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

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «**Математическая модель и алгоритм принятия решений в гостиничном бизнесе в условиях неопределенности**»

Студент	Е.С. Быкова	
	(И.О. Фамилия)	(личная подпись)
Руководитель	Э.В. Егорова	
	(И.О. Фамилия)	(личная подпись)
Консультанты	А.В. Москалюк	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н, доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____2018 г.

Тольятти 2018

АННОТАЦИЯ

Тема выпускной квалификационной работы – Математическая модель и алгоритм принятия решений в гостиничном бизнесе в условиях неопределенности.

Ключевые слова: математическая модель, алгоритм, принятие решений в условиях неопределенности, принятие решений в гостиничном бизнесе.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка используемой литературы и приложения. Объем работы составляет 43 страницы, 13 рисунков, 10 формул и 9 таблиц. Использовано 22 источника литературы.

Цель исследования – разработать математическую модель и алгоритм принятия решений в гостиничном бизнесе в условиях неопределенности.

Объектом исследования является бизнес-процесс выбора и обработки заявок на бронирование в гостиничном комплексе «Звезда Жигулей»

Предметом исследования является создание математической модели процесса выбора и обработки заявок на бронирование.

Введение к ВКР включает в себя описание цели и актуальности создания модели и алгоритма принятия решений в гостиничном бизнесе, а также краткую структуру ВКР.

В первой главе проведен анализ методов принятия решений в условиях неопределенности. Определен подход к разработке.

Во второй главе сформулированы требования к создаваемому программному продукту. Также описано и математически решена задача о принятии решения в гостиничном бизнесе в условиях неопределенности.

В третьей главе представлена программная реализация и тестирование разработанного программного продукта решающего задачи гостиничного бизнеса в условиях неопределённости, а также представлен пользовательский интерфейс.

В заключении представлены развернутые выводы по проделанной работе и ее результатам.

ABSTRACT

The title of the graduation work is the mathematical model and decision-making algorithm under uncertainty in the hotel business.

The subject of the work is the automation of the process of selecting and processing requests for reservations.

The goal of the graduation work is the development and application of a mathematical model and decision-making algorithm in the development of a software product under uncertainty in the hotel business.

This work is relevant due to the need to automate the business process of selecting and processing applications for the reservation department of the hotel " Zvezda Zhiguli ", the divisions owned by Agat LLC.

The first chapter is devoted to the analysis of the subject domain of automation, rationale for chosen methodology, as well as the identification of shortcomings of existing business process and recommendations for its improvement

with the help of modern technology.

The second part of the work is focused on modeling and software development for the multi-agent order distribution system.

The third part of the work is devoted to testing the created system.

The result of the work is the running software for the multi-agent order distribution system, which automatically assigns distributors to customers with the help of communication agents.

The work consists of an explanatory note on 43 pages, including 13 images, 10 formulas, 9 tables and a list of 22 references.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
ГЛАВА 1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ, ОБЗОР МЕТОДОВ ПРИНЯТИЯ РЕШЕНИЙ, ОСНОВАННЫХ НА ПАРНОМ СРАВНЕНИИ АЛЬТЕРНАТИВ И ПОСТАНОВКА ЗАДАЧИ 8
1.1 Характеристика предприятия и его деятельности 8
1.2 Систематизация моделей и методов принятия решений 10
1.3 Модели линейного упорядочивания 12
1.4 Обоснование и постановка задач по дальнейшей реализации модели линейного программирования в программном продукте 14
ГЛАВА 2 РАЗРАБОТКА КОМБИНАТОРНОГО АЛГОРИТМА ПРИНЯТИЯ РЕШЕНИЙ, ОПИСАНИЕ И МАТЕМАТИЧЕСКО РЕШЕНИЕ ЗАДАЧИ ПРИНЯТИЯ РЕШЕНИЯ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ 16
2.1 Описание задачи ПР в общем виде, принцип вычисления главного собственного вектора примитивных матриц 16
2.2 Математическое решение задачи принятия решения в гостиничном бизнесе 19
ГЛАВА 3 РАЗРАБОТКА ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ПРИНЯТИЯ РЕШЕНИЙ С НЕЧЕТКОЙ ИСХОДНОЙ ИНФОРМАЦИЕЙ 28
3.1 Реализация алгоритма и модели принятия решения в гостиничном бизнесе в условиях неопределенности 28
3.2 Тестирования программы 39
ЗАКЛЮЧЕНИЕ 41
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 42
ПРИЛОЖЕНИЕ 44

ВВЕДЕНИЕ

В настоящее время развитие компьютерных технологий обуславливает значительные изменения в форме и качестве принятия управленческих решений и разработки стратегических планов развития, как на уровне отдельных компаний, так и отраслей экономики.

В сфере гостиничных услуг, в которой ведет свою деятельность ООО «Агат», наблюдается бурный рост отрасли гостиничного комплекса «Звезда Жигулей».

Важная роль принадлежит системе бронирования гостиницы. Администраторы принимают и обрабатывают заявки, занимаются их структуризацией. В настоящее время многие предприятия, в особенности крупные, имеют филиалы во многих городах. В связи с этим возникает необходимость отправлять своих сотрудников в длительные и многочисленные командировки. ГК «Звезда Жигулей» является первым в приоритете на выбор гостиницы командировочных гостей, в связи со своим расположение в центре города и в шаговой доступности от промышленной зоны города Тольятти.

На рассмотрение и принятие решений о бронировании на длительный период или на большое число спальных мест гостиница имеет время в течении одного рабочего дня. После чего при поступлении всех заявок администратор вручную производит расчеты релевантности заявок и выбирает несколько их них. Данные расчеты зачастую неверны и имеют погрешность из-за личной вовлеченности администратора.

Важная роль в этой деятельности принадлежит службе бронирования гостиницы. Диспетчеры занимаются приемом и обработкой заявок клиентов на бронирование номеров, формированием и распределением заявок среди номерного фонда гостиницы. Для организации эффективной и качественной системы, способной в значительной степени сократить время ожидания клиента и обеспечить конкурентоспособность компании, необходимо обеспечить высокий уровень автоматизации работы диспетчеров. В ООО «Агат» с этой целью принято решение о разработке и внедрении программного модуля для

основной программы Hotel_3.pro, обеспечивающего автоматизацию бизнес-процесса учета и обработки заявок отдела бронирования. Таким образом, актуальность темы ВКР обусловлена необходимостью автоматизации бизнес-процесса выбора и обработки заявок отдела бронирования компании ООО «Агат» подразделения ГК «Звезда Жигулей».

Объектом исследования является бизнес-процесс выбора и обработки заявок на бронирование в гостиничном комплексе «Звезда Жигулей»

Предметом исследования является создание математической модели процесса выбора и обработки заявок на бронирование.

Целью выпускной квалификационной работы является разработка математической модели и алгоритма принятия решений в гостиничном бизнесе в условиях неопределенности.

Для достижения цели выпускной квалификационной работы необходимо решить следующие задачи:

− построить математическую модель гостиничного бизнеса;

− определить и сформулировать программную часть решения задачи;

− применить математическую модель и алгоритм принятия решений при моделировании программы;

− разработать модель программного продукта;

− разработать программный продукт на основании выявленных требований.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованной литературы и приложения.

Введение содержит в себе обоснование актуальности выбранной темы, цель, предмет исследования, а также обозначены основные задачи для решения.

Первая глава посвящена анализу предметной области автоматизации, обоснованию выбора методологии, а также выявлению недостатков существующего бизнес-процесса и рекомендациям по его усовершенствованию с помощью современных технологий.

Во второй главе приводится алгоритм решения задачи в условиях неопределенности и рассматривается процесс проектирования программного приложения, основанного на работе реализованного алгоритма.

В третьей главе описан процесс разработки программного приложения, позволяющего автоматизировать процесс принятия решения, а также проверка работоспособности его.

В заключении даны основные выводы по проделанной работе.

ГЛАВА 1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ, ОБЗОР МЕТОДОВ ПРИНЯТИЯ РЕШЕНИЙ, ОСНОВАННЫХ НА ПАРНОМ СРАВНЕНИИ АЛЬТЕРНАТИВ И ПОСТАНОВКА ЗАДАЧИ

1.1 Характеристика предприятия и его деятельности

Гостиничный комплекс «Звезда Жигулей» бы построен в 1976ом. ГК «Звезда Жигулей» зарегистрирован по адресу г.Тольятти, ул.Мира-77. ГК «Звезда Жигулей» имеет следующие подразделения:

- Гостиница «Звезда Жигулей»;
- Бар-ресторан «У Валентины»;
- Банкетный зал;
- Кулинария.

В ходе выпускной квалификационной работы рассматривается деятельность службы бронирования гостиницы «Звезда Жигулей». Рассмотрим виды ее деятельности:

- прием, выбор и обработка заявок на бронирование;
- бронирование номеров для гостей;
- регулировка состояния номерного фонда;
- контроль работы бара с гостями;
- контроль работы клининг-службы с гостями и номерами.

В связи с повышением поступления массовых (длительный период проживания/большое кол-во человек) заявок на бронирование от различных организаций служба бронирования все решения о подобных заявках принимает в конце рабочего дня. В течении суток поступающие заявки обрабатывают и вручную ведутся вычисления более релевантных заявок для гостиницы. В связи с тем, что в штат администраторов в основном набирают некомпетентных работников, которые в связи с отсутствием опыта и должного образования при поступлении заявки не могут оценить перспективу хода развития событий и возможных доходах или убытках. Во время принятия управленческих решений

сотруднику службы бронирования следует полагаться не только на собственный опыт, но и использовать отлично разработанные в наше время математические модели помощи в принятии решений, которые имеют возможность корректно делать выборку более выгодных альтернатив из тех, которые даны. Быстрый рост предприятия зависит от того, насколько квалифицированно и грамотно происходит поддержка принятия управленческих решений. Сотрудники, которые принимают решения, не используя вспомогательную аналитическую поддержку, применяют упрощенные, а порой и противоречивые решающие правила, что пагубно влияет на успех работы всей гостиницы.

Для реализации точной и эффективной системы бронирования, которая способна в большой степени уменьшить время ожидания клиента и подкрепить конкурентоспособность компании, следует обеспечить высокий уровень автоматизации работы диспетчеров. Программный модуль - это некоторое дополнение к основной программе, благодаря которому сотрудникам не понадобится более производить сложные математические вычисления для выявления лучшей альтернативы.

При моделировании и внедрении в рабочий процесс алгоритма принятия решений в условиях неопределенности в службе бронирования, обеспечиваются следующие возможности:

- ускорение обработки заявок на бронирование;
- отсутствие вероятности фактора личного отношения администратора при выборе между заявками;
- повышение прибыли из-за отсутствия ряда ошибок некомпетентного персонала;
- снижение риска, связанного с неопределенностью;
- снижение количества трансформационных издержек при передаче информации о клиенте.

Следовательно, требуется разработать модель и алгоритм принятия решений в условиях неопределённости, реализованный в виде программы, которая будет самостоятельно принимать решения, основываясь на предоставленные данные без погрешности на личную оценку вариантов, предложенных к сравнению. Нельзя упускать тот момент, что часть критериев для однозначной оценки отсутствует или является представленными в нечетком виде.

Таким образом получаем задачу в общем виде «Пусть дано ... надо найти лучшую альтернативу...». В роли решения данной задачи традиционно используются методы принятия решения – модель линейного упорядочивания. Для вычисления более подходящих модели и метода принятия решения проанализируем общую классификацию методов.

1.2 Систематизация моделей и методов принятия решений

Рассмотрим классификацию моделей и методов принятия решений. Форма задачи принятия решений является в формате: $<t, X, R, A, F, G, D>$, где t — формулировка проблемы (к примеру, подобрать один наилучший вариант либо организовать все множество вариантов); X — множество возможных вариантов; R — множество критериев оценки уровня достижения установленных целей; A — множество шкал замера по критериям (шк. интервальные, отношений, наименований, а так же порядковые,); F — отображение множества возможных альтернатив в множество критериальных оценок; G — система предпочтений решающего компонента; D — главный принцип, показывающий концепцию предпочтений. Структуризация моделей задач принятия решений может происходить в соответствии с ниже указанными критериями:

- по принципу их отображения F — детерминированное, вероятностное либо неясное, возможно отметить в соответствии с этим: ЗПР в

ситуациях определенности, ЗПР в ситуациях риска, ЗПР в ситуациях неопределенности;

- по мощности множества R — множество с одним элементом или состоящее из нескольких критериев, выделяются соответственно: ЗПР со скалярным критерием, ЗПР с векторным критерием (многокритериальные задачи);
- по типу системы *G* — отражает предпочтения одного лица или коллектива в целом, выделяются задачи индивидуального ПР, задачи группового ПР.

Моделью выбора считается пара *(X, R),* составленная из числа альтернатив *X* и бинарного отношения *R* на нем. При установлении модели ПР подразумевается, то что имеется некое множество исходных структур предпочтений и рассматривается определенная ЗПР, ход решения которой подразумевается наилучшим выбором метода обработки исходной структуры из определенного основного класса методов. Считается, что задана модель решения представленной ЗПР на множестве исходных структур, если указано условие, при котором некоторый набор методов сопоставляется произвольному отношению. Определенным базовым структурам сопоставляются определенные модели, ориентированные на соответствие тех или иных методов принятия решений.

В этой работе рассматривается возможность анализа определенного вопроса управления методом принятия решений в условиях неопределенности. Так как при исследовании финансовых, общественных, а также иных структур гостиницы, все больше информации бывает передана от людей, у которых есть опыт работы с системой и которые знают ее со всех сторон, имеющие образ о целях функционирования системы

Рассматривая систему бронирования и конкретно поставленную задачу (где ведется выбор среди множества заявок лучшей альтернативы), выбираем наиболее подходящий образ задачи принятия решения. А именно она должна отвечать следующим требованиям:

- ЗПР с векторным критерием (многокритериальная задача);
- ЗПР в условиях неопределенности;
- ЗПР отражающая предпочтения одного лица (в данном случае материальная выгода гостиницы).

В результате, появляющиеся во время управления гостиничным комплексом вопросы, обладающие свойствами неструктурированных задач ПР, имеется возможность проанализировать их с нескольких точек зрения методами, учитывающими неопределенности. Имея множество критериев к попарному сравниванию, необходимо их упорядочить и структурировать. Приведем сравнительную таблицу методов для выявления более оптимального варианта работы с поставленной задачей работы. В то же время под неопределенностью имеется в виду явление, которое невозможно проанализировать и измерить с какой-либо большой точностью

Таким образом, были рассмотрены методы принятия решения, анализ которых показал, что необходимо уделить методу линейного упорядочивания.

1.3 Модели линейного упорядочивания

Применяемые в практике модели прямолинейного упорядочивания классически разделяются на две крупные категории, отличающиеся собственным подходом к решению проблемы упорядочивания предметов.

В моделях первой категории, применяются статистические методы, любому объекту x_i, сопоставляется определенный интегральный показатель π_i, определяющий итоги его сравнений с иными объектами, а в дальнейшем объекты упорядочиваются в порядке убывания значений данного решающего фактора. В моделях второй категории разбираются показатели всей классификации и определяется классификация, увеличивая некоторый функционал качества. При этом оценок важности не производится. Кратко изучим некоторые модели первой группы

Всякая модель линейного упорядочивания для матриц попарных сравнений нуждается в определенных ограничений.

Модели спортивного типа - модель, где в качестве решающего фактора применяется полученная объектом «сумма очков».

Модель интегральной степени превосходства по сути своей достаточно схожа с моделей спортивного типа; используется для кососимметрических матриц. Недостатками этой модели являются введение коэффициентов весов объектов. В свою очередь данных весов заранее задано не может быть. Модель сводится к турнирной, а самостоятельно интереса не имеет.

Модель функции доминируемости рассчитана с целью обработки неясных отношений предпочтения. Используется для калибровок турнирной и кососимметрической матриц.

Модель Брэдли-Терри удобна для несложных структур без равноценных элементов и целочисленных турнирных матриц. Полученные компоненты нормализованного вектора могут служить количественными оценками важности объектов.

Модель Бэржа-Брука-Буркова используется с целью обработки простых структур, матриц с турнирной и степенной калибровками.

Модель обладает следующими свойствами: инвариантность к растяжению, транспонируемость. Элементы собственного вектора, которые были получены, являются оценкой важности объектов.

Стохастическая модель Ушакова предложена для обработки матриц, заданных в степенной и вероятностной калибровках.

Получаемые элементы конечного распределения могут быть применены в качестве количественных оценок важности объектов. Не обладает свойствами инвариантность к сдвигу, инвариантность к растяжению.

Модель равномерного сглаживания используется для положительных матриц с калибровками турнирная и спортивная. Обладает свойством инвариантности к растяжению. Для количественной оценки важности объектов используются полученные коэффициенты.

В системах поддержки принятия решений является достаточно полезным выбор модели упорядочивания с определенными свойствами, желаемыми в

данном конкретном случае. В свою пользу весомые доводы предоставляют такие модели как: функции доминируемости, Брэдли-Терри, Бержа-Брука-Буркова, стохастическая модель Ушакова, модель равномерного сглаживания. Модель Бержа-Брука-Буркова имеет возможность принимать во внимание неопределенность событий, не поддающихся вычислению с любым уровнем точности и соответственно учитывая неясность. Модель Бержа-Брука-Буркова также не имеет свойства инвариантности к сдвигу, однако на основании данной модели имеет метод к вычислению приоритетов для неполной матрицы.

Таким образом, для линейного упорядочивания в условиях неопределенности предпочтительнее пользоваться именно этой моделью. Рассмотрим методы принятия решений, ориентированные конкретно на модель Бержа-Брука-Буркова. Метод анализа иерархий развивает модель Бержа-Брука-Буркова и является одним из самых распространённых. С ним и будет идти работа в дальнейшем.

1.4 Обоснование и постановка задач по дальнейшей реализации модели линейного программирования в программном продукте

Рассматривается вопрос восстановления отношений в соответствии вектору приоритетов альтернатив. Такая возможность сформирована для вектора недоминируемых альтернатив, в соответствии с которой, показано как возможно восстановить обратно-симметричную матрицу МАИ. Восстановление неясного отношения нестрогого предпочтения в соответствии вектору приоритетов МАИ с точностью вплоть до постоянной осуществить невозможно.

Общеизвестно, то что при выполнении оптимального подбора сначала необходимо выявить требования, оказывающие большое влияние на процесс исследуемого события либо его результаты. Конкретные требования при этом поддаются формализованному суждению (т.е. имеют возможность быть представленными численно), а некоторые нет, и могут быть проявлены только при помощи индивидуальных оценок специалистов. Согласно данному

основанию главной чертой методов ПР считается их возможность принимать во внимание беспристрастную часть многокритериальной проблемы, представленную в виде численных значений. Модели ПР в условиях неопределенности используются с целью выбора наиболее приоритетных альтернатив из имеющихся в ситуациях, определяемых неточностью, неполнотой данных. Реализация распределения альтернатив в данной ситуации можно реализовать различными методами. Для ЛПР немаловажно, чтобы итоги применения методов предоставляли равное ранжирование альтернатив.

Реализуем сопоставление итогов распределения альтернатив МАИ и методов принятия решений при неясной исходной информации.

Таким образом получаем основные задачи для дальнейшей работы: реализовать выбранный метод МКА в качестве программы; применить метод МКА в случае выбора среди нескольких заявок на бронирование одновременно.

Вывод по главе.

Результат анализа моделей и методов принятия решений в условиях неопределенности показал, что для автоматизации работы службы бронирования наиболее подходящей является модель линейного программирования.

Результат анализа модели линейного программирования показал, что для автоматизации работы службы бронирования самым оптимальным является метод анализа иерархий, который развивает модель Бержа-Брука-Буркова.

Была произведена постановка конкретных задач для дальнейшей работы службы бронирования. Модель Бержа-Брука-Буркова является самой оптимальной для решения данных задач. Одним из следствий этой модели является метод анализа иерархий.

Была произведена постановка конкретных задач для дальнейшей работы над предметной областью.

ГЛАВА 2 РАЗРАБОТКА КОМБИНАТОРНОГО АЛГОРИТМА ПРИНЯТИЯ РЕШЕНИЙ, ОПИСАНИЕ И МАТЕМАТИЧЕСКО РЕШЕНИЕ ЗАДАЧИ ПРИНЯТИЯ РЕШЕНИЯ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ

В данной главе представлены нюансы проектирования смешанного (комбинаторного) алгоритма принятия решений в условиях неопределенности, помогающего с анализом мультифакториальной затрудненной ситуации разнообразными методами принятия решений. Смешанный алгоритм данного рода гарантирует рациональный подбор.

Следует подчеркнуть, что во время разработки смешанного алгоритма принятия решений тщательный фокус уделялся учету особенностей управленческих решений, что выразилось в последующих вопросах:

- каким образом отобразить в ходе принятия решений всевозможные взаимодействующие компоненты системы, в результате которой принимается решение;
- какие способы правильнее применять во время расчетов приоритетов принимаемых объектов.

2.1 Описание задачи ПР в общем виде, принцип вычисления главного собственного вектора примитивных матриц

Основные методы анализа иерархий строятся согласно классической теории матриц.

Матрицы парных сравнений МАИ являются положительно обратно-симметричными неприводимыми матрицами, к которым предъявляется условия согласованности.

Квадратные матрицы $A = \ a_{ij}$ где $a_{ij} > 0, i, j = 1,2,\dots,n$, $a_{ij} = \frac{1}{a_{ij}}, i, j, = 1,2,\dots,n$. именуются положительными обратно-симметричными матрицами.

Данные матрицы $A = a_{ij}$, для элементов которых производится соотношение $a_{ik} = a_{ij}a_{jk}$, $i, g, k = 1,2,\ldots,n$, представляются согласованными.

Квадратная матрица — неприводимая, в случае где матрица не имеет возможности быть приведена в виде $\begin{matrix} A_1 & 0 \\ A_2 & A_3 \end{matrix}$, где A_1 и A_3 — квадратные матрицы, *0* — нулевая матрица. Иначе матрицу именуют приводимой.

Теорема. Квадратная матрица или неприводима, или не может быть приведена путем перестановок индексов к виду:

$$\begin{matrix} A_1 & 0 & \cdots & 0 & 0 & \cdots 0 \\ 0 & A_2 & \cdots & 0 & 0 & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & A_k & 0 & \cdots 0 \\ A_{k+1,1} & A_{k+1,2} & \cdots & A_{k+1,k} & A_{k+1} & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mk} & A_{mk+1} & \cdots A_m \end{matrix}$$

включающий блок - диагональную матрицу с неприводимыми матрицами A_i, на диагонали. Вместе с тем, как минимум, единственная из матриц с двойным индексом в каждой строке, в которой они появляются, нулевая.

С помощью МАИ возможно рассчитать приоритеты альтернатив $U = \{u_1, u_2, \ldots, u_n\}$. Данными приоритетами, составленными основываясь на матрицы их парных сравнений, являются нормализованные значения главного собственного вектора матрицы.

Векторы *x*, где $A_x = \lambda_x$ $(x \neq 0)$, именуются собственными векторами, в то время как соотнесенные им числа λ — характеристическими, или собственными числами матрицы *A*.

Собственные значения матрицы A представляют собой корни характеристического уравнения матрицы $|A - \lambda \mathrm{I}|$, где I — единичная матрица. Как полиномиальное уравнение относительно λ, оно имеет N корней. Очередному собственному числу представляется в соответствие собственный вектор, поставленный с точностью до скалярного множителя.

Теорема Фробениуса говорит, что неприводимая неотрицательная матрица A во всех случаях содержит действительное положительное простое собственное значение λ_{max}. При этом модули любых иных характеристических значений не превосходят λ_{max}.

Теорема (Перрон –Фробениуса)

Пусть $A \geq 0$ - неприводимая матрица. Тогда:

1. А имеет действительное положительное простое собственное значение λ_{max}, где по модулю не меньше всякого другого собственного значения матрицы A (комплексными числами могут быть некоторые из них).
2. Собственный вектор A, соответствующий собственному значению λ_{max}, имеет положительные компоненты и единственен.
3. Число λ_{max} удовлетворяет условию

$$\lambda_{max} = \max_{x \geq 0} \min_{1 \leq i \leq n} \frac{(Ax)_i}{Xi} = \min_{x \geq 0} \max_{1 \leq i \leq n} \frac{(Ax)_i}{Xi} \quad ; \quad x \geq 0$$

— произвольно.

Следствие. Пусть $A \geq 0$ неприводима и пусть $x \geq 0$ произвольно. Тогда корень Перрона удовлетворяет условию

$$\min_{1 \leq i \leq n} \frac{(Ax)_i}{x_i} \leq \lambda_{max} \leq \max_{1 \leq i \leq n} \frac{(Ax)_i}{x_i}$$

λ_{max} называют корнем Перрона (в обобщённом варианте главным собственным значением) матрицы A, а парный ему собственный вектор w — главным собственным вектором.

Подобным образом, находить приоритеты элементов необходимо следующим образом: формируется характеристическое уравнение матрицы А, в числе корней полученного уравнения берется самое большое, исчисляется соответствующий собственный вектор w, числа вектора нормализуются. Формирование и вычисление уравнений подобного типа, применимое к каждой матрице в методе анализа иерархий — довольно кропотливый и тяжелый процесс.

В МАИ задано применять иные средства расчетов главного собственного вектора матрицы, где единственный их них определяется на следствии к

теореме Перрона-Фробениуса. Условие примитивности обязательно к выполнению в данном случае матрицы.

Если неприводимая неотрицательная матрица имеет всего h характеристических чисел: $\lambda_1, \lambda_{2,\ldots,} \lambda_h$, то такая матрица называется примитивной при $h = 1$ (главное собственное значение по модулю является единственным $\lambda_1 = \lambda_{max}$) при $h > 1$ матрица называется импримитивной.

Основная теорема МАИ формулируется для примитивных матриц.

Теорема. Для примитивной матрицы A $\lim_{k\to\infty} \frac{A^k e}{|A^k|} = cw, \quad A^k = e^T A^k e,$

где c - постоянная, а w - собственный вектор, соответствующий $\lambda_{max} = \lambda_1$.

Применяя следствие теоремы Перрона, получается наименее трудозатратный (во время компьютерной реализации) метод расчета вектора приоритетов матрицы парных сравнений: возводя матрицу в большие степени, рассчитываются суммы элементов строк матрицы-результата, а суммы, который получились в результате нормализуются. Выше изложенный метод предоставляет возможность определить главный собственный вектор матрицы A не используя характеристического уравнения, что значительно упрощает процесс его расчета.

Таким образом, анализ метода анализа иерархий с использованием парных сравнений показал, что для решения задачи принятия решений в гостиничном бизнесе он является наиболее приемлемым.

2.2 Математическое решение задачи принятия решения в гостиничном бизнесе

Целью задачи является выбор более релевантной заявки на бронирование гостиницы. Необходимо выбрать более приоритетную заявку учитывая все критерии.

Было выявлено пять основных критериев эффективности о выборе заявки на бронирование:

- количество дней проживания по заявке (зачатую гости не остаются на весь забронированный период в гостинице, что означает определенный риск в потере прибыли, так как из-за существующей брони приходится отказывать другим желающим забронировать номер на занятый период);
- категория номера (существует шесть категорий номеров, отличающихся количеством спальных мест, площадью и комфортабельностью номера, наличием дополнительных условий комфорта, а также ценой). Категории номеров по международному регламенту:

1. ЧКД – четвертая категория двухместный (эконом двухместный);
2. ЧКО – четвертая категория одноместный (эконом одноместный);
3. ТКД – третья категория двухместный (стандарт двухместный);
4. ТКО – третья категория одноместный (стандарт одноместный);
5. ВК – вторая категория (комфорт);
6. ПК – первая категория (полулюкс);
7. ПКП – первая категория плюс (люкс).

- Специальный тариф – тариф, выставленный на ту или иную категорию номера, отличающийся от основного прейскуранта благодаря скидочной системе по договору с приезжающей стороной.
- Количество человек, указанных в заявке – от количества человек и категории номера, прописанных в заявке, зависит количество спальных мест и номеров соответственно. Данная информация необходима для отслеживания состояния номерного фонда.
- Оценка приезжающей стороны – оценка формируется автоматически программой Hotel_3.pro за счет истории проживания в предыдущие заезды. Таким образом исходя из полученных штрафов в связи с порчей имущества, просрочки оплаты за проживание и прочих фиксируемых данных производится оценка желательности гостя по пятибалльной шкале.

Пусть в службу бронирования потупило три заявки:

1. Ассоциация спорта:

- Количество дней проживания по заявке: 21;
- Категория номера: ЧКО, ЧКД;
- Специальный тариф: 900 руб./сутки (отсутствует питание в стоимости проживания);
- Количество проживающих гостей: 14 человек;
- Оценка приезжающей стороны: 2 (систематический выезд раньше указанного срока, порча гостиничного имущества на постоянной основе).

2. Театр им. Плеханова:

- Количество дней проживания по заявке: 14;
- Категория номера: ВК;
- Специальный тариф: 3000 руб./сутки;
- Количество проживающих гостей: 10 человек;
- Оценка приезжающей стороны: 3 (нарушение спокойствия других постояльцев, порча гостиничного имущества).

3. ИП Иванов:

- Количество дней проживания по заявке: 29;
- Категория номера: ТКО;
- Специальный тариф: 2000 руб./сутки;
- Количество проживающих гостей: 25 человек;
- Оценка приезжающей стороны: 4.

Необходимо вычислить более приоритетную заявку на бронирование по данным критериям.

Устройство данной задачи представляется в образе иерархической структуры, изображённой на Рисунке 1.

Выполним попарное экспертное сравнение элементов всех уровней иерархии.

Согласованность суждения оценивается индексом однородности (индексом согласованности) или отношением однородности (отношением согласованности) при использовании приведенных ниже формул:

$$ИО = ИС = \frac{\lambda_{max} - n}{n - 1}$$

$$ОО = ОС = \frac{ИО}{М(ио)}$$

М(ио) - среднее значение индекса однородности случайным образом получившейся матрицы парных сравнений, основанное на экспериментальных данных. Значение *М(ио)*- это табличная величина, входным параметром выступает размерность матрицы (таблица 1).

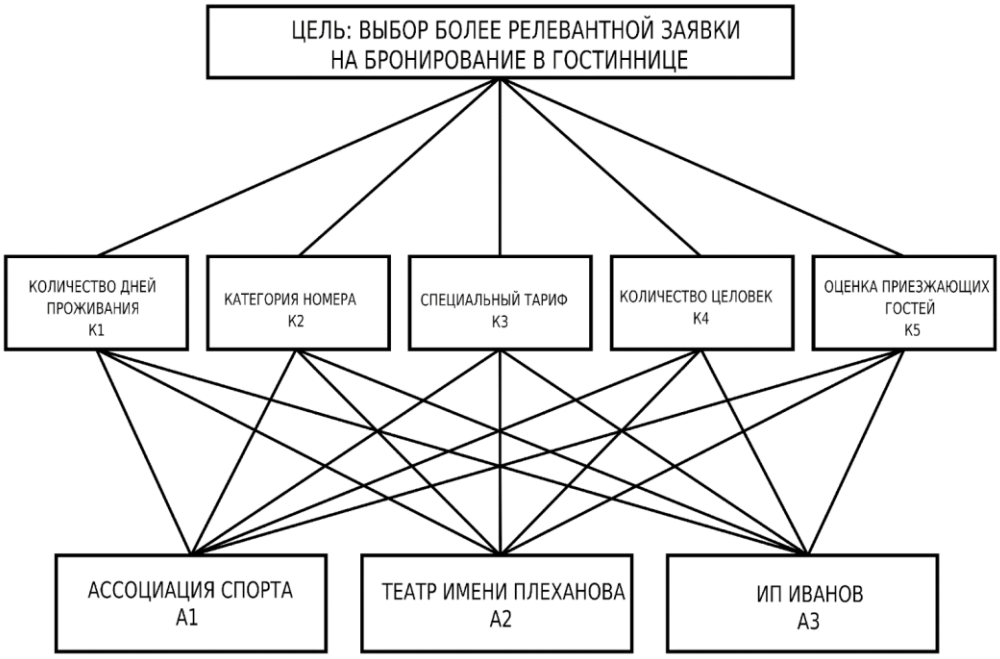

Рисунок 1 - Иерархическая структура проблемы

Таблица 1 -Среднее значение индекса однородности в зависимости от порядка матрицы

N	1	2	3	4	5	6	7	8	9	10	11
М(ио)	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51

Допустимым значением применяется ОО ≤ 0.1. В случае, когда матрица парных сравнений ОО > 0.1, констатирует о крупном нарушении логики суждений, сделанным экспертом во время заполнении матрицы, в этом случае

эксперту рекомендуется переоценить данные, которые были использованы для составления матрицы, с целью повышения однородности.

Иерархический синтез применяется с целью вычисления собственных векторов матриц попарных сравнений альтернатив весами критериев, содержащихся в иерархии, а также в случаях нахождения суммы по всем соответствующим взвешенным элементам собственных векторов нижнего уровня иерархии. Ниже разбирается алгоритм иерархического синтеза, учитывающий обозначения примененных выше.

Найдем векторы приоритетов W_i сравнительно с последним уровнем иерархии. С этой целью составляем матрицы парных сравнений $[E_i]$ и выявляем максимальные собственные значения каждой матрицы (с целью оценки однородности суждений) и главные собственные вектора (приоритеты).

Для уровней иерархии, находящихся выше обработка матрицы происходит аналогичным способом. Эти матрицы, относительно элементов вышележащего уровня, определяют приоритетность элементов выбранного иерархического уровня

Таблица 2 - Матрица сравнения для первого уровня иерархии

	дни	категория	спец.тариф	кол-во человек	Оценка
Дни	1	3	1	1	3
Категория	$^1/_3$	1	3	1	1
спец.тариф	1	$^1/_3$	1	3	1
кол-во человек	1	1	$^1/_3$	1	3
Оценка	$^1/_3$	1	1	$^1/_3$	1

Нормированный собственный вектор:

$W = (0.28; 0.208; 0.209; 0.186; 0.117)$

$\lambda max = 5.90$

$UC=\frac{5.90-5}{5-1}=0.225$

$ОС = 0.225/1.12 = 0.201$

Таблица 3 - Матрица сравнения для второго уровня иерархии для критерия «дни»

	Ассоциация спорта	Театр им. Плеханова	ИП Иванов
Ассоциация спорта	1	3	$^1/_5$
Театр им. Плеханова	$^1/_3$	1	$^1/_7$
ИП Иванов	5	7	1

Собственный вектор: $V = (0.26; 0.11; 1)$

Нормированный собственный вектор: $W = (0.19; 0.0803; 0.73)$

$\lambda max = 3.06$

$UC=\frac{3.06-3}{3-1}=0.03$

$ОС = 0.03/0.58 = 0.0517$

Таблица 4 - Матрица сравнения для второго уровня иерархии для критерия «категория»

	Ассоциация спорта	Театр им. Плеханова	ИП Иванов
Ассоциация спорта	1	$^1/_5$	$^1/_3$
Театр им. Плеханова	5	1	3
ИП Иванов	3	$^1/_3$	1

Собственный вектор: V=0.5; 2.47; 1

Нормированный собственный вектор: W=0.126; 0.622; 0.252

$\lambda_{max}=3.04$

$UC=\frac{3.04-3}{3-1}=0.02$

OC=0.02/0.58=0.0345

Таблица 5- Матрица сравнения для второго уровня иерархии для критерия «спец. тариф»

	Ассоциация спорта	Театр им. Плеханова	ИП Иванов
Ассоциация спорта	1	$^1/_5$	$^1/_3$
Театр им. Плеханова	5	1	3
ИП Иванов	3	$^1/_3$	1

Собственный вектор: $V = 0.5; 2.47; 1$

Нормированный собственный вектор: $W = 0.126; 0.622; 0.252$

$\lambda max = 3.04$

UC=$\frac{3.04\text{-}3}{3\text{-}1}$=0.02

$OC = 0.02/0.58 = 0.0345$

Таблица 6 - Матрица сравнения для второго уровня иерархии для критерия «кол-во человек»

	Ассоциация спорта	Театр им. Плеханова	ИП Иванов
Ассоциация спорта	1	2	$^1/_3$
Театр им. Плеханова	$^1/_2$	1	$^1/_5$
ИП Иванов	3	5	1

Собственный вектор: $V = 0.35; 0.19; 1$

Нормированный собственный вектор: $W = 0.227; 0.123; 0.649$

$\lambda_{max} = 3$

UC=$\frac{3\text{-}3}{3\text{-}1}$=0

$OC = 0/0.58 = 0$

Таблица 7 - Матрица сравнения для второго уровня иерархии для критерия «оценка»

	Ассоциация спорта	Театр им. Плеханова	ИП Иванов
Ассоциация спорта	1	$^1/_3$	$^1/_5$
Театр им. Плеханова	3	1	$^1/_2$
ИП Иванов	5	2	1

Собственный вектор: $V = 0.19; 0.53; 1$

Нормированный собственный вектор: $W = 0.11; 0.308; 0.581$

$\lambda_{max} = 3$

UC=$\frac{3\text{-}3}{3\text{-}1}$=0

$OC = 0/0.58 = 0$

3. Выполняем иерархический синтез. По порядку высчитываем вектора приоритетов альтернатив W_E^A относительно элементов E_j^i, лежащих на всех иерархических уровнях. Расчет векторов приоритетов выполняется от нижних уровней к верхних, учитывая конкретные связи между элементами, относящихся различным уровням. Нахождение выполняется с помощью перемножения соответствующих векторов и матриц

$$\begin{pmatrix} 0{,}19 & 0{,}103 & 0{,}103 & 0{,}227 & 0{,}11 \\ 0{,}0803 & 0{,}638 & 0{,}638 & 0{,}123 & 0{,}308 \\ 0{,}73 & 0{,}258 & 0{,}258 & 0{,}649 & 0{,}581 \end{pmatrix} \begin{pmatrix} 0{,}28 \\ 0{,}209 \\ 0{,}21 \\ 0{,}184 \\ 0{,}117 \end{pmatrix} = \begin{pmatrix} 0{,}150995 \\ 0{,}348474 \\ 0{,}499895 \end{pmatrix}$$

Максимальным элементом в итоговой матрице получился **0.499**. Следовательно, наиболее важным параметром при выборе будет являться А3 – ИП Иванов. Из этого следует, что выбор заявки на бронирование от компании ИП Иванов будет являться более приоритетным для гостиницы с точки зрения прибыли.

Метод анализа иерархий характеризуется многоступенчатым подходом, при котором первый и самый важными шагами к целям является определение исследуемой системы, а также иерархически структурированный список. Верхний уровень состоит из единственной цели системы. Следующие слои систематически приводят к группам логически связанных атрибутов - субатрибуты (древовидная структура). Следующие шаги относятся к распределению приоритетных весов для атрибутов на каждом уровне иерархии. Весовые коэффициенты должны быть определены последовательно путем попарного сравнения соответствующих критериев, в силу чего использование матриц настоятельно рекомендуется. Кроме того, стандартизованная схема оценки (l = в равной степени важный / предпочтительный; 3 = немного важнее; 5 = более важно; 7 = гораздо важнее; 9 = абсолютно более важно). Следующий шаг включает пошаговое нахождение собственного вектора каждой базовой матрицы оценки и затем нормируя его на сумму до I. Аналогичные вычисления должны быть выполнены для матриц, относящихся к более высоким уровням в иерархии системы. Наконец, составные веса должны быть определены путем агрегирования весов по иерархии. Результатом является нормализованный вектор общего приоритета. Аналогичные исследования могут проводиться на основе модифицированных весов (анализ чувствительности) или для альтернативных (например, существующих) систем.

Расчеты, которые должны быть сделаны для исследований МАИ, обычно оказываются достаточно сложными, они будут призывать к использованию специальных пакетов программного обеспечения.

Вывод по главе:

Общие положения задачи, решаемой при помощи МАИ, которые универсальны и легко проецируемы на гостиничный бизнес. Проведенный анализ решения задачи позволил сформулировать задачу в общем виде и предложить алгоритм ее решения, что в дальнейшем может быть переложено для реализации программного продукта.

ГЛАВА 3 РАЗРАБОТКА ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ПРИНЯТИЯ РЕШЕНИЙ С НЕЧЕТКОЙ ИСХОДНОЙ ИНФОРМАЦИЕЙ

Методы принятия решений с множественной адаптацией приобретают признание. Среди различных подходов к аналитическому иерархическому процессу (AHP) представляет собой мощное и легко понятное средство для систематизации. В области делового администрирования аналитический процесс особенно подходит для анализа и обоснования интеграционного компьютерного производства.

В этой главе описывается реализуемый программный продукт, предназначенный для анализа и выявления лучшей альтернативы среди множества заявок на бронирование, основополагающим методом которой представляется комбинаторным алгоритмом бинарных отношений.

3.1 Реализация алгоритма и модели принятия решения в гостиничном бизнесе в условиях неопределенности

Приступим к первой стадии разработки проекта - Анализ требований проекта.

Все начинается со сбора информации о проекте, с целью того, чтобы уменьшить период и ресурсы на корректировку погрешностей и доработку проекта. Все без исключений прибывающие сведения необходимо изучить и классифицировать. Результатом этой стадии должно быть формирование детальной спецификации, соответствующей всем условиям клиента. По итогам нашей работы обнаружились последующие условия:

- приложение должно работать в Desktop, без возможности web-подключения;
- приложение имеет тип программного модуля на основную рабочую программу Hotel_3.pro;
- программный модуль должен быть разработан на языке Java;

- программный модуль должен иметь графический-интерактивный интерфейс;
- дизайн интерфейса должен соответствовать цветовой гамме и стилистике основной рабочей программе Hotel_3.pro;
- удобный и интуитивно-понятный интерфейс;
- отображение списка заявок с названиями организаций для сравнения со всеми их критериями (а именно - количество дней проживания, категория номера, специальный тариф, количество человек, оценка);
- окно ввода новой заявки и выбора уже введенной заявки.

Со второй стадии начинается проектирование.

Для того, чтобы определиться с клиентом начет образа программы и границами проекта, составим примерный скетч программы, который соответствует требованиям заказчика, и который будет согласован с ним.

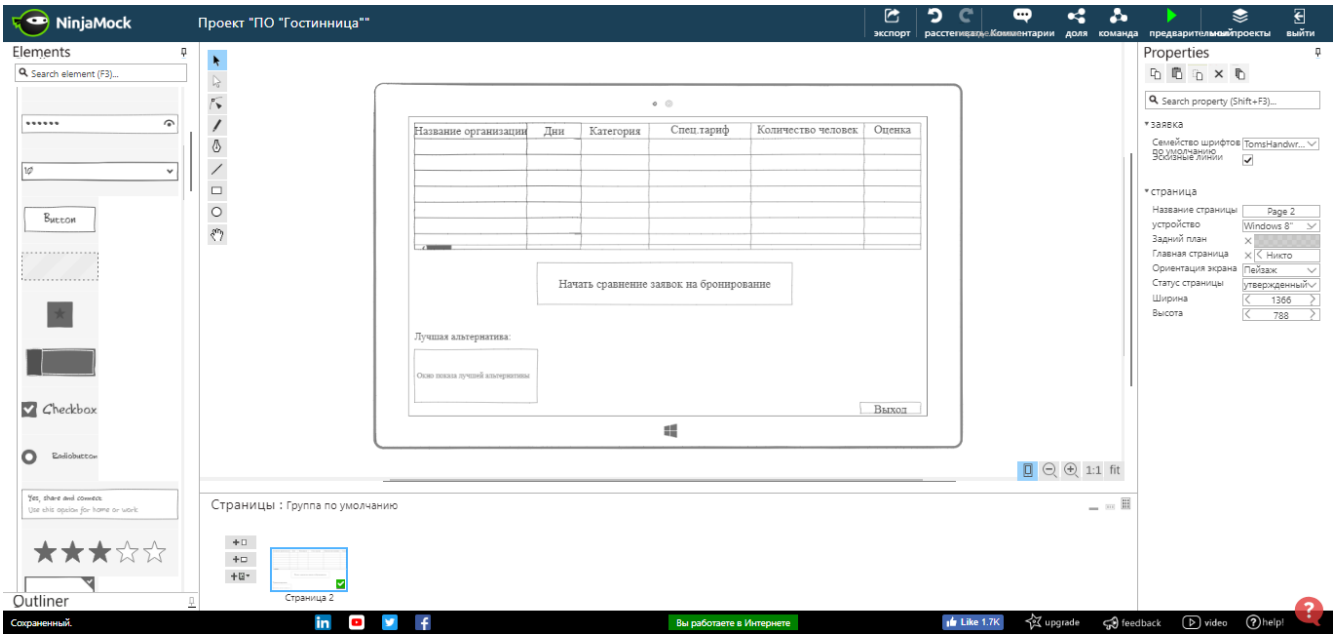

Рисунок 2 - Графический проект образа программы, одобренного заказчиком.

Так же на данном этапе последовало уведомление заказчика с его требованиями к тому, каким образом будет поддерживаться программа, а именно:

- выделение контактных данных для консультаций;
- обучение пользователей программы;

- устранение проблем и ошибок, возникающих в работе ПО.

Третьим этапом является реализация ПО.

После согласования условий поддержки, процесс разработки переходит в сторону формирования макета, который поможет установить архитектуру программы в общем виде. С этой целью необходимо выяснить следующее:

1. Способ получения и обработки данных на входе;
2. Форма, в которой будут предоставлены выходные данные.

Данные будут записываться непосредственно пользователем программы с помощью клавиатуры. Сохраняться эти данные будут в SQL-таблицу, а обработка будет реализоваться с помощью непосредственного кода программы на языке Java.

После создания прототипа в графическом интерфейсе JavaFX программы в и написания кода разметки для нее на fxml, приступаем к определению и реализации архитектурного решения программы. Для этого перейдём к логическому моделированию.

Для этого, чтобы создать логическую модель архитектуры программы используется унифицированный язык UML, который дает возможность реализовать исследование с различных точек зрения.

Начнем с диаграммы альтернатив применения. Она дает возможности для предоставления функциональной структуры систем, не акцентируя внимание на деталях ее реализации.

При имеющейся ситуации для решения имеем следующего и единственного актера:

- Администратор гостиницы.

Разрабатываемая модель вычисляет лучшую альтернативу для брони номеров в гостинице, согласно определенным критериям. На основании этого возможно выявить следующие моменты, которые обязаны быть реализованы в модели:

Таблица 8 - Прецедент модели и его описание

Прецедент	Краткое описание
Заполнение SQL-таблицы	Таблица заполняется данными для сравнения и выявления лучшей альтернативы из представленных заявок на бронирование
Сохранение новых организаций	Для получения части данных нужно выбрать организацию от которой поступила заявка
Выбор организаций	Из списка организаций надо выбрать ту от которой поступила заявка
Ввод критериев	Для сравнения альтернатив требуются критерии часть которых вводится вручную
Сравнение альтернатив	Взаимодействую с полученными данными, программа выполняет поставленную задачу и вычисляет лучшую альтернативу для брони.
Вывод результата более приоритетной заявки на бронирование	Для получения конечных результатов расчетов программа выводит результат и его значение в отдельное окно

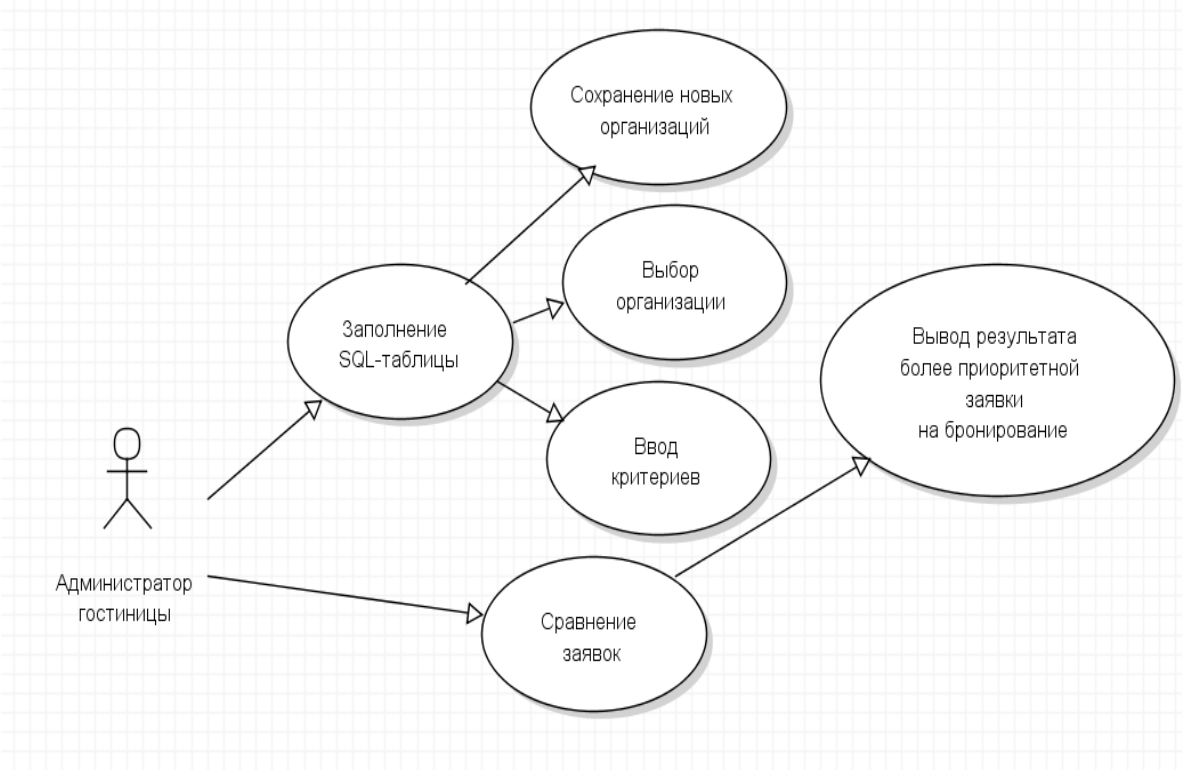

Рисунок 3 - Диаграмма вариантов использования

Для представления классов системы, их атрибутов и взаимосвязей необходимо построить диаграмму классов.

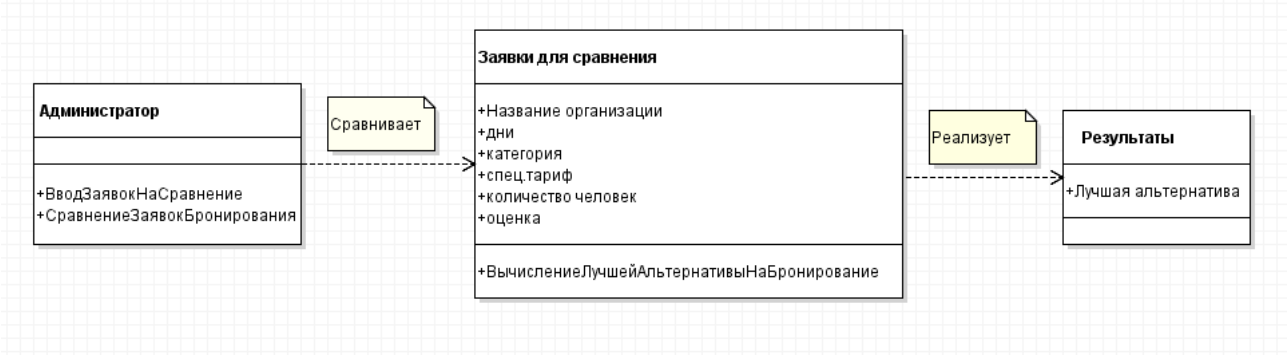

Рисунок 4 - Диаграмма классов приложения, вычисляющего более приоритетную заявку для бронирования в условиях неопределенности

На представленной диаграмме отображено три класса.

Спецификация классов:

- Администратор — класс, осуществляющий ввод заявок на сравнение;
- Заявки для сравнения — класс, вычисляющий лучшую альтернативу на бронирование;
- Результат — класс, выдающий лучшую альтернативу.

Для отображения динамических аспектов системы, следует построить диаграмму последовательностей. Данная диаграмма строится с целью демонстрации жизненного цикла модели и нахождения решения задачи.

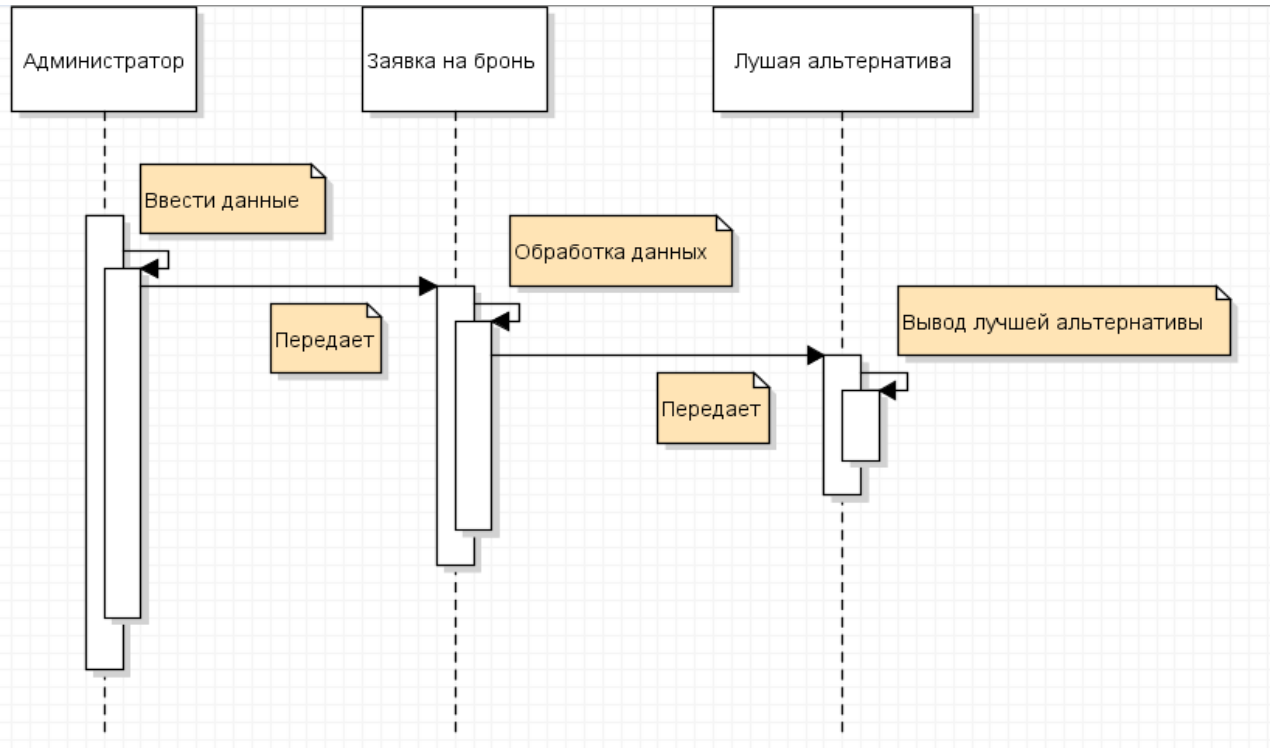

Рисунок 5 - Диаграмма последовательности процессов по выбору лучшей альтернативы

Администратор гостиницы вводит данные для сравнения, поступившие ему из заявки. Они передаются в программу, которая вычисляет решение задачи и передает ответ в вывод лучшей альтернативы.

Логическое моделирование будущей программы дало понимание того, как она должна выглядеть и быть реализована. Имея будущую структуру программы можно приступить к ее реализации.

Для разработки программы была установлена среда разработки IntelliJ IDEA. Для написания кода и отладки программы использовался язык Java. JavaFX Scene Builder использовался для создания и редактирования интерактивного графического интерфейса пользователя.

Разработка начинается с создания «скелета» программы, где импортируются необходимые библиотеки, объявляются и инициализируются переменные, определяются методы программы и выполняются математические преобразования данных, для получения конечного результата.

```
public class Main extends Application {

    private static Stage stage;
    private static Scene root;
    private static Criterion c[] = new Criterion[6];
    private static String kr[] = {"Время проживания", "Категория номеров", "Спец.тариф", "Кол-во человек", "Оценка"};
    private static String kr2[] = {"Ассоциация спорта", "Театр им.Плеханова", "ИП Иванов"};
    private static double Q1, Q2, Q3;
    private static DiagramController diagramController;

    public static void calcQ(){
        double[] qCriterionK1 = c[1].getqCriterion();
        double[] qCriterionK2 = c[2].getqCriterion();
        double[] qCriterionK3 = c[3].getqCriterion();
        double[] qCriterionK4 = c[4].getqCriterion();
        double[] qCriterionK5 = c[5].getqCriterion();
        double[] qCriterion = c[0].getqCriterion();
        Q1 = qCriterionK1[0] * qCriterion[0] + qCriterionK2[0] * qCriterion[1] + qCriterionK3[0] * qCriterion[2] + qCriterionK4[0] * qCriterion[
        Q2 = qCriterionK1[1] * qCriterion[0] + qCriterionK2[1] * qCriterion[1] + qCriterionK3[1] * qCriterion[2] + qCriterionK4[1] * qCriterion[
        Q3 = qCriterionK1[2] * qCriterion[0] + qCriterionK2[2] * qCriterion[1] + qCriterionK3[2] * qCriterion[2] + qCriterionK4[2] * qCriterion[
    }

    public static double getQ1() {
        return Q1;
    }

    public static double getQ2() {
        return Q2;
    }

    public static double getQ3() {
        return Q3;
    }
```

Рисунок 6. - Снимок экрана фрагмента кода с частью проектирования скелета программы

Следующим этапом является создание графического интерфейса пользователя. С помощью JavaFX Scene Builder (рис.7.), а после пишется код для обработки элементов графического интерфейса, таких как таблица вывода информации об организациях, являющихся потенциальными клиентами, из-за которых нужно решить задачу с выявлением приоритетной заявки, а также окно ввода новых организаций, кнопки обработок сравнения организаций по критериям и поля вывода лучшей альтернативы из предложенных организаций.

На последнем этапе прописывается код графического интерфейса (рис.7). Это необходимо для того, чтобы все части графического интерфейса отвечали требованиям к необходимому для них функционалу.

```java
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;

public class Controller {

    @FXML
    private TableView table, leftTable;
    @FXML
    private Slider slider;
    @FXML
    private RadioButton rb1, rb2, rb_all;
    @FXML
    private Label lambdaLabel, ISLabel, OSLabel;
    private int cell_i, cell_j;
    private boolean reversed = false;

    private void initTable(){
        TableColumn tc[] = new TableColumn[5];
        for(int i = 0; i < 5; i++){
            tc[i] = new TableColumn(((Integer)(i+1)).toString());
            tc[i].setMinWidth(100);
            tc[i].setMaxWidth(100);
            tc[i].setCellValueFactory(new PropertyValueFactory<>("k" + ((Integer)(i+1)).toString()));
            tc[i].setSortable(false);
        }
        table.getColumns().setAll(tc[0], tc[1], tc[2], tc[3], tc[4]);
        table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
        table.getSelectionModel().setCellSelectionEnabled(true);
```

Рисунок 7 - Проработка кода графического интерфейса

Последним этапом является подключение базы данных на MySQL и написание кода к ней для того, чтобы сохранять данных об организациях и оперативно пользоваться ими при выполнении поставленной задачи.

```java
package javaapplication1;
import java.sql.*;

public class Main {

    public static void main(String[] args) throws SQLException {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/db_name",
            "user", "password");

            if (conn == null) {
                System.out.println("Нет соединения с БД!");
                System.exit(0);
            }

            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM users");

            while (rs.next()) {
```

Подключение производится с помощью Java DataBase Connectivity.

Рисунок 8– Фрагмент кода подключения базы данных

Далее подробно рассматривается работающий функционал программы выбора более приоритетной заявки.

Основное окно программы (рисунок 9) разделено на две рабочие зоны, такие как таблица данных, которая включает в себя поля «Название организации», а также значения основных критериев для сравнивания: «Дни проживания по заявке», «Категория номера», «Специальный тариф», «Количество человек», «Оценка приезжающего гостя». Дня удобства и более эстетического внешнего вида названия критериев были записаны сокращенно. Поле «Лучшая альтернатива» является полем вывода расчетов, произведенных программой сравнения заявок. В главном окне программы так же присутствуют две функциональные кнопки. Первая кнопка «Начать сравнение заявок на бронирование» запускает программу, которая начинает сравнивание всех критериев, введенных в таблицу. Вторая кнопка «Выход» закрывает всю программу.

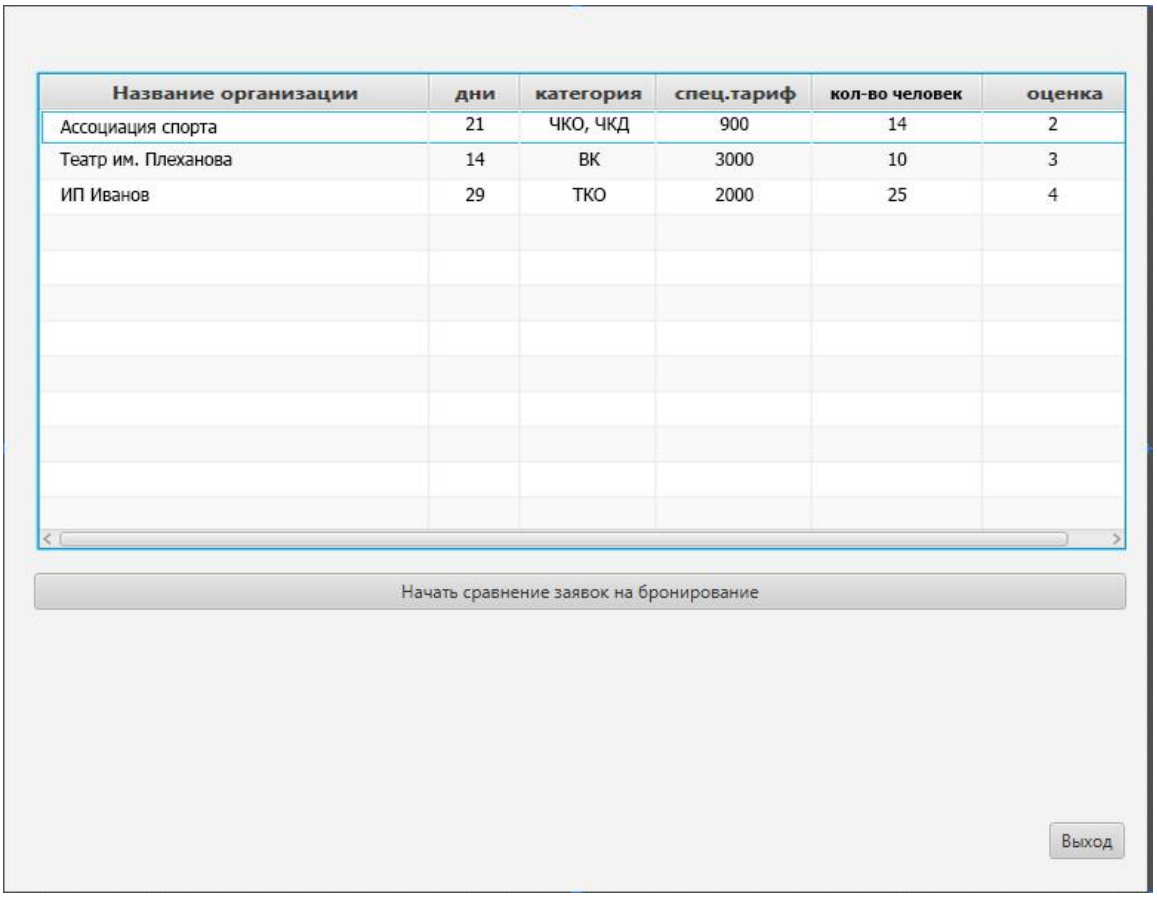

Рисунок 9 - Основное поле программы

Возможность добавления данных в таблицу неограниченна и открывается двойным щелчком левой кнопки мыши по таблице (рисунок 10.). В всплывающем окне заполняются данные по заявке бронирования от той или иной организации. Интерактивное меню выбора предоставляет выбор организаций из всплывающего списка. Дальше представлена возможность добавления основных критериев сравнивания заявок на бронирование гостиниц. Все поля заполняются вручную пользователем, кроме последнего. Критерий «Оценка гостей» вносится в таблицу автоматически из базы данных программы Hotel_3.pro (основная рабочая программа в ГК «Звезда Жигулей»). Данный критерий не подлежит изменению (исключается погрешность из-за личной заинтересованности пользователя) и является основным при отборе. После заполнения всех необходимых полей кнопка «Добавить организацию» вносит заявку для сравнивания в таблицу. Кнопка «Отмена» позволяет отменить все действия и закрыть окно «Организация для сравнения».

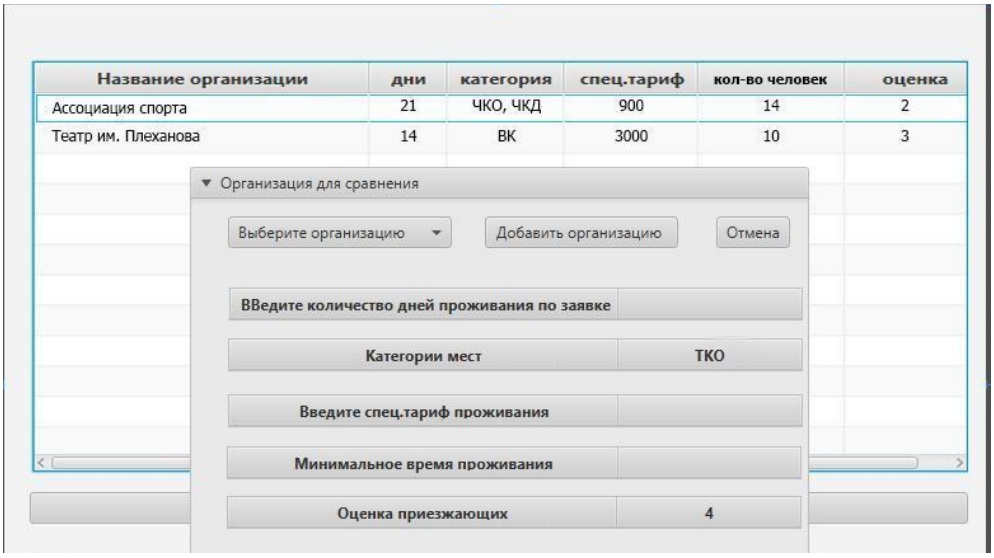

Рисунок 10. - Окно добавления организации

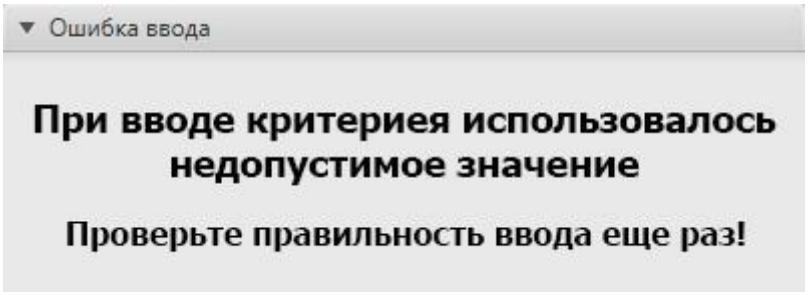

Рисунок 11 - Сообщение об ошибке

При нажатии кнопки «Начать сравнивание заявок» начинается обработка данных из таблицы. Во время обработки данных появляется окно ожидания (рисунок 12).

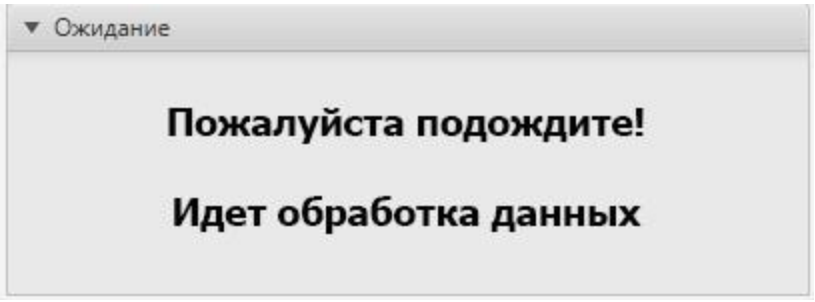

Рисунок 12. - Окно ожидания отклика программы

После завершения обработки всех данных и вычисления более приоритетной заявки на бронирование появляется окно, в котором показывается итоговый результат вычисления (рисунок 13).

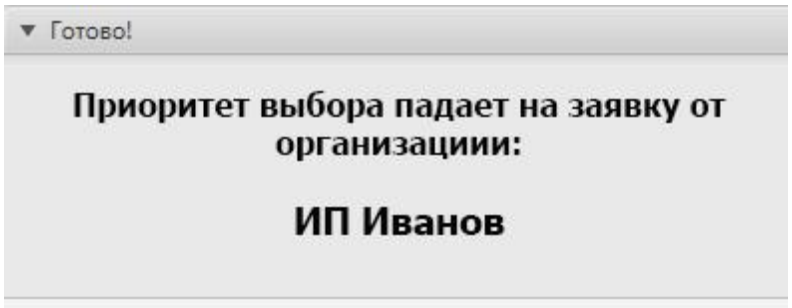

Рисунок 13. - Окно вывода итогов работы программы

Таким образом, была описана реализации программного продукта, которая включила описание проектирование на основе применения диаграмм UML.

3.2 Тестирования программы

Тестирование программного продукта завершающая часть реализации программного решения задачи о принятии решения в гостиничном бизнесе. В процессе тестирования были выявлены и устранены недочеты разработанного программного продукта.

В связи с функционалом продукта тестирование проводилось вручную методом черного ящика.

Тестирование по черному ящику предполагает, что тестировщик проверяет функциональные и нефункциональные компоненты без доступа к внутренней структуре компонентов программы.

Таблица 9 – Тест-кейсы для тестирования программы.

Действия пользователя	**Результат программы**
Дважды кликнуть левой кнопкой мыши на SQL-таблице	Открывается форма «Добавление организации»
При вводе данных в поля «Дни», «Спец.Тариф», «Кол-во чел.» ввести отрицательное, нулевое и буквенное значения	Поле ввода численного или буквенного значения выводит сообщение об ошибке, в котором прописаны правильные числовые значения для ввода в данной категории
При вводе данных в поле «Категория» ввести отрицательное, нулевое и положительное числовые значения	Поле ввода числового значения выводит сообщение об ошибке, в котором прописаны правильные буквенные значения для ввода
Нажать кнопку «Добавить» в форме «Добавление организации»	В таблице добавляется новая организация с ее данными
Нажать кнопку «Отмена» в форме «Добавление организации»	Форма «Добавление организации» закрывается, в таблице новых данных

	не появляется
Запустить сравнение заявок на бронирование	При запуске сравнивания заявок на бронирование должно выводится системное сообщение ожидания
Нажать кнопку «Выход» на основной форме программы	Программа закрывается

При тестировании основные проблемы возникали при работе с графическим интерфейсом. Все ошибки и недочеты были устранены и функционирование программы на данный момент происходит в штатном режиме.

После внедрения программного продукта в систему Hotel_3.pro в гостинице «Звезда Жигулей» доступ к программе был предоставлен группе пользователей в количестве шести человек. Программа функционировала в течении двух недель без перерыва с момента запуска. Ошибок в ходе работы за данный период выявлено не было.

Выводы по главе.

В ходе выполнения работы был составлен и применен алгоритм создания программы для решения задачи гостиничного бизнеса в условиях неопределенности.

Была создана программа на базе языка Java в среде разработки IntelliJ IDEA

Итоговое тестирование программы дефектов не выявила.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке математической модели и алгоритма принятия решений в гостиничном бизнесе в условиях неопределенности В ходе выполнения выпускной квалификационной работы был произведен анализ предметной области, бизнес-процессы ГК «Звезда Жигулей»» по учету и обработке заявок, а также взаимодействие персонала компании с клиентами в целом. По результатам анализа были выделены основные процессы, нуждающиеся в автоматизации для улучшения эффективности работы и повышения удобства рабочего процесса. На основе произведенного анализа было принято решение о необходимости создания и внедрения программного продукта, обеспечивающего автоматический отбор более релевантных заявок на бронирование. Для определения необходимого функционала была смоделирована логическая модель структуры будущей программы для проверки ее логического соответствия предъявленным функциональным требованиям. По результатам работы была создана программа на базе языка Java в среде разработки IntelliJ IDEA. Были выявлены основные требования к функционалу и графическому интерфейсу программы, чтобы обеспечить необходимую степень автоматизации службы бронирования. После внедрения модуля в работу компании будет автоматизирована работа службы бронирования (учет и обработка заявок на бронирование). Для компании, использование внедренной системы позволит оптимизировать ресурсы персонала для решения поставленных задач и предоставления услуг и минимизировать количество ошибочных действий и решений.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Учебники и учебные пособия

1. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. — М.: НИЦ ИНФРА-М, 2013. — 283 с.

2. Вигерс, К. Разработка требований к программному обеспечению. 3-е изд., дополнительное / К. Вигерс, Д. Битти., Пер. с англ. – М.: Издательство «Русская редакция» ; СПб.: БХВ-Петербург, 2014. – 736 с.

3. Волович, М.Е., Дерюгина, О.А. Верификация UML-моделей программных систем Cloud of science. — М., 2015. — 9 с.

4. Гайдамакин, Н.А. Автоматизированные информационные системы, базы и банки данных. Вводный курс: учебное пособие / Н.А. Гайдамакин. - М.: Гелиос АРВ, 2013 – 430 с

5. Гвоздева В. А., Лаврентьева И. Ю. Основы построения автоматизированных информационных систем. М.: ФОРУМ, 2014. - 320с. 11

6. Гонсалвес Э. Изучаем Java EE 7. Издательство: СПб.:Питер, 2014 г. – 640 стр.

7. Голицина, О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. — М.: Формум: ИНФРА-М, 2013. — 352 с.

8. Канеман Д. Принятие решений в неопределенности. Правила и предубеждения. / Д. Канеман, П. Словик, А. Тверский - Гуманитарный центр, 2018. – 536 с.

9. Котляров В.П. Основы тестирования программного обеспечения. / В.П. Котляров, Т.В Коликова – М.: БИНОМ,206-285 с.

10. Хорстманн К.С. Java SE 9. Базовый курс. / Вильямс, 2018. – 576 с

11. Чистов, Д. В. Проектирование информационных систем. Учебник и практикум / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — М.: Юрайт, 2016. — 260 с.

12. Шелухин, О. И. Моделирование информационных систем: учеб. пособие. 004 / О. И. Шелухин. - 2-е изд., перераб. и доп. – М. : Горячая линия - Телеком, 2016. - 516 с.

13.. Ягола А. Обратные задачи и методы их решения. / А. Ягола, В. Янфей, И. Степанова – Бином, 2013. 216 с.0

Электронные ресурсы.

14. SpringerLink: [Электронный ресурс] URL: https://link.springer.com/

15. ScienceDirect: [Электронный ресурс] URL: https://www.sciencedirect.com/

16. Разработка и внедрение информационной системы [Электронный ресурс]: лекция / Национальный Открытый Институт «ИНТУИТ». — М., 2013 URL: http://www.cfin.ru/vernikov/idef/idef0.shtml, свободный (дата обращения 13.01.2017)

Литература на иностранном языке

17 Clemen R. Making Hard Decisions: An Introduction to Decision Analysis. / Belmont CA: Duxbury Press, 1996. – P 349-358

18 Halvorson, K. Content Strategy for the Web. - 2nd Edition, 2015.

19 Nixon R. Learning PHP, MySQL, JavaScript, CSS & HTML5 - 3rd Edition, 2014.

20 Schildt H. Java: The Complete Reference, Ninth Edition. / Oracle, 2014. – P. 1237-1301.

ПРИЛОЖЕНИЕ

Class Main

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TableView;
import javafx.stage.Stage;

public class Main extends Application {

    private static Stage stage;
    private static Scene root;
    private static Criterion c[] = new Criterion[6];
    private static String kr[] = {"Время проживания", "Категория номеров",
"Спец.тариф", "Кол-во человек", "Оценка"};
    private static String kr2[] = {"Ассоциация спорта", "Театр им.Плеханова",
"ИП Иванов"};
    private static double Q1, Q2, Q3;
    private static DiagramController diagramController;

    public static void calcQ(){
        double[] qCriterionK1 = c[1].getqCriterion();
        double[] qCriterionK2 = c[2].getqCriterion();
        double[] qCriterionK3 = c[3].getqCriterion();
        double[] qCriterionK4 = c[4].getqCriterion();
        double[] qCriterionK5 = c[5].getqCriterion();
        double[] qCriterion = c[0].getqCriterion();
        Q1 = qCriterionK1[0] * qCriterion[0] + qCriterionK2[0] * qCriterion[1] +
qCriterionK3[0] * qCriterion[2] + qCriterionK4[0] * qCriterion[3] +
qCriterionK5[0] * qCriterion[4];
        Q2 = qCriterionK1[1] * qCriterion[0] + qCriterionK2[1] * qCriterion[1] +
qCriterionK3[1] * qCriterion[2] + qCriterionK4[1] * qCriterion[3] +
qCriterionK5[1] * qCriterion[4];
        Q3 = qCriterionK1[2] * qCriterion[0] + qCriterionK2[2] * qCriterion[1] +
qCriterionK3[2] * qCriterion[2] + qCriterionK4[2] * qCriterion[3] +
qCriterionK5[2] * qCriterion[4];
    }

    public static double getQ1() {
        return Q1;
```

```
    }

    public static double getQ2() {
        return Q2;
    }

    public static double getQ3() {
        return Q3;
    }

    public static Criterion getCriterion(int x){
        return c[x];
    }

    public static String getKr(int i){
        return kr[i];
    }

    public static String getKr2(int i){
        return kr2[i];
    }

    public static void updateDiagram(){
        diagramController.update();
    }

    public void init() throws Exception{
        c[0] = new Criterion(5);
        for(int i = 1; i < 6; i++) c[i] = new Criterion(3);

Parents.setAim(FXMLLoader.load(getClass().getResource("ParentsFXML/aim.fxml")));

Parents.setTime(FXMLLoader.load(getClass().getResource("ParentsFXML/time.fxml"))
);

Parents.setDiff(FXMLLoader.load(getClass().getResource("ParentsFXML/diff.fxml"))
);

Parents.setCost(FXMLLoader.load(getClass().getResource("ParentsFXML/cost.fxml"))
);

Parents.setMin(FXMLLoader.load(getClass().getResource("ParentsFXML/min.fxml")));
```

```
Parents.setQuality(FXMLLoader.load(getClass().getResource("ParentsFXML/quality.f
xml")));
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("ParentsFXML/diagram.fxml"));
        Parents.setDiagram(loader.load());
        diagramController = loader.getController();
    }

    @Override
    public void start(Stage primaryStage) throws Exception{
        stage = primaryStage;
        stage.setTitle("BKP");
        stage.setWidth(800);
        stage.setHeight(630);
        stage.setResizable(false);
        root = new Scene(Parents.getAim());
        stage.setScene(root);
        stage.show();
    }

    public static void changeScene(Parent parent){
        root.setRoot(parent);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Class Criterion

```
  public class Criterion {

    double[][] criterion;
    double[] wCriterion;
    double rCriterion;
    double[] qCriterion;
    double[] sCriterion;
    double[] pCriterion;
    double hmax;
    double IS;
    double OS;
```

```
    public double getQValue(int i){
        return qCriterion[i];
    }

    public double[][] getCriterion() {
        return criterion;
    }

    public double[] getqCriterion() {
        return qCriterion;
    }

    public void change(int i, int j, double val){
        criterion[i][j] = val;
        criterion[j][i] = 1.0 / val;
        calc();
    }

    public double getValue(int i, int j){
        return criterion[i][j];
    }

    public double getrCriterion() {
        return rCriterion;
    }

    public double getHmax() {
        return hmax;
    }

    public double getIS() {
        return IS;
    }

    public double getOS() {
        return OS;
    }

    public Criterion(int size){
        criterion = new double[size][size];
        for(int i = 0; i < size; i++) for(int j = 0; j < size; j++)
criterion[i][j] = 1;
```

```
        calc();
    }

    public void calc(){
        wCriterion = w(criterion);
        rCriterion = r(wCriterion);
        qCriterion = q(wCriterion, rCriterion);
        sCriterion = s(criterion);
        pCriterion = p(sCriterion, qCriterion);
        hmax = hmax(pCriterion);
        IS = IS(hmax, pCriterion);
        OS = OS(pCriterion, IS);
    }

    private static double[] w(double[][] array) {
        double[] w = new double[array.length];
        for (int i = 0; i < array.length; i++) { //заполняем массив нулями
            for (int j = 0; j < array.length; j++) {
                w[i] = 1;
            }
        }
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array.length; j++) {
                w[i] *= Math.pow(array[i][j], 1d / array.length);
            }
        }
        return w;
    }

    private static double r(double[] array) {
        double r = 0;
        for (int i = 0; i < array.length; i++) {
            r += array[i];
        }
        return r;
    }

    private static double[] q(double[] array, double r) {
        double[] q = new double[array.length];
        for (int i = 0; i < array.length; i++) {
            q[i] = array[i] / r;
        }
```

```
        return q;
    }

    private static double[] s(double[][] array) {
        double[] s = new double[array.length];
        for (int j = 0; j < array.length; j++) { //сумма столбцов
            for (int i = 0; i < array.length; i++) {
                s[j] += array[i][j];
            }
        }
        return s;
    }

    private static double[] p(double[] arrayS, double[] arrayQ) {
        double[] p = new double[arrayS.length];
        for (int i = 0; i < arrayS.length; i++) {
            p[i] = arrayQ[i] * arrayS[i];
        }
        return p;
    }

    private static double hmax(double[] array) {
        double hmax = 0;
        for (int i = 0; i < array.length; i++) {
            hmax += array[i];
        }
        return hmax;
    }

    private static double IS(double hmax, double array[]) {
        double IS;
        IS = (hmax - array.length) / (array.length - 1);
        return IS;
    }

    private static double OS(double[] array, double IS) {
        double OS;
        double SI = 0;
        switch (array.length) {
            case 3:
                SI = 0.58;
                break;
```

```
            case 5:
                SI = 1.12;
                break;
        }
        OS = IS / SI;
        return OS;
    }
}
```

Class Controller

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;

public class Controller {

    @FXML
    private TableView table, leftTable;
    @FXML
    private Slider slider;
    @FXML
    private RadioButton rb1, rb2, rb_all;
    @FXML
    private Label lambdaLabel, ISLabel, OSLabel;
    private int cell_i, cell_j;
    private boolean reversed = false;

    private void initTable(){
        TableColumn tc[] = new TableColumn[5];
        for(int i = 0; i < 5; i++){
            tc[i] = new TableColumn(((Integer)(i+1)).toString());
            tc[i].setMinWidth(100);
            tc[i].setMaxWidth(100);
            tc[i].setCellValueFactory(new PropertyValueFactory<>("k" +
((Integer)(i+1)).toString()));
            tc[i].setSortable(false);
        }
        table.getColumns().setAll(tc[0], tc[1], tc[2], tc[3], tc[4]);
        table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
```

```
        table.getSelectionModel().setCellSelectionEnabled(true);
        table.setItems(getData());
        setLabels();
    }

    private void initLeftTable(){
        TableColumn num = new TableColumn("№");
        num.setMinWidth(50);
        num.setMaxWidth(50);
        num.setCellValueFactory(new PropertyValueFactory<>("num"));
        num.setSortable(false);

        TableColumn name = new TableColumn("Критерий");
        name.setMinWidth(128);
        name.setMaxWidth(128);
        name.setCellValueFactory(new PropertyValueFactory<>("name"));
        name.setSortable(false);

        TableColumn vp = new TableColumn("ВП");
        vp.setMinWidth(50);
        vp.setMaxWidth(50);
        vp.setCellValueFactory(new PropertyValueFactory<>("vp"));
        vp.setSortable(false);

        leftTable.getColumns().setAll(num, name, vp);
        leftTable.setItems(getLeftTableData());
    }

    private ObservableList getLeftTableData(){
        ObservableList data = FXCollections.observableArrayList();
        for(int i = 0; i < 5; i++){
            data.add(new Cell(i, 0));
        }
        return data;
    }

    private ObservableList getData(){
        ObservableList data = FXCollections.observableArrayList();
        for(int i = 0; i < 5; i++){
            data.add(new Cell(i, 0, 5));
        }
        return data;
```

```
    }

    @FXML
    private void onEdit(MouseEvent event){
        cell_i = table.getFocusModel().getFocusedCell().getRow();
        cell_j = table.getFocusModel().getFocusedCell().getColumn();
        int value = (int) Main.getCriterion(0).getValue(cell_i, cell_j);
        if(value < 0) value = (int) Main.getCriterion(0).getValue(cell_j,
cell_i);
        slider.setValue(value);
        if(cell_i == cell_j){
            rb1.setDisable(true);
            rb2.setDisable(true);
            rb_all.setDisable(false);
            slider.setDisable(true);
            rb_all.setSelected(true);
            slider.setValue(1);
        }
        else{
            rb1.setDisable(false);
            rb2.setDisable(false);
            rb_all.setDisable(false);
            if(!rb_all.isSelected()) slider.setDisable(false);
        }
        rb1.setText(Main.getKr(cell_i));
        rb2.setText(Main.getKr(cell_j));
    }

    @FXML
    private void rbClicked(MouseEvent event){
        if(rb1.isSelected()){
            reversed = false;
            slider.setDisable(false);
        }
        else if(rb2.isSelected()){
            reversed = true;
            slider.setDisable(false);
        }
        else if(rb_all.isSelected()){
            slider.setDisable(true);
            slider.setValue(1);
            sliderEvent(event);
```

```
        }
    }

    private void update(){
        TableColumn tc =
table.getFocusModel().getFocusedCell().getTableColumn();
        table.setItems(getData());
        table.getSelectionModel().select(cell_i, tc);
        leftTable.setItems(getLeftTableData());
        setLabels();
    }

    private void setLabels(){
        lambdaLabel.setText("λ = " +
(double)Math.round(Main.getCriterion(0).getHmax() * 1000) / 1000);
        ISLabel.setText("ИС = " +
(double)Math.round(Main.getCriterion(0).getIS() * 1000) / 1000);
        OSLabel.setText("ОС = " +
(double)Math.round(Main.getCriterion(0).getOS() * 1000) / 1000);
    }

    @FXML
    private void sliderEvent(MouseEvent event){
        int i = cell_i, j = cell_j;
        if(reversed){
            i ^= j; j ^= i; i ^= j;
        }
        long value = Math.round(slider.getValue());
        slider.setValue(value);
        Main.getCriterion(0).change(i, j, value);
        update();
    }

    @FXML
    private void initialize(){
        initTable();
        initLeftTable();
    }

    @FXML
    private void onTimeClicked(){
        Main.changeScene(Parents.getTime());
```

```
    }

    @FXML private void onDiffClicked(){
        Main.changeScene(Parents.getDiff());
    }

    @FXML private void onCostClicked(){
        Main.changeScene(Parents.getCost());
    }

    @FXML private void onMinClicked(){
        Main.changeScene(Parents.getMin());
    }

    @FXML private void onQualityClicked(){
        Main.changeScene(Parents.getQuality());
    }

    @FXML private void onOkClicked(){
        Main.updateDiagram();
        Main.changeScene(Parents.getDiagram());
    }

    @FXML
    private void onExit(MouseEvent event){
        System.exit(0);
    }

}
```

Class Parents

```
import javafx.scene.Parent;

public class Parents {

    private static Parent aim;
    private static Parent time;
    private static Parent diff, cost, min, quality;
    private static Parent diagram;

    public static Parent getAim() {
        return aim;
    }
```

```
public static void setAim(Parent aim) {
    Parents.aim = aim;
}

public static Parent getTime() {
    return time;
}

public static void setTime(Parent time) {
    Parents.time = time;
}

public static Parent getDiff() {
    return diff;
}

public static void setDiff(Parent diff) {
    Parents.diff = diff;
}

public static Parent getCost() {
    return cost;
}

public static void setCost(Parent cost) {
    Parents.cost = cost;
}

public static Parent getMin() {
    return min;
}

public static void setMin(Parent min) {
    Parents.min = min;
}

public static Parent getQuality() {
    return quality;
}

public static void setQuality(Parent quality) {
```

```
        Parents.quality = quality;
    }

    public static Parent getDiagram() {
        return diagram;
    }

    public static void setDiagram(Parent diagram) {
        Parents.diagram = diagram;
    }
}
```

Class DiagramController

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.geometry.Side;
import javafx.scene.Parent;
import javafx.scene.chart.PieChart;
import javafx.scene.layout.GridPane;

import java.awt.*;

public class DiagramController {

    private double Q1, Q2, Q3;
    private LabeledPieChart pieChart;
    @FXML
    private GridPane gridPane;

    public void update(){
        Main.calcQ();
        Q1 = Main.getQ1();
        Q2 = Main.getQ2();
        Q3 = Main.getQ3();
        ObservableList<PieChart.Data> data = FXCollections.observableArrayList(
                new PieChart.Data("Асоциация спорта", (double)Math.round(Q1 *
1000) / 1000),
                new PieChart.Data("Театр им.Плеханова", (double)Math.round(Q2 *
1000) / 1000),
                new PieChart.Data("ИП Иванов", (double)Math.round(Q3 * 1000) /
1000)
        );
```

```
        pieChart.getData().clear();
        pieChart.getData().addAll(data);
        pieChart.setLegendSide(Side.RIGHT);
    }

    @FXML
    private void onOkClicked(){
        Main.changeScene(Parents.getAim());
    }

    @FXML
    private void initialize(){
        pieChart = new LabeledPieChart();
        gridPane.add(pieChart, 0, 0);
        update();
    }

}
```

Class LabeledPieChart

```
import java.util.HashMap;
import java.util.Map;
import javafx.collections.ListChangeListener;
import javafx.scene.chart.PieChart;
import javafx.scene.layout.Region;
import javafx.scene.shape.Arc;
import javafx.scene.text.Text;

public class LabeledPieChart extends PieChart {

    private final Map<Data, Text> _labels = new HashMap<>();

    public LabeledPieChart() {
        super();
        this.getData().addListener((ListChangeListener.Change<? extends Data> c)
-> {
            addLabels();
        });

    }

    @Override
    protected void layoutChartChildren(double top, double left, double
```

```
contentWidth, double contentHeight) {
        super.layoutChartChildren(top, left, contentWidth, contentHeight);

        double centerX = contentWidth / 2 + left;
        double centerY = contentHeight / 2 + top;

        layoutLabels(centerX, centerY);
    }

    private void addLabels() {

        for (Text label : _labels.values()) {
            this.getChartChildren().remove(label);
        }
        _labels.clear();
        for (Data vData : getData()) {
            final Text dataText;
            final double yValue = vData.getPieValue();
            dataText = new Text(Double.toString(yValue));
            _labels.put(vData, dataText);
            this.getChartChildren().add(dataText);
        }
    }

    private void layoutLabels(double centerX, double centerY) {

        double total = 0.0;
        for (Data d : this.getData()) {
            total += d.getPieValue();
        }
        double scale = (total != 0) ? 360 / total : 0;

        for (Map.Entry<Data, Text> entry : _labels.entrySet()) {
            Data vData = entry.getKey();
            Text vText = entry.getValue();

            Region vNode = (Region) vData.getNode();
            Arc arc = (Arc) vNode.getShape();

            double start = arc.getStartAngle();

            double size = (isClockwise()) ? (-scale *
```

```
Math.abs(vData.getPieValue())) : (scale * Math.abs(vData.getPieValue()));
            final double angle = normalizeAngle(start + (size / 2));
            final double sproutX = calcX(angle, arc.getRadiusX() / 2, centerX);
            final double sproutY = calcY(angle, arc.getRadiusY() / 2, centerY);

            vText.relocate(
                    sproutX - vText.getBoundsInLocal().getWidth(),
                    sproutY - vText.getBoundsInLocal().getHeight());
        }

    }

    private static double normalizeAngle(double angle) {
        double a = angle % 360;
        if (a <= -180) {
            a += 360;
        }
        if (a > 180) {
            a -= 360;
        }
        return a;
    }

    private static double calcX(double angle, double radius, double centerX) {
        return (double) (centerX + radius * Math.cos(Math.toRadians(-angle)));
    }

    private static double calcY(double angle, double radius, double centerY) {
        return (double) (centerY + radius * Math.sin(Math.toRadians(-angle)));
    }

}
```