задач между сотрудниками.

Данная выпускная квалификационная работа состоит из: введения, трёх основных глав, заключения, а также списка используемой литературы и задействованных источников.

Введение раскрывает актуальность темы, поставленную цель и задачи для реализации автоматического распределения задач между сотрудниками.

Первая глава содержит описание математической модели распределения задач между сотрудниками, обзора существующих методов решения и выбор оптимального из них.

Вторая глава содержит описание разработки системы распределения задач между сотрудниками, описание проектируемой базы данных и выбора СУБД.

Третья глава представляет собой программную реализацию системы распределения задач между сотрудниками, графического интерфейса и описание тестирования.

Заключение содержит выводы, которые были сделаны в результате проведенной работы.

Результатом работы является программа с графическим интерфейсом пользователя для автоматизированного распределения задач между сотрудниками.

В работе было использовано 23 рисунка и 25 ссылок на внешние ресурсы. Общий объем выпускной квалификационной работы 44 страницы.

Abstract

The title of the graduation work is Development of an algorithm for distributing tasks between employees.

The senior paper consists of an introduction, three parts, a conclusion, list of references including foreign sources.

The key issue of the graduation work is to automate the process of distributing tasks between employees.

We touch upon the problem of inefficient and time-consuming distribution of tasks between employees in the organization.

The purpose of the work is to develop and implement a model for the effective distribution of tasks between employees.

The graduation work may be divided into several logically connected parts which are: problem statement, description of a mathematical model for the distribution of tasks between employees, analysis and selection of the best solution method; development of an information system by drawing up a conceptual model of the system and a logical database model; describing the operation of the system using a flowchart, creating a physical data model, implementing the program interface and testing it.

Finally, we present a developed application that increases the speed and efficiency of task distribution between employees in large companies. In this way, it reduces the workload on HR managers, employees and supervisors.

In conclusion we'd like to stress the information system for distributing tasks between employees allows you to increase the efficiency and productivity of the organization by providing a convenient application for the HR manager with a graphical interface.

Оглавление

Введение

Распределение задач между сотрудниками является важным аспектом любой компании. Для каждой поставленной задачи необходимы грамотные и квалифицированные специалисты. Во многих компаниях работает огромное количество человек, из-за чего происходят трудности в распределении задач между ними.

Из-за неэффективного распределения, сотрудники вынуждены концентрироваться на менее главных задачах и соответственно, не могут полностью реализовать свой потенциал и принести максимальную пользу компании. Именно в таком случае на помощь приходит делегирование.

Распределение задач нужно для основных целей:

- снижение нагрузки на других сотрудников;
- снижение нагрузки для руководителей;
- повышение эффективности работы каждого сотрудника;
- повышение стимуляции сотрудника к работе.

Актуальность бакалаврской работы обусловлена тем, что она позволяет рассмотреть различные аспекты применения математического моделирования к реальным проблемам в компании. Кроме этого, тема является актуальной с точки зрения использования современных технологий, вместо ручного труда.

Целью данной работы является разработка и реализация модели эффективного распределения задач между сотрудниками.

Для достижения выше поставленной цели, необходимо решить следующие задачи:

- формализовать постановленную задачу о назначении;
- разработать модель распределения задач между сотрудниками;
- проанализировать существующие подходы решения задачи о назначении;
- разработать требования к программе для распределения задач;

– реализовать программу процесса распределения задач между сотрудниками;
– провести тестирование программы;
– сделать вывод о применимости получившейся модели.

Первая глава работы посвящена формализации проблемы распределения задач между сотрудниками, построению математической модели, разбору существующих методов, их анализу и определению наиболее подходящего для решения поставленной задачи.

Вторая глава описывает требования к разрабатываемой системе, разработку концептуальной модели, логическое описание базы данных и выбор СУБД.

В третьей главе проводится описание работы системы распределения задач между сотрудниками, создание физической модели данных, разработка интерфейса и тестирование системы.

Результатом работы является реализованная модель распределения задач между сотрудниками в виде приложения с графическим интерфейсом.

Глава 1 Определение подхода для оптимального распределения задач между сотрудниками

1.1 Описание математической модели распределения задач между сотрудниками

Во многих компаниях существует проблема эффективного и быстрого распределения задач между сотрудниками. Делегирование – это и есть комплекс мероприятий, направленный на максимально оптимальное распределение задач между сотрудниками. Проблема делегирования связана с большим количеством работников на предприятии, в следствии чего возникают две глобальные проблемы:

- сложность оптимального распределения;
- большие затраты по времени.

В связи с этим, решать подобные задачи в «ручную» является достаточно трудоемким и неэффективным процессом.

Для решение данной проблемы необходимо разработать подход, позволяющий компании получить максимальную прибыль и при этом, потратить минимум времени на распределение поставленных задач. Исходя из этого, необходимо произвести постановку задачи для решения существующей проблемы.

Постановка задачи: перед руководителем стоит n видов задач, и в его распоряжении имеются m сотрудников. Таким образом, параметры m и n являются натурными числами, причем выполняется неравенство $m \geq n$. На каждую из задач должен быть определен минимум один сотрудник. Распределение сотрудников должно быть максимально эффективным для выполнения работ.

Кроме четко поставленной задачи, необходимо определиться с показателями, по которым будет распределение. Для любого типа рабочей деятельности представлен набор компетенций; каждой компетенции дается

количественная или качественная оценка. Разнообразные задачи компании имеют собственные наборы показателей эффективности, но можно выделить базовые показатели, которые являются универсальными для любой задачи и на которые можно ориентироваться [20]. Такими показателями, например, являются стаж, ответственность, образование и другие. Во всех крупных компаниях данные показатели называются цифровым профилем сотрудника и содержатся в базе данных предприятия, контроль и управление над которыми осуществляет менеджер по персоналу.

Для решения задачи о назначении необходимо составить математическую модель. Для этого требуется решить три аспекта:

- формализовать основные показатели сотрудников для решения производственных задач [19];
- вычислить коэффициенты эффективности выполнения сотрудниками каждой из задач [5];
- составить матрицу коэффициентов эффективности.

Для нахождения коэффициентов эффективности необходима экспертная оценка показателей каждого сотрудника. Показатели бывают двух видов, имеющие числовое значение, например, должность, стаж, возраст и имеющие категориальное значение, например, личностные характеристики или работа с коллективом, которые будут оцениваться значениями от 0 до 10 [6].

Данные показатели необходимо нормализовать. Для этого разделим показатель на диапазон значений

$$a_{i,k} = \frac{r_k - r_{min}}{r_{max} - r_{min}}, \quad (1)$$

где $a_{i,k}$ – нормированный k-й показатель i-го сотрудника ($i = \overline{1,m}$);

r_{k} – конкретное значение показателя;

r_{min} – минимальное значение показателей среди всех работников;

r_{max} – максимальное значение показателей среди всех сотрудников.

Последним этапом нахождения коэффициента эффективности будет сумма произведений нормированных показателей сотрудника на коэффициент полезности этого показателя для той или иной задачи.

$$c_{i,j} = \sum_{k=1}^{K} a_{i,k} * p_{j,k}, \quad (2)$$

где $c_{i,j}$ – коэффициент эффективности;

K – количество показателей;

$p_{\mathrm{i,k}}$ – коэффициент полезности k-го показателя j-й задачи.

Таким образом, мы получаем матрицу коэффициентов эффективности

$$\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{pmatrix}. \quad (3)$$

В этой матрице по строкам распределены сотрудники, а по столбцам задачи.

Для нахождения оптимального решения распределения задач между сотрудниками необходимо решить матрицу коэффициентов эффективности. Итоговым решением будет являться набор сотрудников и подходящих для них задач.

1.2 Методы решения задачи о назначении

1.2.1 Метод потенциалов

Основной идеей метода потенциалов является преобразование первичного плана распределения в наиболее выгодный, до тех пор, пока не найдется более оптимальный план [3].

Алгоритм решения задачи о назначении с помощью метода потенциалов.

Шаг 1. Достраивается матрица коэффициентов эффективности с добавлением столбца запасов и строки потребностей.

Шаг 2. Проверяется задача на закрытость. Для этого суммируются значения в столбце запасов и значения в строке потребностей. Если сумма запасов равна сумме потребностей, то задача закрытого типа, иначе открытого.

Шаг 3. Составляется опорный план распределения. Находятся максимальные элементы в матрице и вычеркивается соответствующая строка и столбец, таким образом, получается первичный план.

Шаг 4. Составляется для каждого сотрудника и задачи величины U_i и V_j так, чтобы для базисного решения выполнялось соотношение

$$c_{i,j} = U_i + V_j. \quad (4)$$

Шаг 5. Проверяется план на оптимальность, для этого вычислим разности

$$\Delta c_{i,j} = c_{i,j} - (U_i + V_j). \quad (5)$$

План оптимален, если все разности не являются отрицательными.

Шаг 6. Если план не оптимален, то для самой минимальной $\Delta c_{i,j}$ строится цикл, в котором кроме этой клетки все остальные являются базисными, перераспределяются значения и происходит переход к шагу 4. В противном случае найдено оптимальное решение.

Базисное решение, полученное после 6 шага, будет неконечным решением задачи о назначении. Для нахождения оптимума необходимо в незадействованных строках найти максимальное значение. В результате будет найдено конечное решение.

1.2.2 Венгерский алгоритм

Венгерский алгоритм подходит как для линейных, так и для нелинейных, но с некоторыми модификациями.

Линейный алгоритм используется при равенстве количества задач и сотрудников. В нашем случае количество сотрудников превосходит количество задач, поэтому необходимо использовать нелинейный Венгерский алгоритм.

Шаг 1. В каждой строке ищется максимальный элемент. Затем, он вычитается из каждого элемента строки [2].

Шаг 2. Следующим действием вся матрица умножается на минус 1 [10].

Шаг 3. В каждом столбце ищется минимальный элемент и из него же вычитается [4].

Шаг 4. Если в каждом столбце есть как минимум по одному нулю, то решение найдено, иначе шаг 5.

Шаг 5. Необходимо провести редукцию матрицы. Для этого, в матрице вычёркиваются столбцы и строчки содержащие нули, среди не тронутых ячеек ищем минимальный элемент и вычитаем его из этих ячеек, а к ячейкам, расположенным на пересечении вычеркнутых столбцов, прибавляем это минимальное значение.

Шаг 6. Снова проверяем получился ли хотя бы один ноль в каждом столбце. Если да, то решение найдено, иначе повторяем шаг 5.

Таким образом, после завершения алгоритма необходимо найти решение методом проб и ошибок. Данный метод заключается в переборе всех альтернативных решений. Так как, результат альтернативных решений одинаков, то в рамках компании достаточно найти хотя бы один. Для этого выберем в конечной матрице первые нули в каждом столбце и любой ноль в каждой строке, при условии ограничения:

- в каждой строке может быть выбран только один ноль;
- в каждом столбце должен быть выбран хотя бы один ноль [17].

1.3 Анализ существующих подходов для эффективного распределения и выбор алгоритма для реализации

Существуют различные подходы для эффективного распределения. Одним из таких является метод потенциалов. Данный подход представляет собой модифицированный симплекс-метод решения транспортных задач и о назначении. У метода присутствуют свои достоинства:

- является достаточно простым и понятным;
- является точным;
- возможно использование для нелинейных задач.

Недостатками данного метода являются: большая чувствительность к изменениям в условиях задачи, трудоемкость при большом количестве параметров. Поэтому метод потенциалов является не самым оптимальным для реализации в большой компании.

Следующим рассматриваемым подходом является Венгерский алгоритм. Данный метод также используется для решения задач оптимизации [**Ошибка! Источник ссылки не найден.**]. Венгерский алгоритм имеет ряд д остоинств:

- обеспечивает достаточно точное решение;
- универсальность, применение во многих областях;
- эффективный алгоритм при больших изначальных данных;
- решение линейных и нелинейных задач [8].

Главным недостатком данного метода является получение множественного или альтернативного решения [12]. Данный недостаток не влияет на правильность или точность найденного решения.

Из всех рассмотренных подходов, в силу своей универсальности и оптимальности, больше всего подойдет Венгерский алгоритм. Данный алгоритм не является трудоемким, что позволяет быстрее решить поставленную задачу и он отлично подойдет для решения нелинейных задач.

Несмотря на то, что выбранный метод решения является средним с точки зрения трудоемкости, но при большом количестве сотрудников и задач, расчет и решение матрицы коэффициентов эффективности становится длительным процессом при ручном решении. Для быстрого и эффективного решения задач в компании, необходимо автоматизировать процесс распределения задач между сотрудниками.

Выводы к главе 1

В рамках данного этапа была выявлена проблема эффективного и быстрого распределения задач между сотрудниками. Для этого была произведена постановка задачи о назначении, по итогам которой определено, что для расчета оптимальности назначения сотрудников, необходимо выделить основные показатели для каждой из поставленных задач. Данные показатели будут взяты из цифрового профиля сотрудника компании.

Перед решением задачи была составлена математическая модель распределения задач между сотрудниками, проведена нормализация показателей сотрудников и на их основе составлена формула расчетов коэффициентов эффективности для построения матрицы эффективности.

Кроме этого, был проведен анализ существующих подходов для эффективного распределения, и выбран наилучший из них, а именно Венгерский алгоритм.

Основываясь на трудоемкости необходимых вычислений, выявлено, что для автоматизации процесса распределения задач между сотрудниками необходимо разработать соответствующий программный модуль.

Глава 2 Разработка информационной системы оптимального распределения задач между сотрудниками

2.1 Определение функциональных требований к разрабатываемой системе

Для определения требований к программному обеспечению были проанализированы пожелания менеджера по персоналу, существующие программы в компании и произведено моделирование системы.

От менеджера были выделены следующие требования к программе:

- возможность авторизации в приложении менеджера;
- возможность создания задачи с учетом важности различных критериев, навыков сотрудников;
- просмотр созданных задач;
- удаление ранее созданных задач с базы данных;
- просмотр всех сотрудников в компании;
- меню с руководством пользователя в приложении;
- возможность перехода между страницами;
- информация об авторе приложения;
- эффективное распределение задач между сотрудниками.

После определения требований к разрабатываемой системе, необходимо выбрать методологию для дальнейшего проектирования. Для наглядности отлично подойдет методология UML.

UML представляет собой унифицированный язык моделирования. Это графический язык, в котором каждой фигуре, символе, стрелке присвоены конкретные значения [7].

В данной работе необходимо разработать алгоритм, модель и провести программную реализацию для компании. Именно благодаря языку моделирования UML можно достаточно просто объяснить работу сложных

систем. Поэтому данный язык является достаточно универсальным инструментом для визуализации работы информационной системы и отлично подойдет для функционального показа работы программы и взаимодействие с ней.

2.2 Разработка концептуальной модели системы

Одним из важнейших факторов создания программы является разработка концептуальной модели системы. Для составления концептуальной модели, необходимо прежде всего определить возможные взаимодействия предполагаемого пользователя с программным продуктом. Наилучшим способом представления этого взаимодействия будет диаграмма вариантов использования. Данная диаграмма представлена на рисунке 1.

Рисунок 1 – Диаграмма вариантов использования

Исходя из диаграммы вариантов использования, можно отметить, что ключевыми факторами в системе будут задачи, сотрудники и само распределение.

После составления диаграммы вариантов использования можно составить карту навигации программы, которая позволяет ориентироваться пользователю в функционале. В карте представлен полный перечень страниц, кнопок и меню приложения. Карта навигации приложения представлена на рисунке 2.

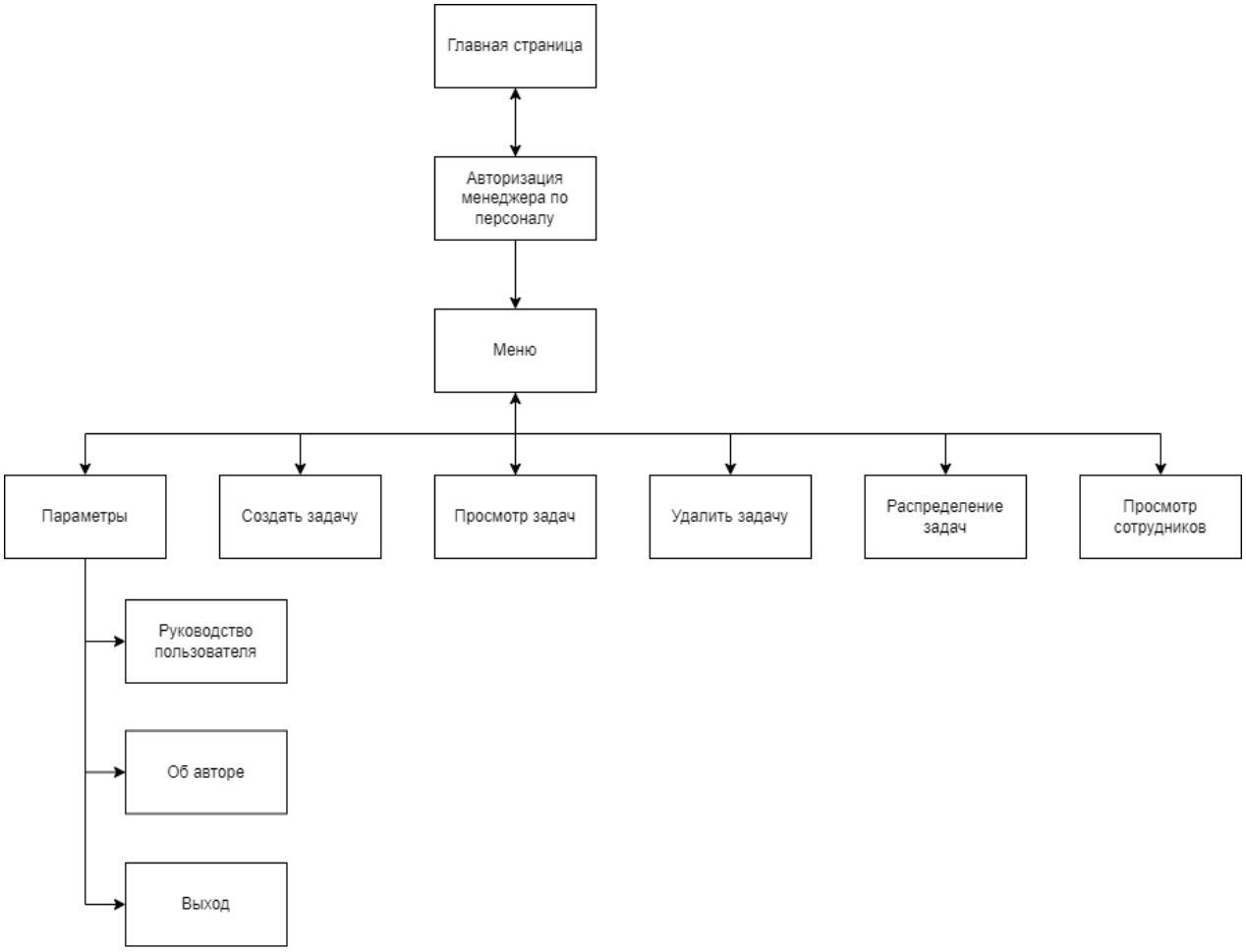


Рисунок 2 – Карта навигации программы распределения задач между сотрудниками

Для полного отображения разрабатываемой системы необходимо спроектировать концептуальную модель на основе составленных диаграмм и

схем. Концептуальная модель системы является абстрактным представлением структуры, функций и взаимосвязей элементов программы [10]. При помощи концептуальной модели возможно описание объекта разработки, средства разработки, программного обеспечения, строки состояний пользователя, рабочего пространства и самого меню приложения. Концептуальная модель системы распределения задач между сотрудниками представлена на рисунке 3.

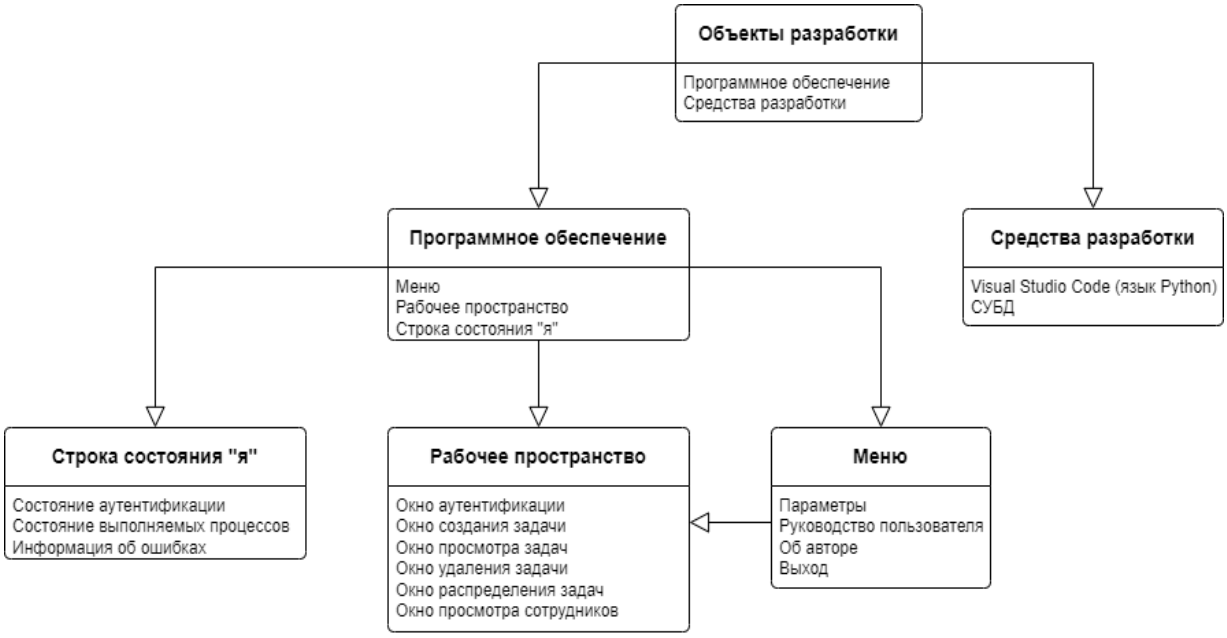


Рисунок 3 – Концептуальная модель системы

Благодаря составленной концептуальной модели можно разработать полноценный интерфейс пользователя и создать необходимый функционал приложения.

2.3 Описание проектируемой базы данных

Важнейшим действием для дальнейшего создания и описания функционала программы является определение структуры базы данных. База данных играет ключевую роль в хранении, организации и управлении

большими объёмами информации [21]. Благодаря ей возможна различная обработка данных, такая как изменение, добавление, удаление и поиск информации.

В современных технологиях существует два вида БД: реляционная и нереляционная. Реляционная база данных хранит в себе информацию в виде таблиц, связанных между собой. В нереляционной БД информация хранится без четко созданной структуры, из-за чего к данным сложнее получить доступ. Для выполнения динамических запросов в системе подойдет реляционная БД.

После определения структуры базы данных, необходимо выделить основные сущности в системе:

– сотрудники;
– задачи;
– таблица распределения.

Первой сущностью является таблица сотрудников. Данные о каждом сотруднике взяты с цифрового профиля в компании, где выделены основные показатели эффективности каждого из них. Ещё одной сущностью является таблица задач, где каждая из задач будет создаваться, удаляться с помощью запросов менеджера по персоналу. Третьей сущностью является таблица распределения. Данная таблица будет хранить в себе результат распределения задач между сотрудниками.

На основе полученных сущностей и их взаимодействия создана логическая модель системы, представленная на рисунке 4. В данной модели существует связь «один ко многим» от таблицы распределения к задачам и связь «один к одному» от таблицы распределения к сотрудникам в системе.

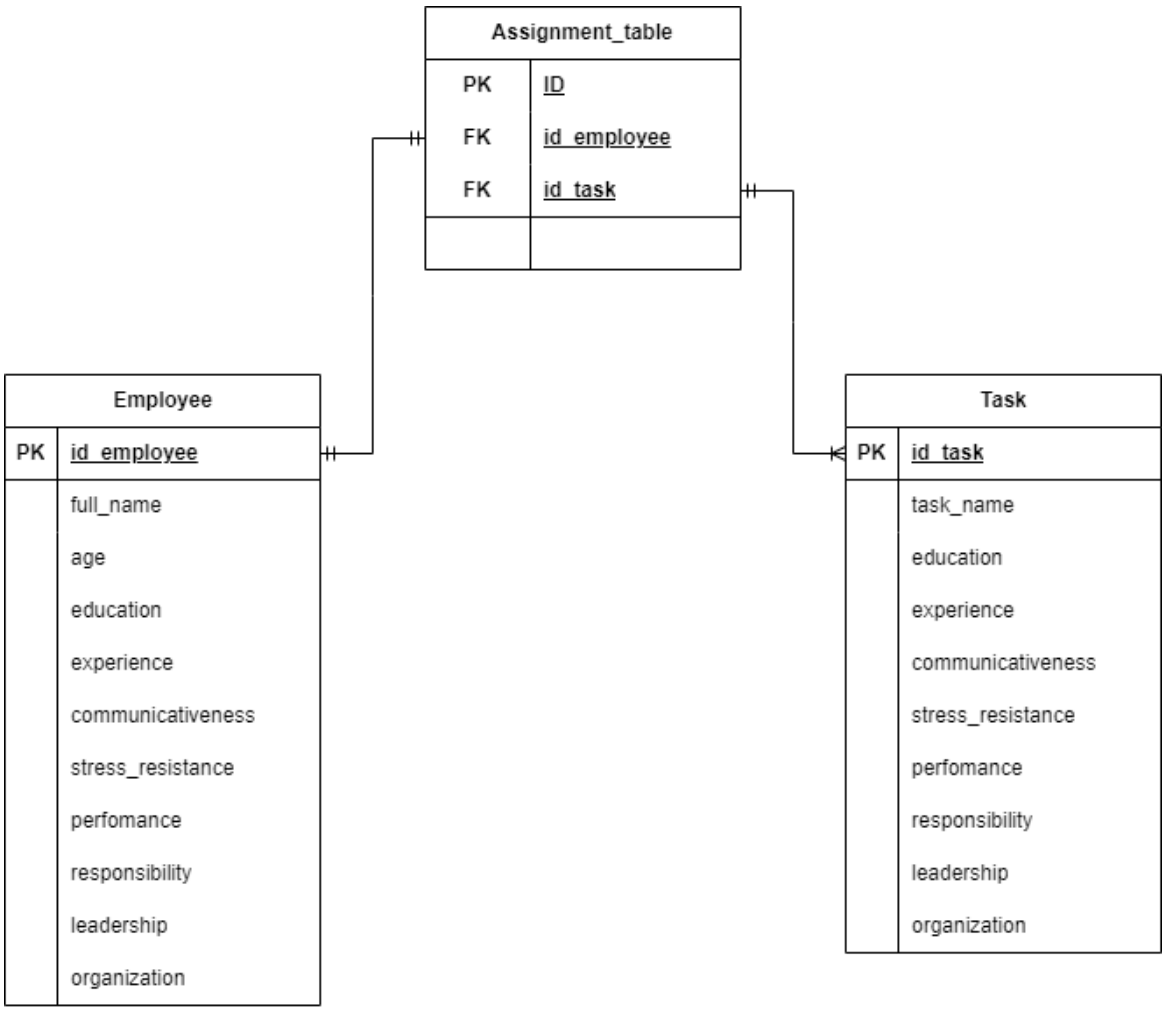


Рисунок 4 – Логическая модель данных

Благодаря полученной логической модели данных становится понятно, какую информацию будет хранить система. Для дальнейшего управления этими данными, необходимо выбрать систему управления базами данных.

2.4 Выбор СУБД для базы данных

В современном мире существует различное множество систем управления базами данных (СУБД), каждая из которых обладает собственными уникальными характеристиками и преимуществами [14]. Главными критериями практически любой СУБД являются: высокая

производительность, надежность, масштабируемость и простота использования [23].

Во многих проектах базы данных являются неотъемлемой частью всей информационной системы компаний и поэтому важно осознанно отнестись к её выбору. В целом, выбор конкретной СУБД зависит от направления проекта, его масштабов и требований к функциональности [18]. Важно учитывать плюсы и минусы каждой из баз данных и подходить к выбору СУБД основательно.

Для реализации поставленной задачи необходимо выбрать максимально подходящую СУБД. Для этого проведем сравнительный анализ нескольких средств управления баз данных.

Первым рассмотренным средством управления базой данных будет MySQL. MySQL является одной из популярных СУБД и используется для большого количества приложения от веб-сайтов до крупных корпоративных систем. Её достоинствами являются:

- высокая надежность;
- производительность;
- поддержка транзакций и различных видов запросов;
- поддержка многих операционных систем;
- открытый исходный код и сообщество разработчиков, которое осуществляет постоянную поддержку и улучшение системы.

Одни из главных недостатков СУБД заключаются в ограниченности масштабирования и управление данными [16]. Ей также не хватает некоторых продвинутых функций, которые доступны в других СУБД.

Следующим рассмотренным средством управления базой данных будет PostrgreSQL. Данная база данных также является одной из популярных в мире. Главными преимуществами PostrgreSQL являются:

- поддержка многопоточности и транзакций;

– богатый набор функций, включая хранимые процедуры, триггеры и полнотекстовый поиск;
– достаточно надежна и стабильна;
– поддержка практически всех современных операционных систем;
– отлично подходит для больших проектов.

Существенными недостатками являются сложность в настройке и использовании, за счет своих масштабов менее производительнее по сравнению с другими СУБД при малейших объемах данных.

Третьей рассмотренной СУБД будет Firebrird. Firebrird является свободной реляционной системой управления базами данных и относительно новой. Выделим некоторые преимущества Firebrird:

– хорошая масштабируемость;
– свой внутренний язык PSQL;
– полная поддержка процедур и триггеров;
– низкое требование к системным ресурсам;
– поддержка большинства операционных систем.

Недостатками Firebrird являются: имеет меньше сторонних инструментов, возможна дополнительная настройка для оптимальной производительности и сложен в настройке.

SQLite является компактным средством управления базой данных, которая интегрирована во множество приложений. У данной СУБД есть ряд преимуществ:

– не требует установки сервера, что экономит ресурсы компьютера;
– легко интегрируется в приложение;
– простая в настройке и использовании;
– полная поддержка языка SQL;
– совмещение с существующими приложениями;
– широкий спектр запросов к базе данных;
– надежная СУБД с минимальным риском потери данных.

Из минусов можно выделить: не поддерживает масштабирование на уровне других СУБД и не подходит для хранения огромного количества данных.

Учитывая требования к разрабатываемой системе, после выделения основных достоинств и недостатков каждой из СУБД, для поставленной задачи отлично подойдет SQLite. SQLite будет выполнять функцию хранения задач организации, независимо от основной базы данных компании с сотрудниками. Задач будет не так много, но они будут часто добавляться и удаляться менеджером по персоналу. Благодаря такому решению, легче выполнять динамические запросы, меньше нагрузки на систему и приложение в целом.

Выводы к главе 2

В данной главе была разработана информационная система распределения задач между сотрудниками.

В первую очередь были определены требования менеджера по персоналу к разрабатываемой системе и выбор методологии UML для её описания.

Следующим этапом составлена диаграмма вариантов использования и карта навигации для показа взаимодействия пользователя с программой. На основе полученных диаграмм была спроектирована концептуальная модель системы для определения функционала программы и её компонентов.

На основе спроектированной концептуальной модели была выбрана реляционная организация БД, выделены основные сущности и составлена логическая модель данных с соответствующими связями. Благодаря описанию базы данных был произведен выбор подходящей СУБД.

Глава 3 Программная реализация системы распределения задач между сотрудниками в организации

3.1 Описание работы системы распределения задач

На основе описанных функциональных требований к системе, карты навигации и концептуальной модели становится возможным реализация системы. Данная система реализована с помощью текстового редактора кода Visual Studio Code на высокоуровневым языке программирования Python [25]. Данный язык программирования отлично подходит для реализации современных приложений и дальнейшей их программной поддержки.

Для начала работы менеджеру по персоналу необходимо пройти авторизацию в приложении. После авторизации пользователь оказывается в главном меню, где на выбор ему предоставляется несколько функций приложения, а именно: создание задачи, просмотр созданных задач, просмотр сотрудников, распределение и удаление задач.

Для создания задачи необходимо задать её название и параметры. Созданные задачи будут добавляться в базу данных SQLite. Затем, менеджер по персоналу сможет просмотреть все созданные задачи и всех сотрудников в компании.

При выборе функции распределения задач между сотрудниками происходит скалярное произведение параметров задач на параметры сотрудников. Таким образом, получается матрица коэффициентов эффективности. Далее необходимо использовать Венгерский алгоритм, чтобы преобразовать полученную матрицу в вектор распределения. Данный вектор представляется в табличном виде и выводится итоговый результат. Если спустя какое-то время задача становится неактуальной, то её можно удалить, выбрав соответствующую функцию и откроется окно, в котором можно выбрать задачу для удаления.

Блок-схема алгоритма работы системы распределения задач между сотрудниками представлена на рисунках 5 и 6.

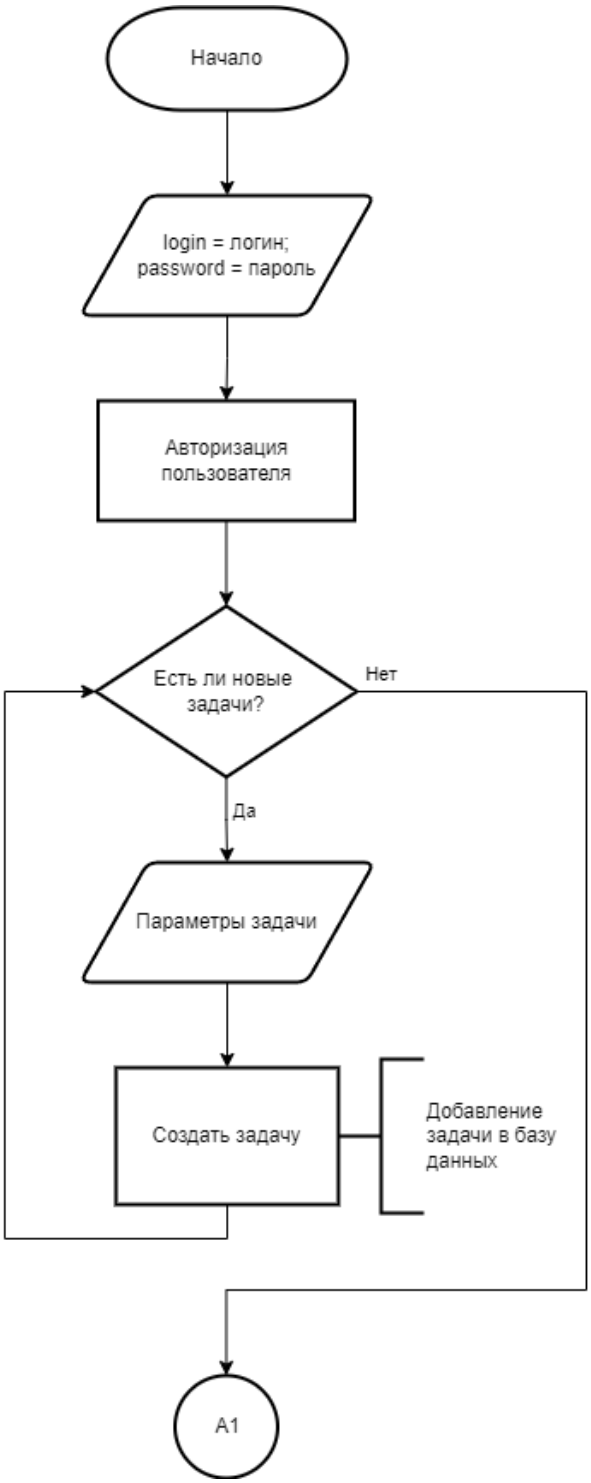


Рисунок 5 – Блок-схема алгоритма работы системы (Часть 1)

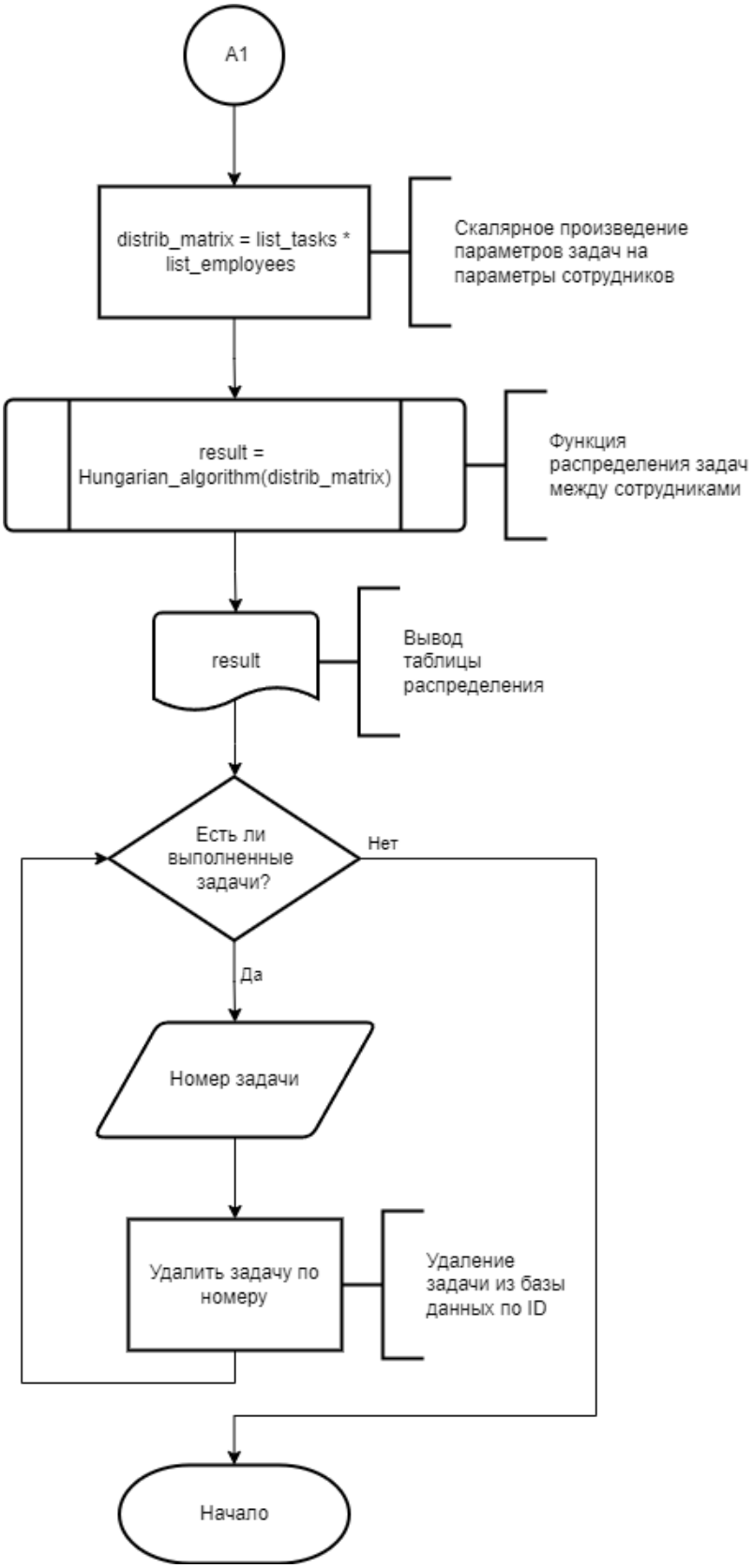


Рисунок 6 – Блок-схема алгоритма работы системы (Часть 2)

Программную реализацию распределения задач между сотрудниками можно представить в виде диаграммы классов. Для этого необходимо создать класс Employee, который будет отвечать за хранение данных сотрудников компании. Эти данные берутся из базы данных самой компании.

Следующим созданным классом будет Task, который необходим для хранения создаваемых задач менеджером по персоналу. Эти данные берутся из созданной базы данных.

Последним классом, который будет реализовывать распределение задач между сотрудниками является DistribitionList. Данный класс хранит в себе списки сотрудников компании и поставленных задач, а в его метод Distrib() встроен Венгерский алгоритм, который позволяет провести распределение. Диаграмма классов системы распределения задач между сотрудниками представлена на рисунке 7.

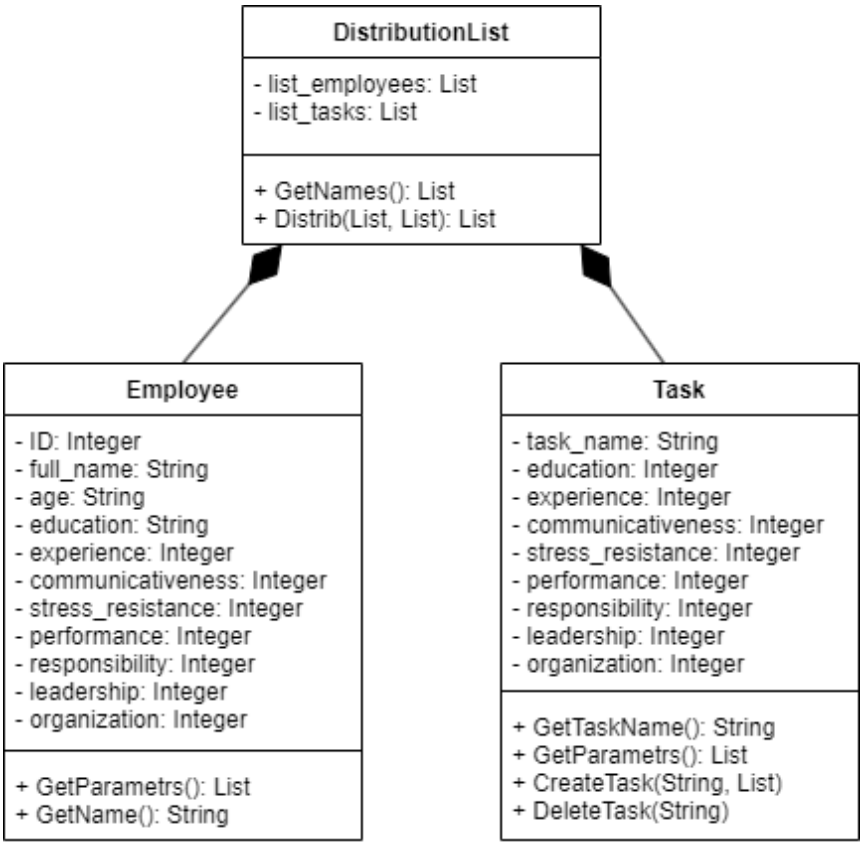


Рисунок 7 – Диаграмма классов системы распределения задач между сотрудниками

Благодаря представленной диаграмме классов можно увидеть взаимодействие задач и сотрудников в системе при распределении.

3.2 Создание физической модели базы данных

Конечным этапом создания модели базы данных является проектирование физической модели на основе созданной ранее логической модели данных. Данная модель представлена на рисунке 8.

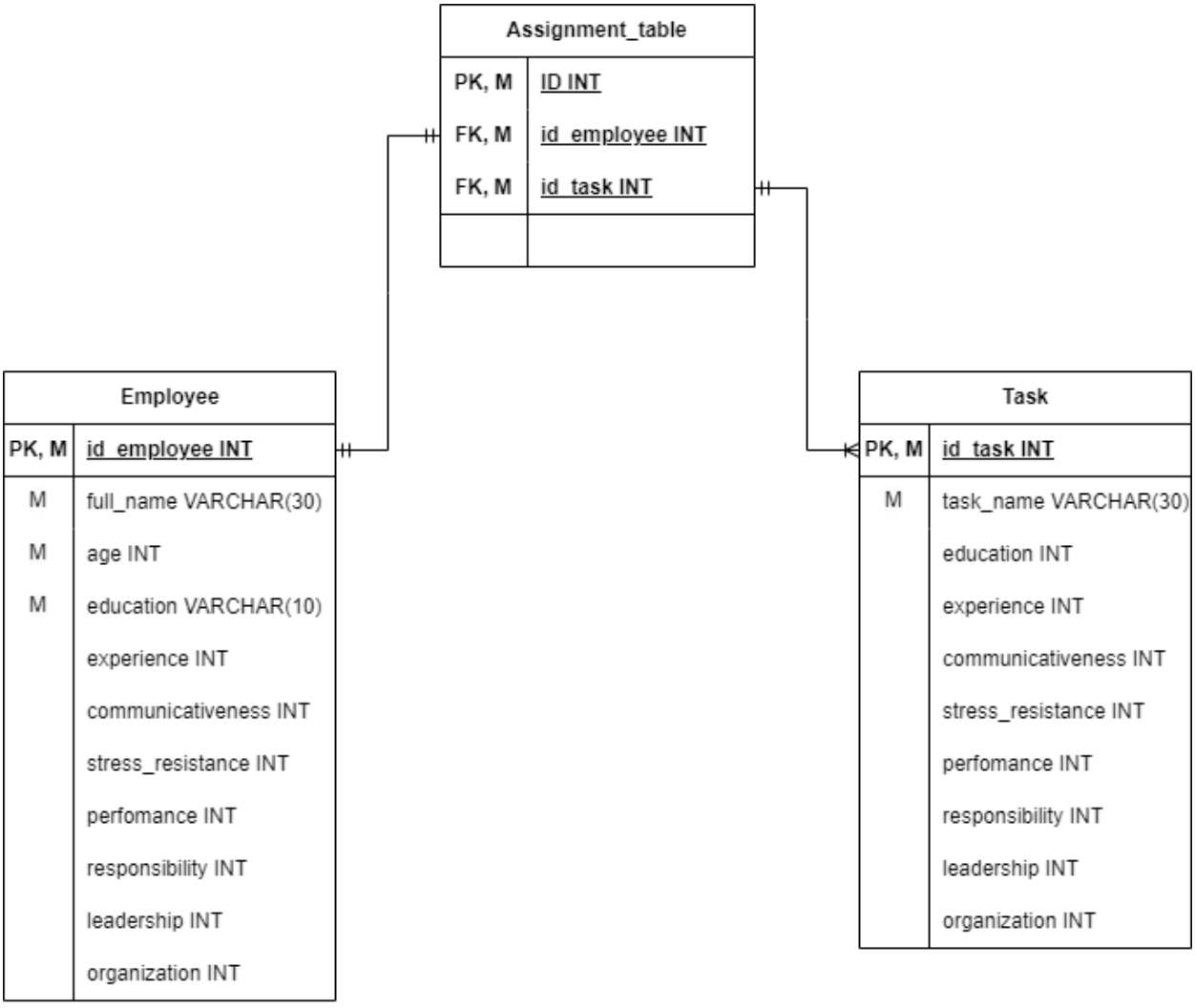


Рисунок 8 – Физическая модель данных

Сущностью физической модели является добавление к полям типов данных. Кроме этого, ключевым аспектом выступает добавление к некоторым полям маркера «М». Помеченные этим маркером поля являются обязательными для заполнения.

Таким образом, к внешним и первичным ключам добавился целочисленный тип данных INT. Такой же тип данных определен ко всем числовым показателям. Ко всем строковым полям добавился символьный тип данных VARCHAR с указанием выделенного количества символом для этих данных.

Физическая модель явно отражает структуру спроектированной базы данных. Данная модель показывает взаимодействие структурных объектов базы данных и позволяет определить типы данных и их обработку.

3.3 Разработка и реализация интерфейса программы

В современном мире интерфейс играет огромную роль в опыте пользователей и в успехе самого приложения. С помощью интерфейса оказывается прямое влияние на пользователей и определяет их общее впечатление от продукта. В первую очередь интерфейс должен быть привлекательным, понятным и удобным в использовании.

Графический интерфейс пользователя (GUI) согласован и оформлен в стиле компании. Функциональность самого приложения полностью соответствует определенным в начале работы требованиям.

Для реализации интерфейса необходимо подключить соответствующую библиотеку. В Python существует библиотека Tkinter содержащая в себе компоненты графического интерфейса пользователя. В графический интерфейс пользователя входят все окна, текстовые поля для ввода, скроллеры, кнопки и многое другое. Для создания GUI необходимо создать главное окно, все виджеты, определить всевозможные события в программе, расположить созданные виджеты и запустить цикл обработки событий [15].

В Tkinter существует различное множество виджетов. Одними из ключевых являются виджеты Entry, Label, Button и Frame [22]. В реализации интерфейса для текстового ввода данных использовался виджет Entry. Также реализован виджет Label, отображающий текст, изображение и виджет Button, позволяющий выполнять действие пользователю при нажатии. Виджет Frame необходим для организации всех виджетов на странице или окне в одной группе.

Взаимодействие пользователя с программой начинается с начальной страницы. На начальной странице приложения содержится приветствие пользователя и кнопка авторизации для менеджера по персоналу. Реализация начальной страницы приложения представлена на рисунке 9.

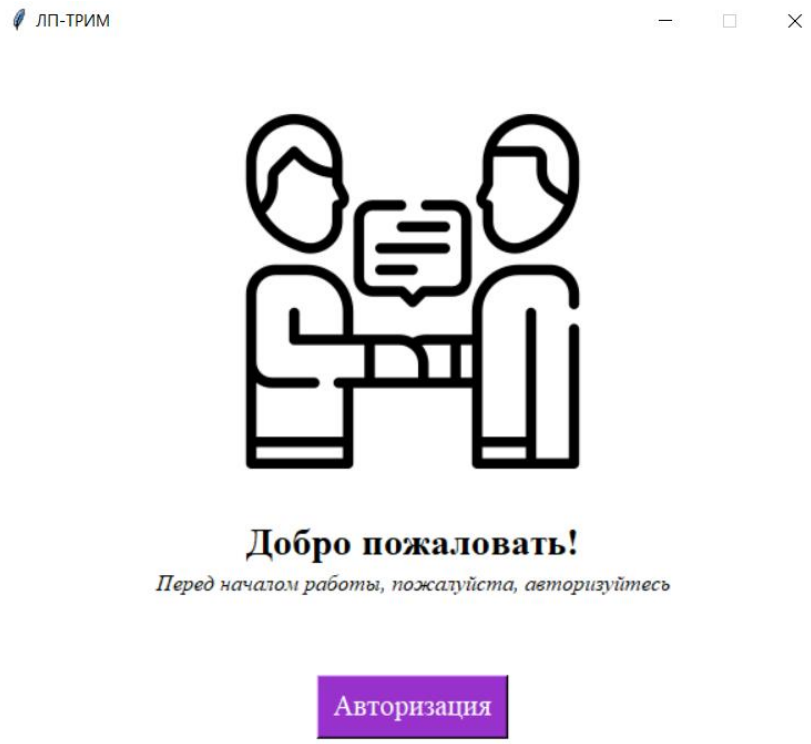


Рисунок 9 – Начальная страница входа в программу

Страница ввода логина и пароля обычно содержит форму, где пользователь должен ввести свой логин и пароль для входа в личный кабинет. При авторизации менеджеру по персоналу необходимо ввести данные от

личного кабинета пользователя для дальнейшего взаимодействия с приложением. Поле для ввода пароля является скрытым. (рисунок 10)

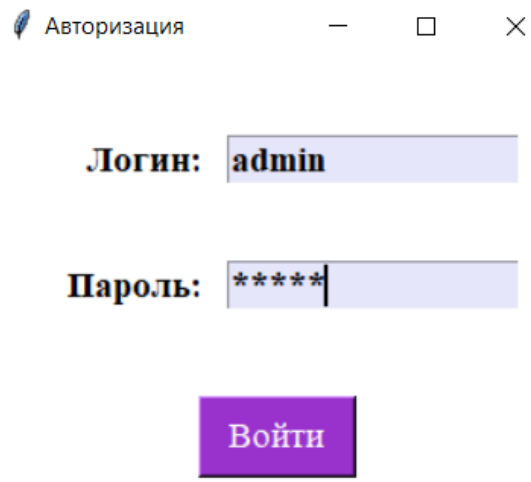


Рисунок 10 – Окно авторизации пользователя

После авторизации менеджера по персоналу в приложении, следующей страницей является главное меню программы. Главное меню служит для навигации авторизованного пользователя. Страница главного меню программы представлена на рисунке 11.

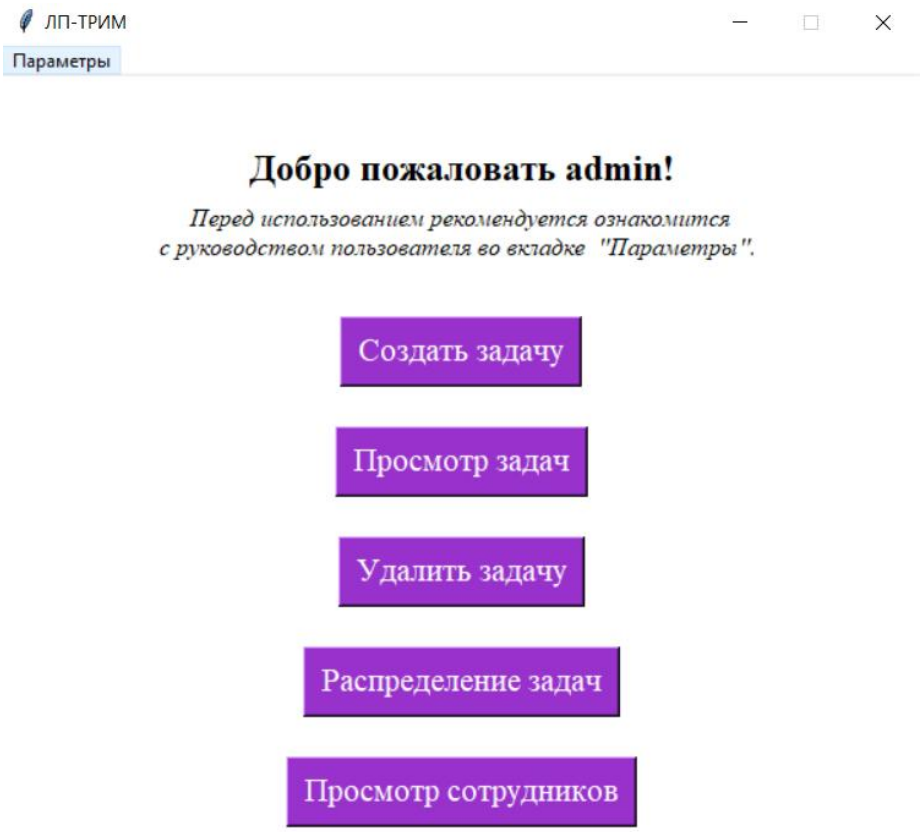


Рисунок 11 – Страница главного меню программы

В главном меню приложения содержится кнопка просмотра сотрудников в компании. С помощью данной функции менеджер по персоналу может полностью просмотреть список сотрудников и их личностные качества. Структура такого списка реализована с помощью виджета Treeview, который необходим для представления данных в табличном видео, а для полного просмотра реализован виджет Scrollbar, отвечающий за полосу прокрутки. На этой же странице содержится кнопка возврата в главное меню приложения. (рисунок 12)

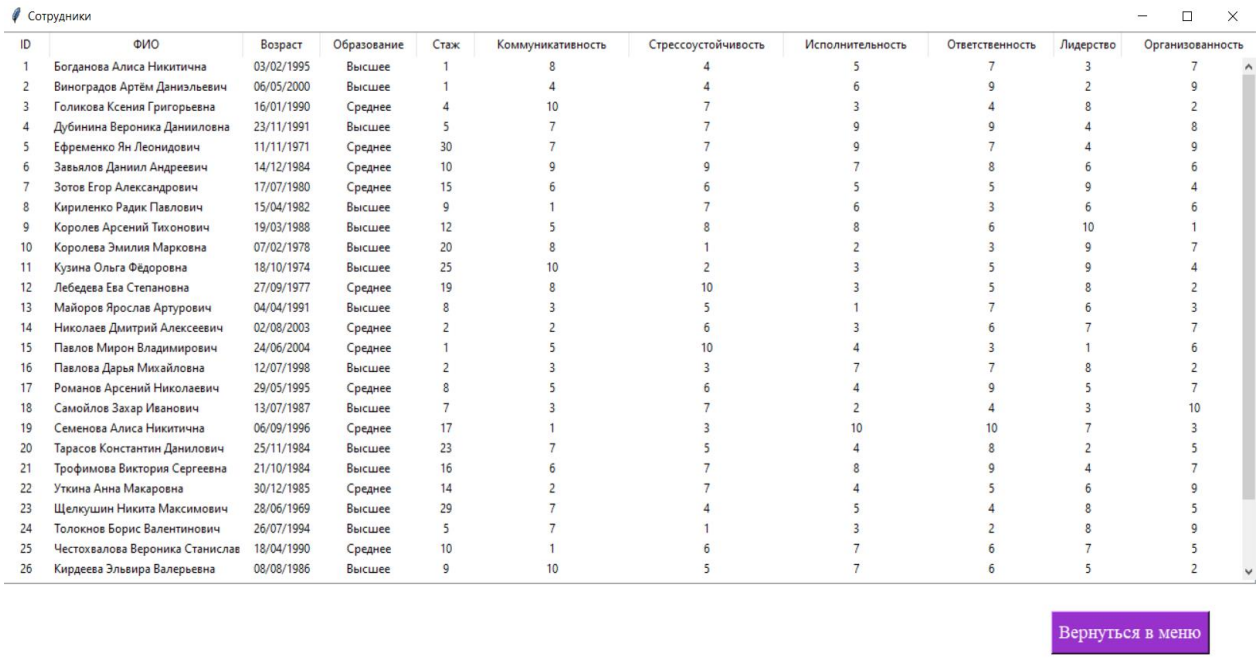


Рисунок 12 – Страница просмотра сотрудников компании

Также в главном меню приложения находится кнопка создания задачи. Менеджеру по персоналу после перехода по этой кнопке открывается страница, в которой обязательно ему необходимо заполнить поле с названием задачи и определить её параметры. На данной странице одним из реализованных виджетов является Combobox, который необходим для выпадающего списка каждого из параметров. После создания необходимых задач менеджер по персоналу также может выйти в главное меню. (рисунок 13)

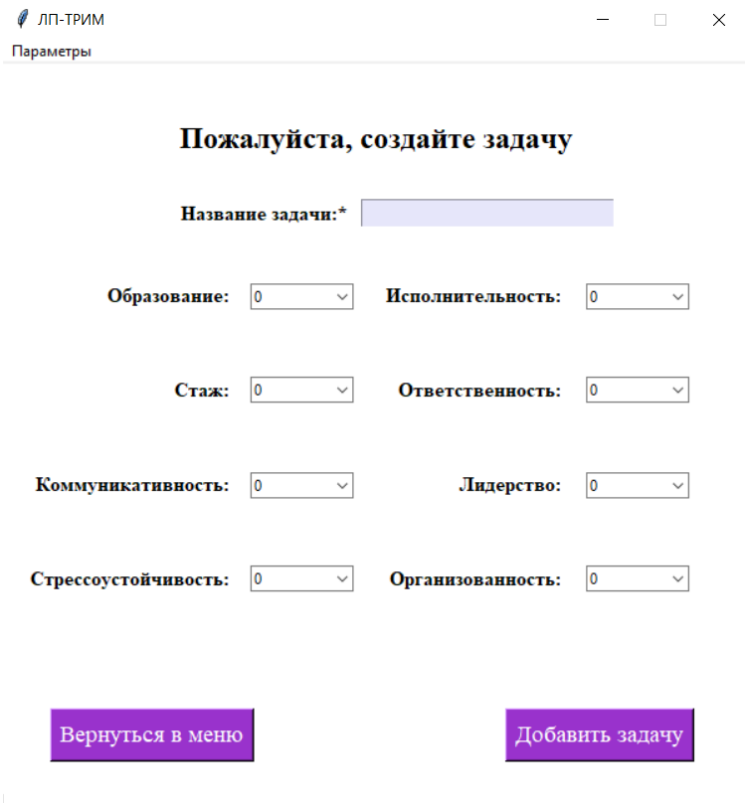


Рисунок 13 – Страница создания задачи и её параметров

После создания необходимого количества задач, менеджер по персоналу может их просмотреть. Для этого действия ему также необходимо перейти с главного меню по кнопке просмотра задач. В открывшейся странице можно увидеть все задачи в системе и их параметры, заданные ранее пользователем. Данная страница представлена на рисунке 14.

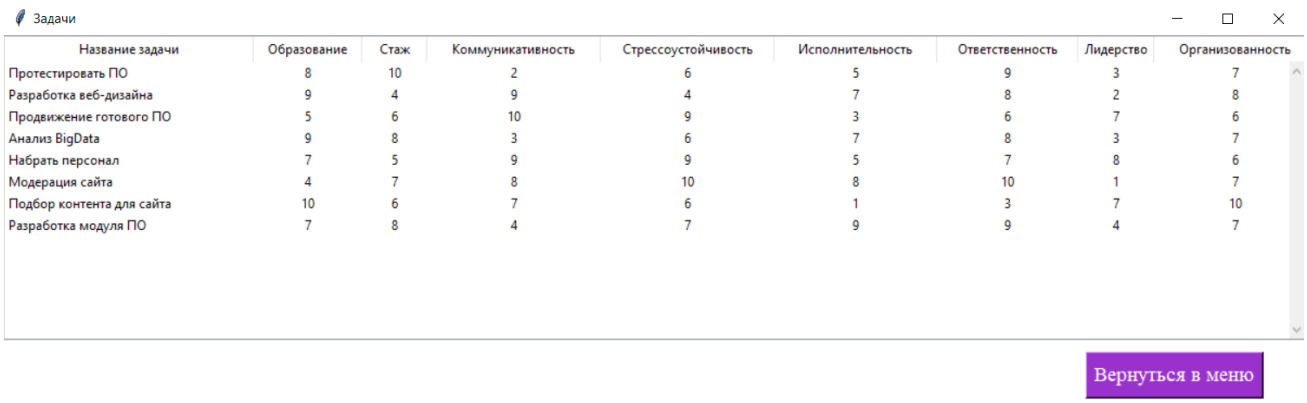


Рисунок 14 – Страница просмотра созданных задач

При наличии в системе неактуальных или случайно созданных задач, менеджер по персоналу может их удалить. Для этого ему необходимо перейти в окно удаления задачи и выбрать необходимую из списка. Данный список представляет собой созданные ранее задачи (рисунок 15).

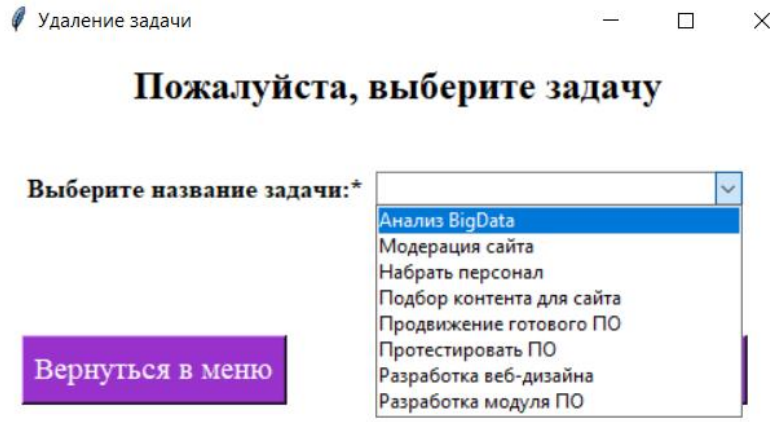


Рисунок 15 – Окно удаления задачи

Интерфейс созданного приложения также включает в себя вкладку параметры, в которой можно ознакомится с руководством пользователя, информацией об авторе и выйти из приложения. Данная вкладка меню создана с помощью виджета Menu.

3.4 Тестирование системы

Тестирование системы имеет огромное значение для обеспечения правильной работоспособности приложения. Без тестирования, все недоработки и ошибки могут повлечь за собой потерю личных данных, сбои в работе и нарушение безопасности приложения в целом. Поэтому тестирование является неотъемлемой частью в разработке любых программных продуктов.

Существует несколько основных видов тестирования системы на работоспособность и соответствия заявленным требованиям.

Одним из главных видов тестов является модульное тестирование. Такое тестирование позволяет проверять отдельные функции, классы или модули в программе на их работоспособность [11].

Интеграционное тестирование необходимо для проверки работоспособности различных модулей и сервисов, используемых в приложении. Благодаря таким тестам можно протестировать взаимодействие программы с базой данных и убедиться в правильности обрабатываемых запросов.

Ещё одним важным видом тестирования является функциональное. Данное тестирование основано на проверке системы в соответствии с заданными требованиями и ожиданиями пользователей. Именно такой вид тестирования проверяет только результаты запросов пользователей, а не промежуточные действия.

Сквозное тестирование также является одним из важных тестов программы. Благодаря таким тестам можно предсказать поведение пользователя в системе и предотвратить множество пользовательских ошибок [24]. Такие тесты работают благодаря созданию сценария работы пользователя с программой. Главным недостатком такого тестирования является его дорогая стоимость.

Следовательно, для правильной работы программы, необходимо включать несколько видов тестирования.

Для реализации модульного тестирования в приложении был использован модуль Unittest, основанный на тестах JUnit языка Java. Unittest является встроенным модулем в Python, включающим все необходимые инструменты для проведения тестирования [9]. В данном модуле содержится класс TestCase с множеством проверочных методов assert для проверки выполнения условия в определенной части кода [1].

Эти тесты используются для проверки на корректность вводимых пользователем значений, а именно логин, пароль и название задачи для дальнейшего добавления или удаления.

При подтверждении пользователем логина и пароля посылается запрос в базу данных на поиск, если такой записи не обнаружено или логин и пароль являются пустыми строками, то запрос вернет значение «None». Поэтому в программе у объекта тестирования self вызывается метод «assertIsNone(x)», который проверяет, является ли значение или переменная «None».

Во время добавления/удаления пользователем задачи необходимо ввести её название/выбрать её название. Ошибкой пользователя может быть не введенное/не выбранное название задачи. Для этого будет использоваться метод «assertEqual(x, y)», который служит для сравнения введенного/выбранного пользователем значения с пустой строкой.

После проведения тестирования были приняты меры по обнаружению и сообщению пользователю о допущенных ошибках. Для этого необходимо, при обнаружении ошибки, использовать метод «showerror()» класса MessageBox, который выведет на экран сообщение об ошибке. Одним из примеров этого метода является функция «check_authorization» для проверки логина, пароля и сообщение об ошибке, представленные на рисунках 16 и 17.

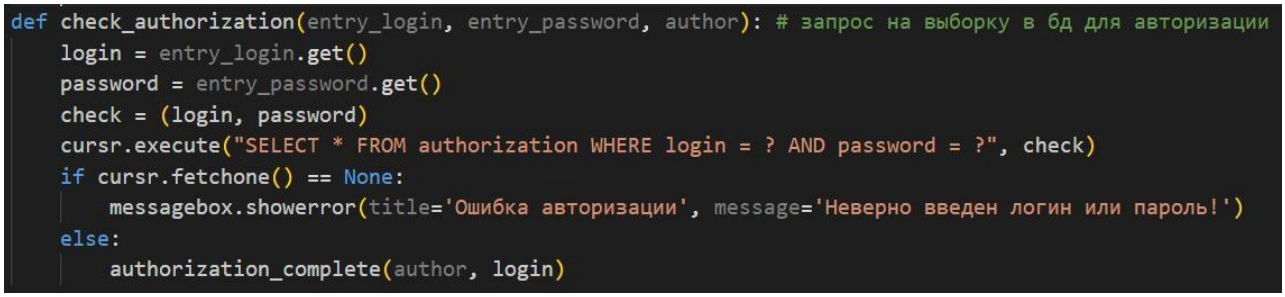


Рисунок 16 – Функция «check_authorization»

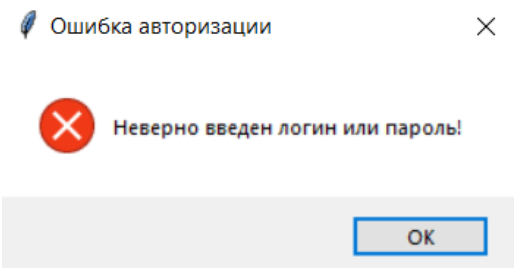


Рисунок 17 – Окно ошибки авторизации

В программе также существуют оконные сообщения об ошибках для других страниц. Например, для создания задачи пользователю обязательно необходимо заполнить поле названия задачи, в противном случае появится окно ошибки. Данное оконное сообщение представлено на рисунке 18.

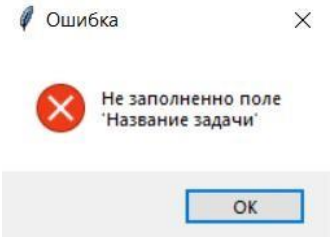


Рисунок 18 – Окно ошибки страницы создания задачи

Также в приложении может возникнуть ошибка на странице удаления задачи. Например, пользователю будет необходимо удалить ранее созданную задачу, но он не выберет её из списка. В таком случае появляется окно ошибки для оповещения пользователя (рисунок 19).

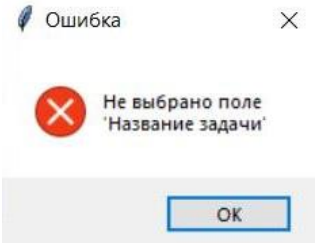


Рисунок 19 – Окно ошибки страницы удаления задачи

Следующим этапом необходимо провести проверку адекватности работы системы распределения задач с помощью функционального тестирования. Одним из главных реализованных функций программы является распределение задач между сотрудниками.

Для проведения проверки была создана база данных сотрудников с коэффициентами эффективности и несколько задач со своими заданными параметрами. В качестве примера возьмём сотрудника Богданову Алису с показателями, представленными на рисунке 20.

ФИО	Возраст	Образование	Стаж	Коммуникативность	Стрессоустойчивость	Исполнительность	Ответственность	Лидерство	Организованность
Богданова Алиса Никитична	03/02/1995	Высшее	1	8	4	5	7	3	7

Рисунок 20 – Окно с показателями сотрудника Богдановой Алисы

Следующим этапом был запущен сам алгоритм распределения задач между сотрудниками. С помощью алгоритма сотрудник Богданова Алиса назначается на задачу разработки веб-дизайна исходя из своих показателей. Результат работы алгоритма представлен на рисунке 21.

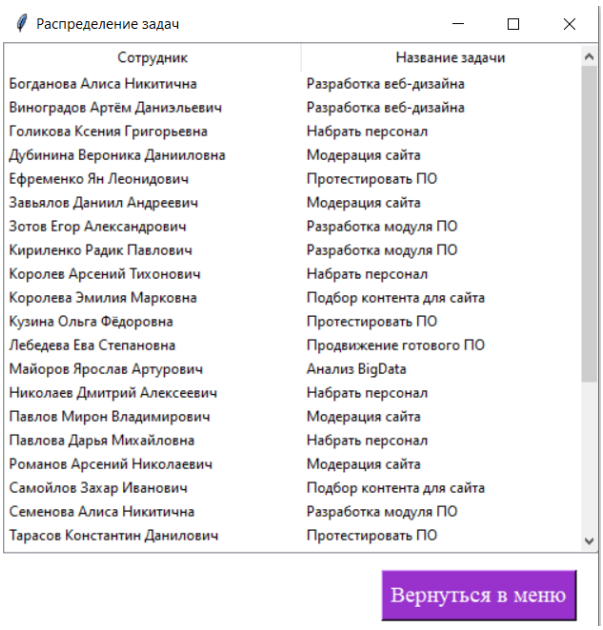
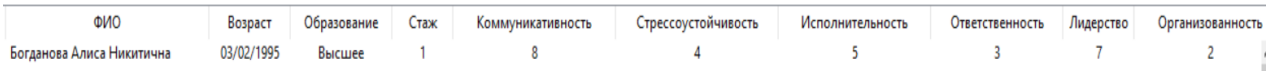


Рисунок 21 – Окно распределения задач между сотрудниками

Вывод итогового распределения задач между сотрудниками представлен в табличном виде. Первый столбец содержит ФИО сотрудников, а второй название задачи, на которую распределяется каждый из сотрудников.

Далее для проверки работоспособности алгоритма необходимо поменять параметры сотрудника, а именно Богдановой Алисы (рисунок 22)



ФИО	Возраст	Образование	Стаж	Коммуникативность	Стрессоустойчивость	Исполнительность	Ответственность	Лидерство	Организованность
Богданова Алиса Никитична	03/02/1995	Высшее	1	8	4	5	3	7	2

Рисунок 22 – Окно с показателями сотрудника

В результате работы программы данный сотрудник назначается на другую задачу (рисунок 23).

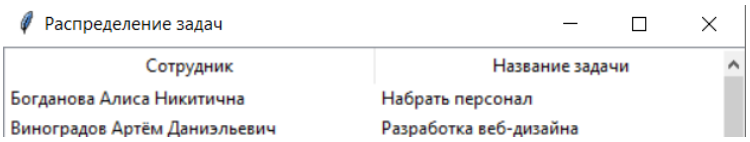


Рисунок 23 – Окно примера распределения задач между сотрудниками

Таким образом, при смене показателей эффективности любого из сотрудников алгоритм работает корректно и распределяет на оптимально подходящую задачу.

Благодаря проведенному тестированию можно сделать вывод, что программа работает корректно и оповещает пользователя об ошибках. Результаты тестирования подтверждают, что реализованная программа является хорошим инструментом для автоматизации процесса распределения задач между сотрудниками в компании.

Выводы к главе 3

В данной главе была описана и программно-реализована работа системы распределения задач между сотрудниками с помощью блок-схемы и

диаграммы классов. Блок-схема составлена на основе описанных во 2 главе функциональных требований и благодаря ей была спроектирована диаграмма классов для реализованной программы.

Следующим этапом создана физическая модель данных на основе логической. Физическая модель отлично отражает структуру спроектированной базы данных.

Основываясь на требованиях компании, был разработан и реализован интерфейс пользователя. Реализованный интерфейс согласован и оформлен в стиле компании, функциональность полностью соответствует поставленным требованиям.

Последним этапом было проведено тестирование приложения распределения задач между сотрудниками. В результате тестирования было выявлено, что система работает стабильно, правильно и оповещает пользователя об возникающих ошибках.

Заключение

Итогом выпускной квалификационной работы является программная реализация алгоритма распределения задач между сотрудниками.

В рамках работы была выявлена проблема эффективного распределения задач между сотрудниками. Данная проблема заключалась в большом количестве задач, сотрудников и сложности используемых алгоритмов для их распределения, поэтому было принято решение автоматизировать данный процесс.

Для решения данной проблемы первым этапом была поставлена задача и составлена её математическая модель. Затем был проведен анализ существующих подходов и выбран наиболее оптимальный из них, а именно Венгерский алгоритм.

Следующим этапом была спроектирована информационная система распределения задач между сотрудниками. На данном этапе были определены функциональные требования к разрабатываемой системе. На основе этих требований была составлена структура системы. Также описана проектируемая база данных с помощью логической модели. Благодаря созданной модели была выбрана СУБД.

Заключительным этапом была программная реализация системы распределения задач между сотрудниками. Для этого была сделана блок-схема алгоритма работы программы и на её основе построена и реализована диаграмма классов проектируемой системы. На этом же этапе благодаря созданной логической модели сделана физическая модель данных. Затем был разработан и реализован интерфейс программы. Заключительным действием было произведено тестирование системы.

Таким образом, разработанное приложение может быть использовано в крупных компаниях для оптимального и быстрого распределения задач между сотрудниками.

Список используемой литературы и используемых источников

1. Васильев, Ю. Обработка естественного языка. Python и spaCy на практике. – СПб.: Питер, 2021. – 256 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-1506-8.
2. Венгерский алгоритм, или о том, как математика помогает в распределении назначений / [Электронный ресурс] URL: https://habr.com/ru/articles/422009/, (дата обращения: 11.01.2024).
3. Галяутдинов, Р.Р. Транспортная задача - решение методом потенциалов // Сайт преподавателя экономики. [2013]. URL: https://galyautdinov.ru/post/transportnaya-zadacha, (дата обращения: 21.02.2024).
4. Данилин, Д.А. Решение задач о назначениях в области разработки программного обеспечения с использованием ключевых показателей эффективности / Д.А. Данилин, З.И. Баусова // Научный Форум: Экономика и Менеджмент. – 2017. – С.39-44.
5. Еремина, И.И. Комбинаторная оптимизация процесса распределения задач и трудовых ресурсов / И.И. Еремина, Д.М. Лысанов // Фундаментальные и прикладные науки сегодня: материалы XXII международной научно-практической конференции. Morrisville, 2020. С. 108-111.
6. Жидкова, Н.В. Методы оптимизации систем : учеб. пособие / Н.В. Жидкова, О.Ю. Мельникова. – Саратов : Ай Пи Эр Медиа, 2018. – 149 с. – ISBN 978-5-4486-0257-3.
7. Забродин, А. В. Основы проектирования информационных систем с помощью языка UML : учебное пособие / Забродин А. В., Бубнов В. П. - Санкт-Петербург : ПГУПС, 2018. - 46 с. - ISBN 978-5-7641-1133-9.
8. Зайцев, М.Г. Методы оптимизации управления и принятия решений: примеры, задачи, кейсы: учебное пособие. – 2-е изд., испр. / М.Г. Зайцев, С.Е. Варюхин. – М.: Дело АНХ, 2017. - 640 с. – ISBN: 978-5-7749-1070-0.

9. Златопольский, Д. М. Основы программирования на языке Python. 2-е изд. – Москва: ДМК Пресс, 2018. – 396 с.: ил. – ISBN 978-5-97060-641-4.

10. Кочегурова, Е. А. Теория и методы оптимизации : учебное пособие для вузов / Е. А. Кочегурова. – Москва : Издательство Юрайт, 2024. – 133 с. – (Высшее образование). – ISBN 978-5-534-10090-7.

11. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Издательский дом «Питер», 2018. – 352 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-0772-8.

12. Медведев, С. Н. Задача о назначениях с дополнительными ограничениями / С.Н. Медведев, О.А. Медведева. – Воронеж : Издательский дом ВГУ, 2015. – 37 с. – URL: https://rucont.ru/efd/590423, (дата обращения: 21.04.2024).

13. Моделирование данных: обзор / [Электронный ресурс]. URL: https://habr.com/ru/articles/556790/, (дата обращения: 04.02.2024).

14. Основы правил проектирования базы данных / [Электронный ресурс]. URL: https://habr.com/ru/articles/514364/, (дата обращения: 12.02.2024).

15. Сарычева Ю.Ю., Белов Ю.С. Применение искусственного интеллекта в автоматизированном тестировании GUI // Научные исследования в современном мире. Теория и практика: сборник избранных статей Всероссийской (национальной) научно-практической конференции. 2022. С. 55–56.

16. Связи между таблицами базы данных / [Электронный ресурс]. URL: https://habr.com/ru/articles/488054/, (дата обращения: 03.02.2024).

17. Сдвижков, О. А. Практикум по методам оптимизации : учебное пособие / О.А. Сдвижков. – Москва : Вузовский учебник : ИНФРА-М, 2022. – 200 с. – ISBN 978-5-9558-0372-2.

18. Смоленцева Т.Е. Формирование модели управления иерархическими многоуровневыми организационными системами//Сборник

научных трудов международной научно–технической конференции. Проблемы и перспективы развития машиностроения, 2016.-с. 235- 238.

19. Стронгин, Р.Г. Исследование операций. Модели экономического поведения / Стронгин Р.Г. – Москва : Национальный Открытый Университет ИНТУИТ, 2016. – 246 с. – (Основы информационных технологий). – ISBN 978-5-94774-547-4.

20. Фомин, Г. П. Экономико-математические методы и модели в коммерческой деятельности : учебник для бакалавров / Г. П. Фомин. – 4-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2021. – 462 с. – (Бакалавр. Академический курс). — ISBN 978-5-9916-3021-4.

21. BPMN Specification – Business Process Model and Notation / [Электронный ресурс] URL: https://www.bpmn.org/, (дата обращения: 12.02.2024).

22. Resources for Developers. Article about Document Object Model / [Электронный ресурс] URL: https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model/Introduction, (дата обращения: 21.03.2024).

23. Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases / [Электронный ресурс] URL: https://logz.io/blog/relational-database-comparison/, (дата обращения: 17.02.2024).

24. The Art of Software Testing, 3rd Edition / [Электронный ресурс] URL: https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf, (дата обращения: 18.04.2024).

25. What Is DOM: A Complete Guide To Document Object Model / [Электронный ресурс] URL: https://www.lambdatest.com/blog/document-object-model/, (дата обращения: 19.02.2024).