СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Туризм является одной из крупнейших и наиболее перспективных отраслей мировой экономики. Туризм играет значительную роль в создании рабочих мест, экспортных поступлений услуг большого ряда стран. Туристическая отрасль подвержена тенденциям мировой экономики. Важнейшими трендами, позволяющими компаниям осваивать новые рынки и бизнес-модели, являются автоматизация и цифровизация процессов управления туристическим бизнесом. Цифровизация предлагает технологии, подходы и инструменты, которые позволяют повысить ценность туристского продукта [1] 0. Использование возможностей цифровых технологий в туристской деятельности существенно увеличивает производительность, что приводит к значимой экономии временных, финансовых и человеческих ресурсов 0, [4].

Благодаря информационным технологиям современные туристические услуги становятся более гибкими и индивидуальными, более привлекательными и доступными для широкого потребителя. Туризм более чем другие отрасли неразделим с информацией, которая несет в себе сведения о предлагаемых турах, туроператорах, услугах, условиях проживания, перемещения, оздоровления и т. д. Оценивая потенциал, особенности и перспективы развития туризма в Туркменистане, исследователи отмечают важность информационной поддержки и необходимость автоматизации бизнес-процессов в сфере туристических услуг [5].

Возможность анализа информации позволяет принять на ее основе единственно правильное решение, обеспечить наилучший выбор услуги согласно индивидуальным требованиям.

Современные автоматизированные информационные системы для организации работы в туристическом бизнесе не только обеспечивают руководителей и менеджеров информацией о различных аспектах деятельности организации для ее своевременной оценки и анализа, но и предоставляют информацию о конкурентоспособности туров в простой и удобной форме.

Целью проекта является автоматизация бизнес-процессов туристической компании.

Объектом является веб-приложение, выполняющее функции создания и управления заказами, учета финансовых операций и учета клиентов.

Для достижения поставленной цели решены следующие задачи:

– выполнен аналитический обзор автоматизированных систем управления в сфере туристических услуг;

– разработаны структура, алгоритм, а также база данных информационной системы поддержки деятельности туристического агентства;

– выполнен расчет надежности разработанной информационной системы;

– проведено функциональное тестирование веб-приложения;

– осуществлено технико-экономическое обоснование разработки.

Целевой аудиторией являются представители туристической индустрии Туркменистана. Разработанная информационная система позволит облегчить процесс управления туристической деятельностью, автоматизирует рутинные процессы, что позволит повысить эффективность работы туристического агентства.

1ИНФОРМАЦИОННАЯ СИСТЕМА ТУРИСТИЧЕСКОГО АГЕНТСТВА

1.1 Анализ предметной области

В индустрии туризма применяются различные информационные технологии, начиная от широко распространенных технологий работы с текстом, электронными таблицами и базами данных до использования специализированных программных продуктов, обеспечивающих автоматизацию работы отдельной туристической фирмы или отеля, и глобальных компьютерных сетей и спутниковых систем навигации [6], [7].

Виды применяемых в туризме информационных технологий и систем приведено на рисунке 1.1

Рисунок 1.1 – Информационные системы в туристической отрасли

Автоматизированные системы управления в туризме – системы управления производственно-обслуживающим процессом в туристических предприятиях и организациях. Они служат для создания эффективной туристической структуры, позволяющей обеспечить комфортные условия труда персонала за счет его профессионального развития и управления его деловой карьерой. Функциональные возможности этих систем должны обеспечивать ввод, редактирование и хранение информации о турах, клиентах, о состоянии заявок; предусматривать вывод информации в форме различных документов; расчет стоимости туров с учетом курса валют, скидок, контроль оплаты туров,

формирование билетов; перевод экспорт данных в другие программные продукты (*Pdf, Word*, *Excel*) и прочие функции [6].

Информационные технологии в туризме представляют собой взаимосвязь компьютерных и коммуникационных технологий. Это позволяет различать несколько уровней автоматизации работы предприятий турбизнеса. Необходимый уровень автоматизации определяется объемами, с которыми сталкивается компания.

Автоматизация деятельности туристического агентства включает в себя следующий набор функций:

- получение и обработка информации от разных туроператоров;
- ведение внутреннего документооборота и бухгалтерии;
- выстраивание взаимоотношений с туроператорами;
- анализ данных и получение статистических отчетов.

В зависимости от используемых средств для реализации этих функций можно рассматривать следующие уровни автоматизации туристического предприятия:

1 Применение стандартного программного обеспечения может быть достаточным для небольших объемов клиентского обслуживания. Безусловным является и наличие возможности осуществления электронных коммуникаций посредством сервисов сети Интернет.

2 Применение специального туристического программного обеспечения. На данном уровне автоматизации требуется более высокая профессиональная подготовка персонала компании.

3 Использование глобальных компьютерных систем бронирования. В этом случае туроператоры рассматривают свою работу в едином информационном пространстве, что позволяет в значительной степени расширить свое представительство в сети Интернет.

4 Участие в электронном бизнесе. Данный уровень автоматизации предполагает расширение собственного присутствия турфирмы в сети Интернет. Это может быть представлено следующими средствами:

1 Сайт-визитка со списком услуг, реквизитами, прайс-листом, фотографиями и другой информацией, дублирующей рекламные проспекты фирмы.

2 Интерактивный сайт для быстрой связи клиентов с фирмой через интернет, способный выдавать информацию по запросу пользователя, отвечать на его вопросы, иметь средства обратной связи с фирмой. Такой сайт способствует увеличению числа потенциальных клиентов.

3 Интернет-магазин, способный принимать платежи за туристические

услуги, бронировать путевки, билеты, заключать договора с клиентами, оповещать их о свободных местах, путевках и др. Такой сайт выполняет функцию полноценного параллельного механизма реализации туристических услуг, позволяет разгрузить менеджеров, ускорить выполнение рутинных операций (прием платежей, подготовка и подпись бумаг, приезд клиента в офис и др.).

4 Подключение туристического агентства на основе абонентской платы к посредническим бизнес-системам, реализующим вышеупомянутые возможности и берущим на себя ответственность за безопасность ведения бизнеса.

Обобщая возможности автоматизации предприятия турбизнеса, можно назвать основные решаемые при этом задачи:

1 Мониторинг состояния рынка – для поиска и бронирования туров, в том числе в режиме онлайн.

2 Автоматизация внутреннего документооборота – выписка необходимых туристу документов (путевка, приходный/расходный кассовый ордер, договор, ваучер и др.), отслеживание жизненного цикла заявки клиента.

3 Автоматизация взаимоотношений с туроператорами – создание и печать бланка заявки, автоматическое отслеживание прохождения заявки от момента ее формирования до момента отправки в архив.

4 Автоматизация бухгалтерии – использование специализированных бухгалтерских программ. В ряде случаев турагентства, работающие по упрощенной системе налогообложения, пользуются услугами аудиторских компаний, сдавая им лишь первичную документацию. В этом случае функций внутриофисных программ, касающихся учета финансов, оказывается достаточно. Возможно сопряжение специализированных внутриофисных туристических программ с бухгалтерскими на уровне обмена файлами.

5 Автоматизация анализа данных и получение статистики – формирование статистических отчетов, показывающих рентабельность работы компании за промежуток времени, среднюю доходность заявок по направлениям и туроператорам и т. д. [7]. Это позволяет агентству правильно ориентироваться на рынке и разрабатывать нужные направления деятельности, выстраивая взаимоотношения с туроператорами, в нужное время давать нужную рекламу и оценивать, как она работает.

Чтобы воспользоваться преимуществами цифровых технологий в полной мере, туристские компании должны интегрировать их в повседневные процессы [8].

1.2 Аналоги информационной системы туристического агентства

Для проведения сравнительного анализа веб-приложения, необходимо понять какими чаще всего пользуются пользователи, принадлежащие к целевой аудитории.

Проведен обзор наиболее популярных веб-приложения Республики Беларусь и Туркменистана. Перечень анализируемых веб-приложений, ранжированных по популярности в Беларуси составлен по результатам опроса [9] и представлен на рисунке 1.2

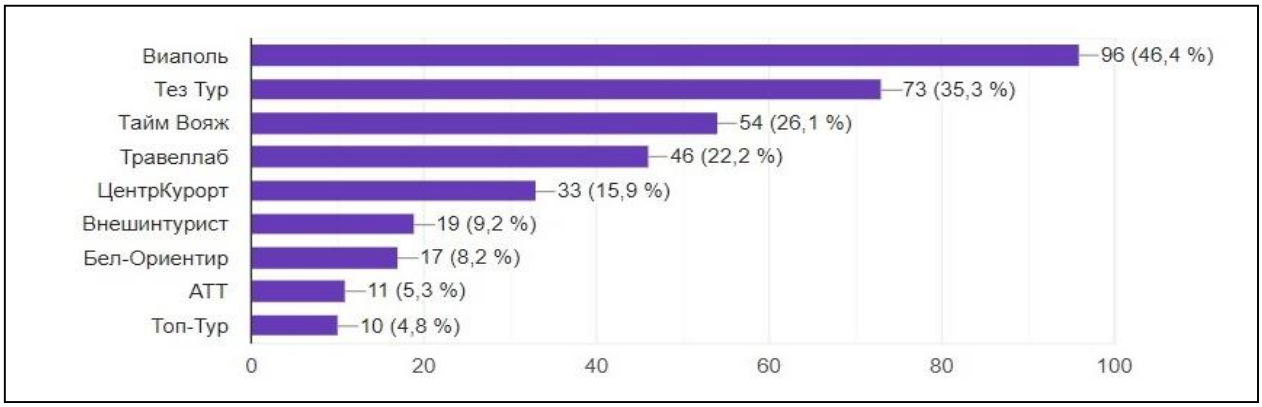

Рисунок 1.2 – Рейтинг веб-ресурсов внутреннего туризма Беларуси

Первую строчку в предпочтениях пользователей белорусских сайтов туристических агентств занимает «Виаполь» [10]. В тройке лидеров также – «Тез Тур» и «Тайм Вояж». Далее по популярности идут такие компании как «Травеллаб», «Внешинтурист», «АТТ», «Бел-Ориентир», «ЦентрКурорт» и «Топ-Тур».

Веб-сайты одних из самых популярных в Республике Беларусь туроператоров – «Тез Тур» и «Тайм Вояж», а также наиболее крупной туристической компании Туркменистана «Азади» рассмотрены в качестве аналогов разработки.

«TezTour» имеет заслуженную репутацию одной из самых высокотехнологичных и надежных компаний на рынке туризма, которая напрямую сотрудничает с такими известными сетями отелей как Hilton, Iberostar, Four Seasons, Radisson, Aldemar, предлагает пакетные туры на основе чартеров, лоукостов и регулярных рейсов. При бронировании туров, туристы получают необходимую визовую поддержку, услуги страхования, трансферы, различные дополнительные услуги и круглосуточную помощь от специалистов в решении любых вопросов касательно путешествия [11].

Бронируя туры на сайте, клиент не только приобретает качественный туристический продукт, но и экономит время. Подбор и бронирование туров онлайн представлен на рисунке 1.3

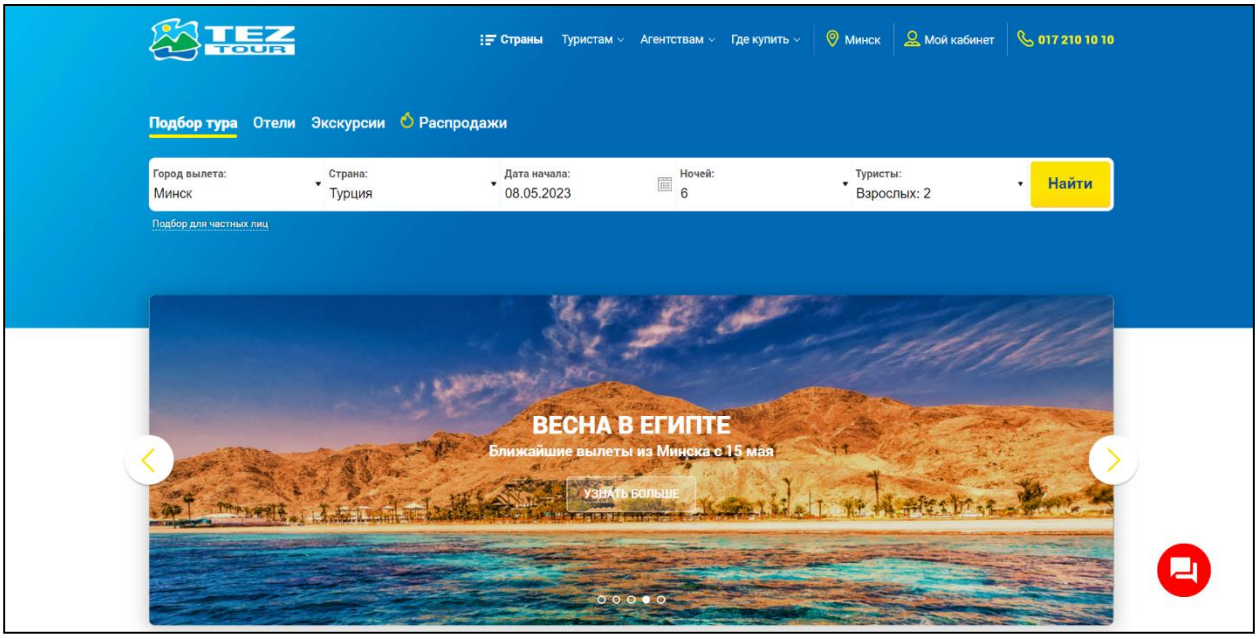

Рисунок 1.3 – Подбор и бронирование туров онлайн веб-приложения «TezTour»

Туристическая компания «Тайм Вояж» [11] существует на рынке туристических услуг Беларуси с 1992 года. За это время компания стала представителем популярной сети услуг туров горящих путевок, членом республиканского союза туристических организаций и уполномоченным агентом крупнейшего туроператора «TezTour».

В веб-приложения представлен каталог служб с возможностью сортировки по рейтинг, типу туров. В краткое описание службы входят название, типы туров. Представлена информация о рейтинговой оценке службы и возможность ознакомиться с отзывами других клиентов. Для поиска определенной службы или тура предусмотрена специальная форма «Поиск». Снимок главной страницы экрана представлена на рисунке 1.4 [12].

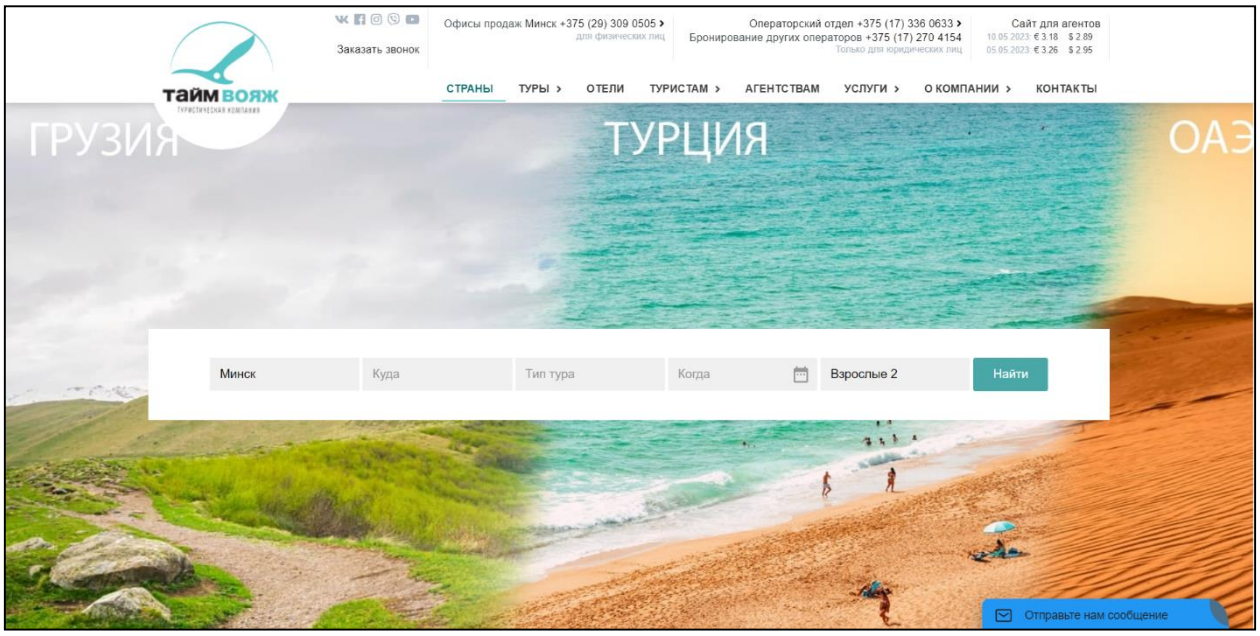

Рисунок 1.4 – Главная страница веб-приложения «Тайм Вояж»

Основными функциями приложения являются: поиск по названию страны, курорты, города; поиск туров по названию; бронирование тура.

Компания «Азади» работает на рынке Туркменистана с 1992 года и осуществляет риэлторскую, ремонтно-строительную деятельность, предоставляет туристические услуги. На сайте компании можно ознакомиться с перечнем услуг и подробным их описанием, получить информацию о турах. Снимок главной страницы экрана представлена на рисунке 1.5. [13].

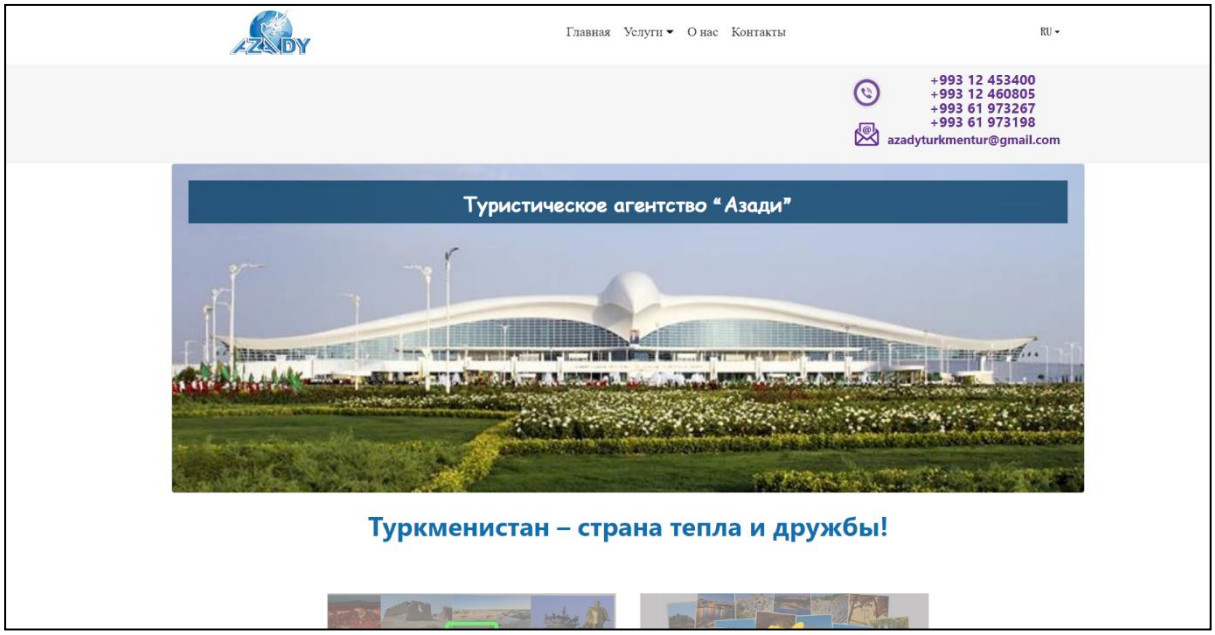

Рисунок 1.5 – Главная страница веб-приложения «Азади»

Основной функцией данного веб-приложения является предоставление информации пользователю и реклама услуг компании.

Важными критериями при оценке информационных систем для поддержания деятельности туристического агентства являются информативность (объем представленных сведений, детальность данных, использование фото- и видеоряда); обновление информции; навигация (структурированность информации, удобство пользования веб-сайтом); технические возможности (наличие информационных справочников, возможности бронирования туров, авиабилетов, гостиниц. Сравнительный анализ аналогов разработки представлен в таблице 1.1.

Таблица 1.1 – Сравнительный анализ информационных систем

Критерии оценки	Информационные системы турагентств		
	TezTour	Тайм Вояж	Азади
Кроссплатформенность	+	+	+
Функциональность: Поиск туров	+	+	–
Выбор услуг	+	+	–
Заказ туров / билетов	+	+	–
Бронирование туров/ билетов	+	+	–
Покупка туров/ билетов	+	+	–

1.3 Выводы и постановка задач на дипломное проектирование

Анализ предметной области турагентства включает в себя определение концептуальных требований и информационных потребностей пользователей ИС, а также определение основных модулей, входящих в состав ИС, и связей между ними.

Концептуальные требования и информационные потребности пользователей информационной системы могут быть следующими:

– учет клиентов турагентства;

– возможность создания профиля клиента;

– хранение данных о контактах клиента;

– возможность просмотра истории бронирований и путешествий клиента;

– учет туров и предложений;

– возможность добавления, редактирования и удаления туров;

– хранение данных о дате, месте, длительности, стоимости и других характеристиках туров;

– возможность просмотра доступных туров и предложений, а также их фильтрация по различным параметрам;

– возможность бронирование туров и управление заказами;

– хранение информации о заказе и его истории изменений;

– возможность оплаты заказа.

Таким образом, информационная система для турагентства должна обеспечивать возможность учета клиентов и доступных туров, создания и управления заказами, учета финансовых операций. Модули ИС должны взаимодействовать между собой, передавая необходимую информацию и обеспечивая циклический процесс работы турагентства. ИС для турагентства должна обеспечивать комплексную автоматизацию бизнес-процессов, связанных с работой турагентства, от учета клиентов и доступных туров до управления заказами, финансовым учетом и эффективности работы. Важно учитывать специфику работы турагентства и потребности его пользователей при разработке ИС, чтобы обеспечить ее эффективность и удобство использования.

Для успешной реализации системы необходимо, в первую очередь, выделить основные задачи, которые будет решать система «Информационная система поддержки деятельности туристического агентства» а также те задачи, которые необходимо выполнить для правильной работы системы.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– выполнить аналитический обзор предметной области и аналогов разработки;

– разработать информационную систему туристического агентства;

– выполнить расчет надежности информационной системы и производственных рисков;

– осуществить функциональное тестирование веб-приложения;

– выполнить технико-экономическое обоснование эффективности разработки;

– разработать мероприятия по повышению работоспособности и производительности труда разработчика.

2 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ТУРИСТИЧЕСКОГО АГЕНТСТВА

2.1 Структура информационной системы

Разработка программного средства информационной системы для туристического агентства имеет цель создания продукта, который облегчит процесс управления туристической деятельностью, автоматизирует рутинные процессы и повысит эффективность работы агентства.

Структура и функционирование информационной систем: информационной систем должна содержать подсистемы для бронирования туров, управления клиентской информацией и.т.д. Режимы функционирования информационной систем работа в онлайн и офлайн режимах.

Надежность информационной систем: ИС должна быть высоконадежной и обеспечивать безопасность информации о клиентах и турах.

Плюсы разрабатываемой информационной систем:

− информационной систем должна обеспечивать быстрый доступ к информации и удобный интерфейс для пользователей;

− повышение эффективности работы сотрудников турагентства и увеличение количества продаж туров.

При описании функциональности системы первым шагом является моделирование требований к ней.

Древовидная структура сайта предполагает иерархическое расположение категорий и подкатегорий, где каждая последующая ветвь вложена в предыдущую. Такая структура используется для удобной категоризации товаров или услуг по брендам, категориям и типам продукции. Большинство современных сайтов используют именно эту структуру, где пользователи могут легко найти нужный раздел, подраздел или конкретный товар, или услугу.

На рисунке 2.1 представлена схема древовидной структуры сайта.

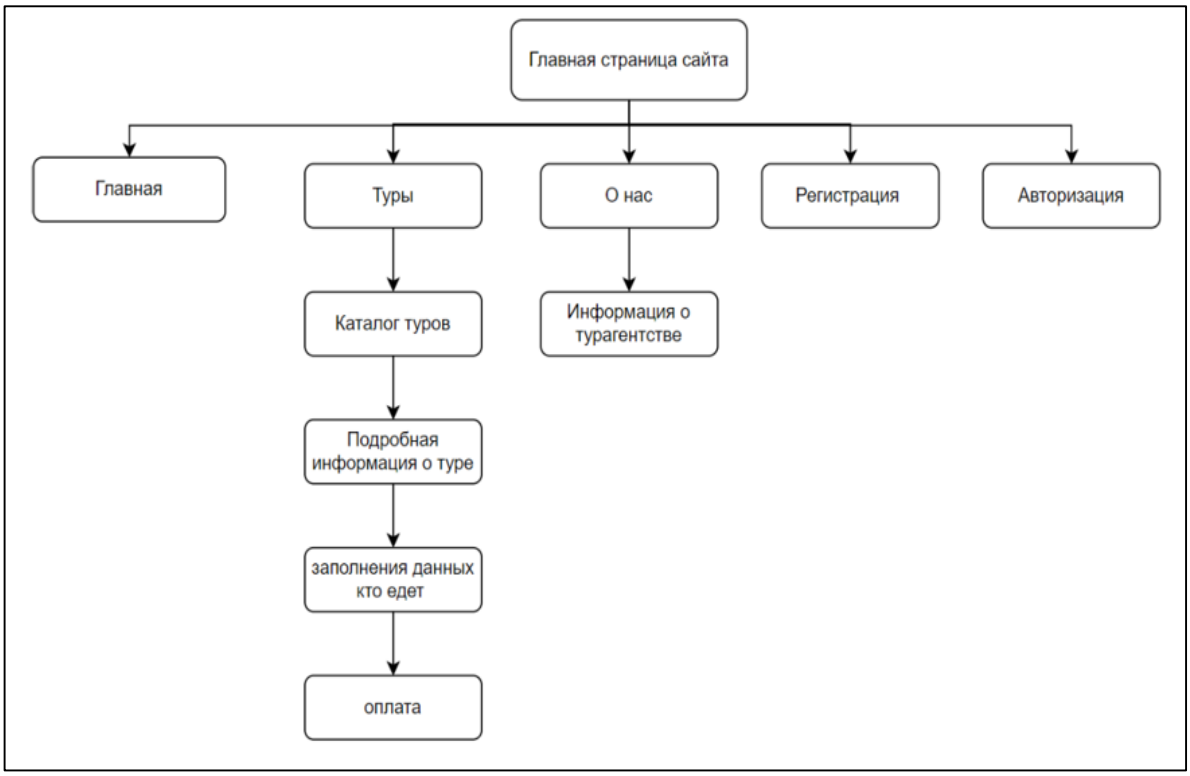

Рисунок 2.1 – Схема древовидной структуры сайта

На диаграммах вариантов использования изображаются акторы и варианты использования, между которыми существуют отношения. Актором будем называть внешнюю по отношению к ИС сущность, которая может взаимодействовать с системой. Акторами могут быть как люди, так и внешние системы или устройства. Следует всегда помнить, что актор – это не конкретный человек или устройство, а роль (должностная обязанность), в которой он выступает по отношению к программной системе.

Смоделировав диаграмму вариантов использования, мы получили наглядное представление о нашей системе. Был определен модель клиента.

На рисунке 2.2 представлена диаграммах вариантов использования.

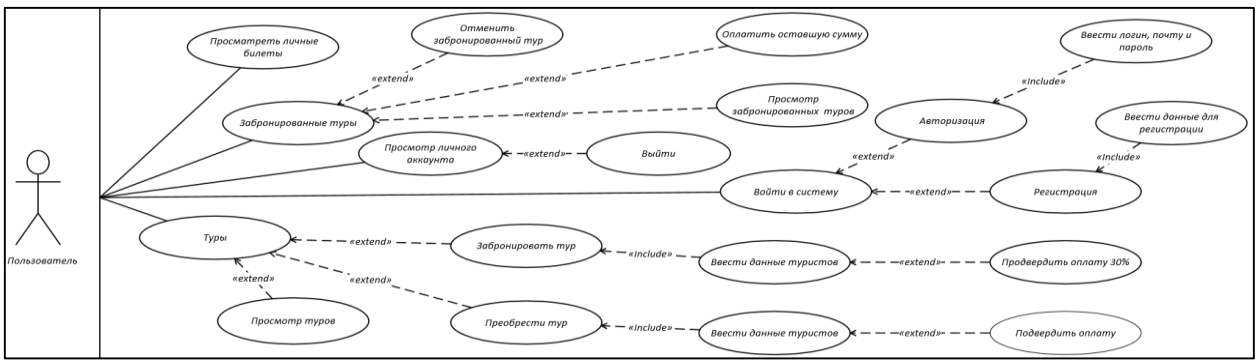

Рисунок 2.2 – Диаграммах вариантов использования

Диаграммы последовательности визуально моделируют поток логики в системе, позволяя документировать и проверять свою логику, и обычно используются как для анализа, так и для целей проектирования. Диаграммы последовательности являются наиболее популярным артефактом UML для динамического моделирования, которое фокусируется на определении поведения в данной системе. Другие методы динамического моделирования включают в себя диаграмму активности, диаграмму связи, временную диаграмму и диаграмму обзора взаимодействия. Диаграммы последовательности, а также физические модели данных, на мой взгляд, являются наиболее важными моделями уровня проектирования для разработки современных Турагенства. Диаграмма вариантов последовательности данного проекта представлена на рисунке 2.3.

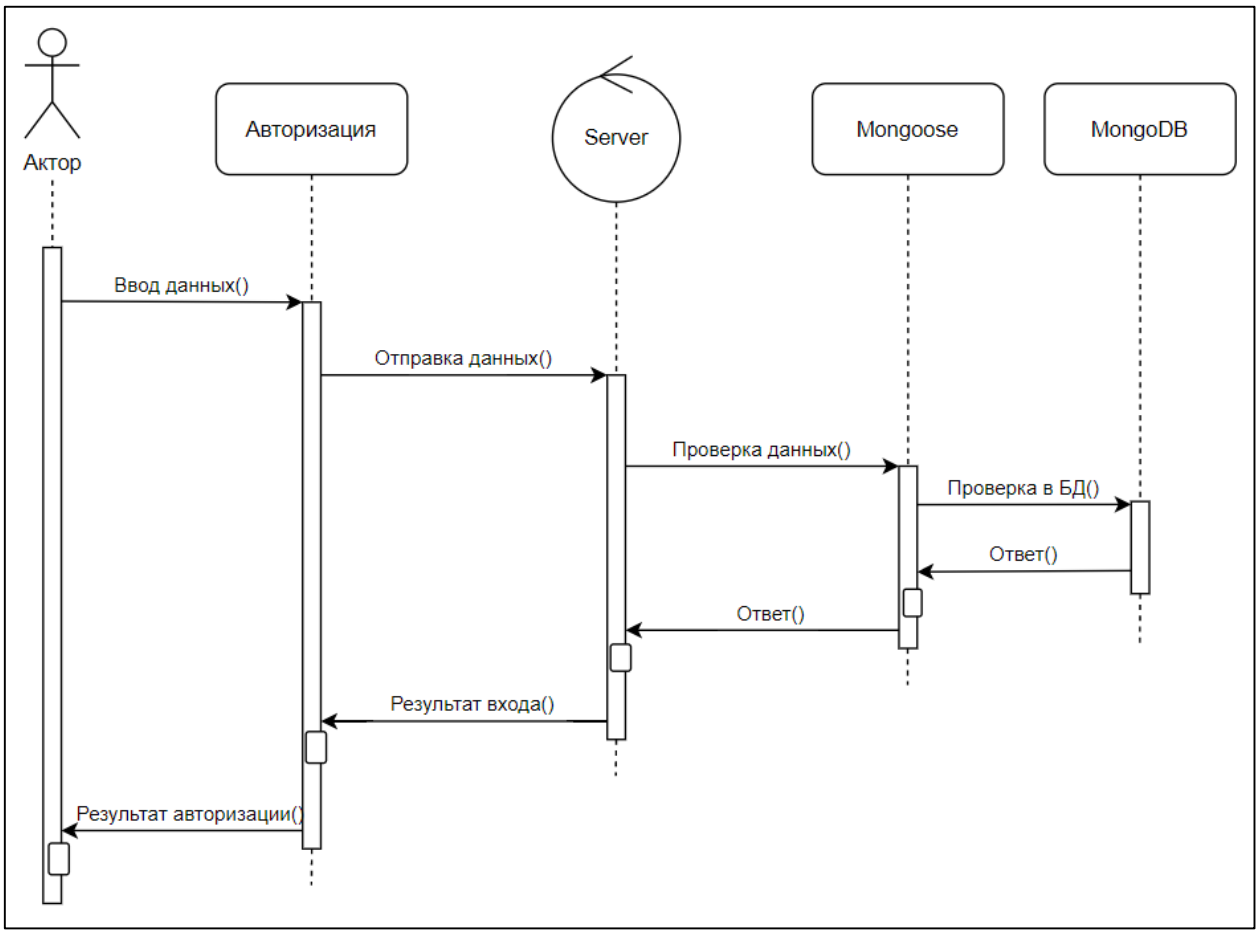

Рисунок 2.3 – Диаграмма вариантов последовательности

Диаграмма состояния Объекты меняют свое состояние в ответ на происходящие события и с течением времени. Диаграмма состояний

представляет состояния объекта и переходы между ними, а также начальное и конечное состояние объекта.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний имеет схожую семантику с диаграммой деятельности, только деятельность здесь заменена состоянием, переходы символизируют действия. Таким образом, если для диаграммы деятельности отличие между понятиями «Деятельность» и «Действие» заключается в возможности дальнейшей декомпозиции, то на диаграмме состояний деятельность символизирует состояние, в котором объект находится продолжительное количество времени, в то время как действие моментально.

Цель данной диаграммы является показать поведение одного объекта в течение его жизни, начиная от создания объекта и заканчивая его уничтожением.

В отличие от других диаграмм, диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. Диаграмма состояния данного проекта представлена на рисунке 2.4.

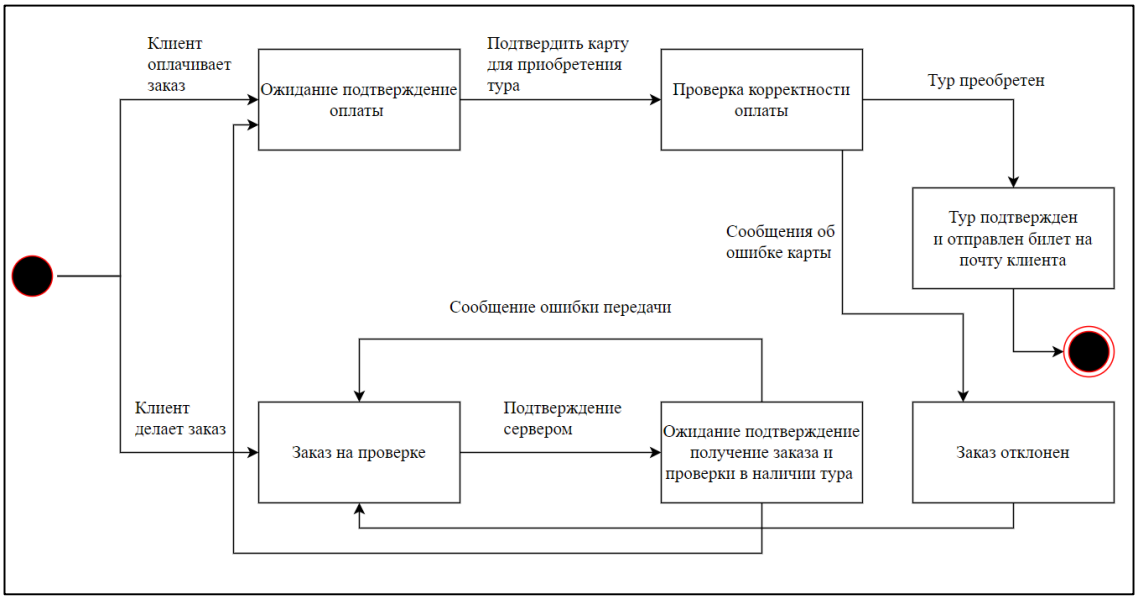

Рисунок 2.4 – Диаграмма состояния

Паттерн проектирования – повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста, иными словами, паттерны

проектирования – это шаблоны необходимых шагов для решения каких-либо определенных задач.

Паттерн *MVC* (*Model – Vie – wController*) состоит из объектов трех видов. Модель – это объект приложения, а вид – экранное представление. Контроллер описывает, как интерфейс реагирует на управляющие воздействия пользователя.

Первым с клиентским запросом взаимодействует контроллер, контроллер разбивает данный запрос на элементы, далее он инициализирует объекты модели. После обработки данные отправляются на уровень представления.

Паттерн *MVC* – простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса.

Диаграмма взаимодействия компонентов для паттерна «*MVC*» изображена на рисунке 2.5.

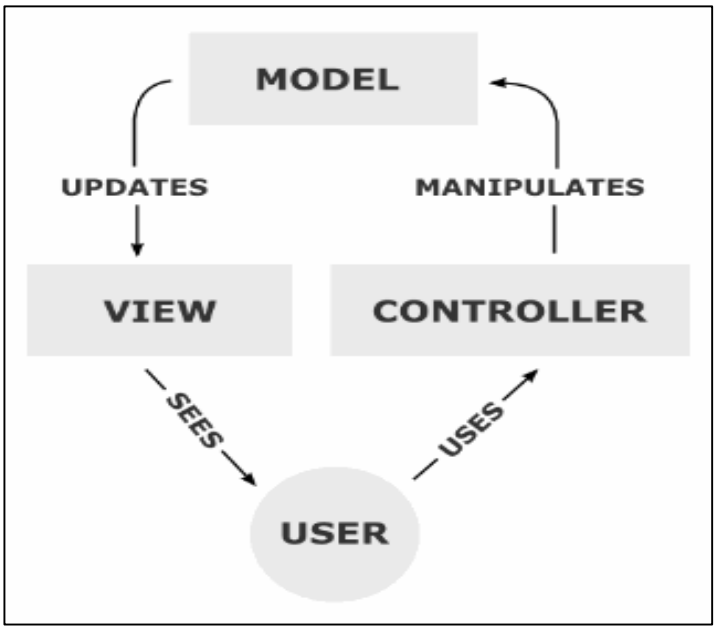

Рисунок 2.5 – Диаграмма для паттерна «*MVC*»

2.2 Алгоритм работы информационной системы

Алгоритм – представляет собой пошаговую процедуру, которая определяет набор инструкций, которые должны быть выполнены в определенном порядке, чтобы получить желаемый результат. Алгоритмы, как правило, создаются независимо от базовых языков, то есть алгоритм может быть реализован на нескольких языках программирования.

Алгоритмы широко используются в различных областях, включая информационные системы, науку о данных, искусственный интеллект, математику, криптографию и многие другие. Алгоритмы позволяют автоматизировать процессы, упрощать решение задач, повышать эффективность и точность вычислений.

Однако создание хорошего алгоритма требует определенных навыков и знаний. Хороший алгоритм должен быть понятным, эффективным, легко поддерживаемым и устойчивым к ошибкам. Кроме того, в процессе разработки алгоритма необходимо учитывать особенности конкретной задачи, области применения и доступных ресурсов.

Основные модули, входящие в состав ИС, могут быть следующими:

1 Модуль учета клиентов и предложений содержит информацию о клиентах турагентства и доступных предложениях. Взаимодействует с модулем бронирования туров и управления заказами.

2 Модуль бронирования туров и управления заказами содержит информацию о созданных заказах и их статусах.

Администратор/пользователь взаимодействуют с модулем учета клиентов и предложений, и модулем учета финансовых операций.

Связи между модулями можно представить в виде цикла, где модуль учета клиентов и предложений связан с модулем бронирования туров и управления заказами, который в свою очередь взаимодействует с модулем учета финансовых операций, который в свою очередь связан с модулем учета клиентов и предложений. Таким образом, при создании заказа на тур, информация о клиенте и выбранном туре передается из модуля учета клиентов и предложений в модуль бронирования туров и управления заказами. После создания заказа, информация о нем передается в модуль учета финансовых операций, где происходит учет финансовых операций, связанных с заказом.

Алгоритм работы функции неавторизованного пользователя, При запуске открывается главная веб-страница, в котором пользователь может просмотреть туры и приобрести их не авторизуясь в системе, алгоритм покупки тура, клиент выбирает тур открывается страница тура, где пользователю нужно вести паспортные данные для биллета и ввести почту для оплаты и для отправки электронного билета, после того как пользователь ведет данные, то на странице покупки открывается меню для оплаты по карте, для подтверждения карты система сама отправит на почту четырехзначный строку для подтверждение а пользователь должен ввести этот код подтверждения в форме для карты и отправить, если код подтверждения и почта совпадает, то система

моментально генерируют онлайн билет и отправляет на почту. Алгоритм работы неавторизованного клиента представлена на рисунке 2.6.

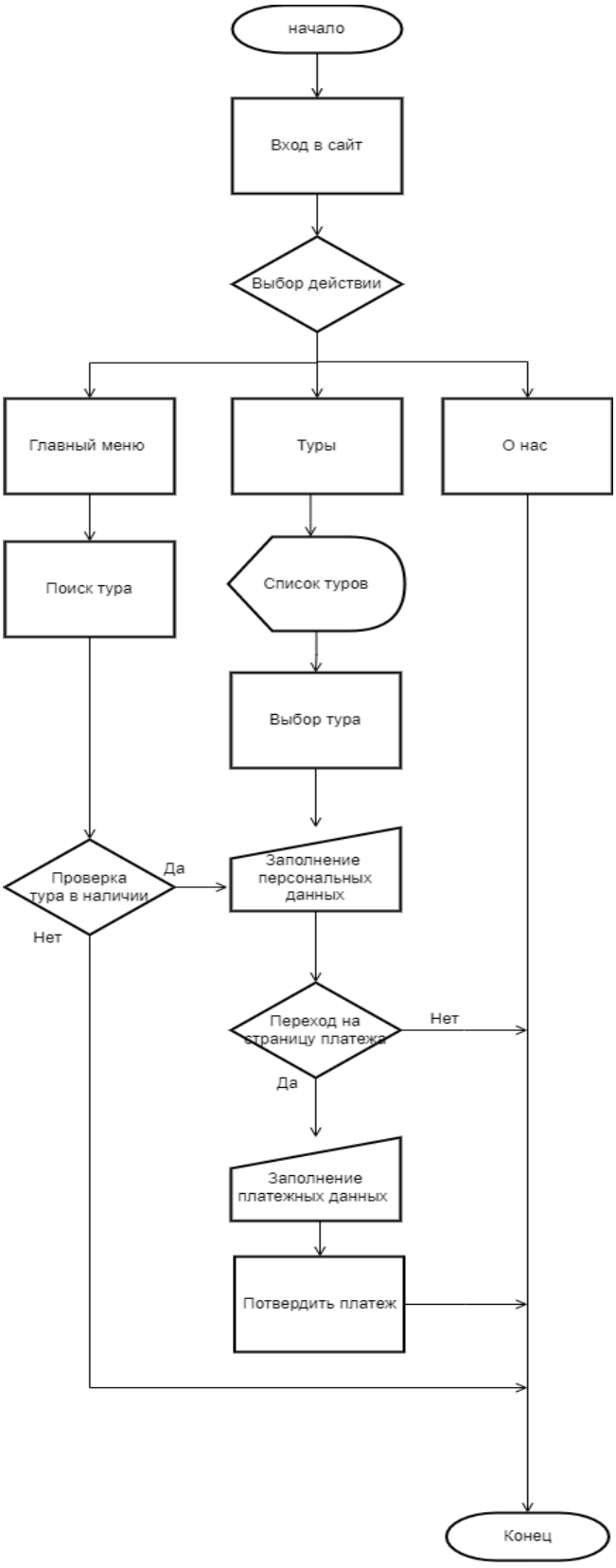

Рисунок 2.6 – Схема алгоритма работы неавторизованного клиента

2.3 Структура базы данных информационной системы

База данных является основой любой информационной системы. Она представляет собой структурированную коллекцию данных, которые используются для хранения, организации и управления информацией. Базы данных могут быть различными по своей структуре, в зависимости от требований конкретной системы.

Структура базы данных определяет, каким образом данные будут храниться, как они будут организованы и как они будут связаны друг с другом. В структуре базы данных могут быть определены таблицы, поля, индексы, отношения и другие элементы, которые необходимы для эффективной работы системы.

Процесс разработки (проектирования) базы данных включает два этапа: разработку логической организации базы данных и создание ее на носителе. Логическая организация базы данных - это предоставление пользователя о предметной области, информация о которой должна храниться в базе данных.

Под физической организацией базы данных понимается совокупность средств и методов размещения данных во внешней памяти и на их основе внутренняя модель данных. Внутренняя модель является средством отображения логической модели данных, показывает, каким образом записи размещаются в базе данных, как они упорядочиваются, как организуются связи, каким путем можно осуществить выборку и так далее.

Проектирование базы данных системы управления Туристической агентством начиналось с создания всех нужных таблиц в базе, всех полей, входящих в каждую таблицу, взаимодействия таблиц между собой с помощью специальных отношений и создание в соответствии с этими параметрами первичных и вторичных ключей.

В данной информационной системе используется база данных *mongoDB*.

MongoDB – это документ-ориентированная база данных, которая использует формат *BSON* (*Binary JSON*) для хранения и организации данных. Она является распространенной *NoSQL* базой данных и используется для множества приложений и сервисов, таких как системы управления контентом, аналитические приложения, онлайн-магазины и многое другое.

MongoDB позволяет хранить данные в формате *JSON*-подобных документов, которые могут содержать любое количество полей и поддокументов. Это дает возможность хранить сложные иерархические данные, которые не могут быть представлены в привычных таблицах и строках. *MongoDB*[16]

также поддерживает масштабирование горизонтально (*scale-out*), что позволяет обрабатывать большие объемы данных и высокую нагрузку. Она также имеет множество функций, таких как автоматическое шардирование данных, репликацию и индексацию, что позволяет создавать высокопроизводительные и отказоустойчивые приложения.

Структура базы данных информационной системы туристического агентства представлен на рисунке 2.7.

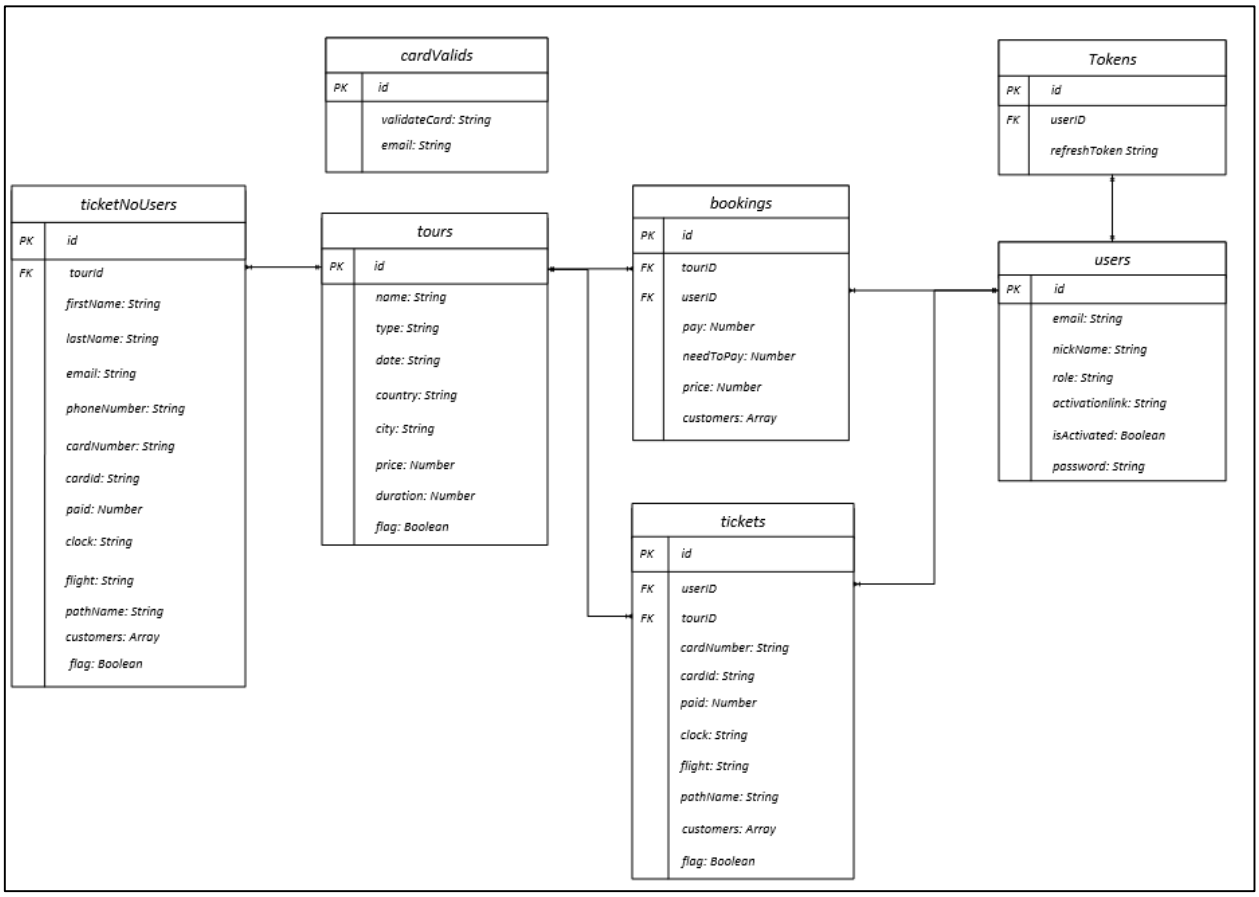

Рисунок 2.7. – Структурная схема базы данных информационной системы туристического агентства

Коллекция «*tokens*» содержит информацию о токенах пользователей, которые используется приложением в процессе сеанса пользователей в системе. Структура представлена в таблице 2.1.

Таблица 2.1 – Структура коллекций «*tokens*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор токена
userID	*String*	Идентификатор пользователя
refreshToken	*String*	Token пользователя

Коллекция «*users*» содержит информацию о пользователях, которая используется приложением в процессе авторизации пользователей в системе. Если пользователь вводит верные данные, то система дает ему доступ к системе. Структура представлена в таблице 2.2.

Таблица 2.2 – Структура коллекций «*users*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор пользователя
email	*String*	Эл. почта пользователя
nickName	*String*	Имя пользователя
role	*String*	Роль пользователя
activationLink	*String*	Активационная ссылка
isActivated	*Boolean*	Активирован ли пользователь
password	*String*	Пароль пользователя

Коллекция «*tours*» предназначена для добавления туров. В коллекции будет содержаться информация о названии тура, стране, город, длительность, цена, тип тура, а также о дате тура. Структура приведена в таблице 2.3.

Таблица 2.3 – Структура коллекций «*tours*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор тура
name	*String*	Название тура
type	*String*	Тип тура
date	*String*	Дата тура
country	*String*	Страна тура
city	*String*	Город тура
price	*Number*	Цена за тур
duration	*Number*	Длительность тура
flag	*Boolean*	Состояния тура

Коллекция «*bookings*» предназначена для бронирования тура, в коллекции будет содержаться информация о стоимости тура сколько оплачено взнос, сколько нужно будет оплатить и кто едет. коллекция приведена в таблице 2.4.

Таблица 2.4 – Структура коллекций «*bookings*»

Имя поля	Тип данных	Описание
1	2	3
id	*String*	Идентификатор броня

Продолжение таблицы 2.4

1	2	3
tourId	*String*	Идентификатор тура
userId	*String*	Идентификатор пользователя
pay	*Number*	Оплачено за тур
needToPay	*Number*	Нужно оплатить
price	*Number*	Общая стоимость тура
customers	*Array*	Кто едет

Коллекция «*tickets*» предназначена для генерирования билета, в коллекции будет содержаться информация о рейсе, кто едет, с какой карты был оплачено тур, и время. коллекция приведена в таблице 2.5.

Таблица 2.5 – Структура коллекций «*tickets*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор билета
tourId	*String*	Идентификатор тура
userId	*String*	Идентификатор пользователя
cardNumber	*String*	С какой карты был оплачен тур
cardId	*Number*	Идентификатор подтверждения карты
clock	*String*	Время полета
flight	*String*	Рейс тура
pathName	*String*	Название билета
customers	*Array*	Кто едет
flag	*Boolean*	Состояние билет

Коллекция «*cardValids*» предназначена для подтверждения оплаты картой, в коллекции будет содержаться информация о почте покупателя и идентификатор подтверждавший личность данного покупателя. коллекция приведена в таблице 2.6.

Таблица 2.6 – Структура коллекций «*cardValids*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор *cardValids*
validateCard	*String*	Идентификатор подтверждения карты
email	*String*	Почта пользователя

Коллекция «*ticketNoUser*» предназначена для не зарегистрированных пользователей при приобретении тура, в коллекции будет содержаться информация о фамилии, имени, почте, номер телефона, с какой карты был оплачен, покупателя и идентификатор подтверждавший личность данного покупателя, также рейс, кто едет состояние билет. коллекция приведена в таблице 2.7.

Таблица 2.7 – Структура коллекций «*ticketNoUser*»

Имя поля	Тип данных	Описание
id	*String*	Идентификатор билета
firstName	*String*	Фамилия пользователя
lastName	*String*	Имя пользователя
email	*String*	Почта пользователя
phoneNumber	*String*	Номер телефона пользователя
cardNumber	*String*	Номер карты пользователя
cardId	*String*	Номер подтверждения пользователя
paid	*String*	Сколько оплаченного тура
clock	*String*	Время полета
flight	*String*	Рейс
pathName	*String*	Название билета
customers	*Array*	Кто едет
flag	*Boolean*	Состояние билет

2.4 Выводы и оценка результатов разработки

В процессе проектирования информационной системы была предложена структура сайта, алгоритм работы и структура базы данных.

Представлена древовидная структура информационной системы. Были реализованы диаграммы вариантов использования, последовательности, состояние и диаграмма паттерного проектирвоание *MVC*.

Представлена пошаговая процедура, которая определяет набор инструкций. Реализован алгоритм работы функции неавторизованного пользователя. Модель базы данных нереляционная, имеет все коллекции данных, связанные между собой с помощью соответствующих структур, что обеспечивает логичную и удобную работу.

Для реализации пользовательского интерфейса была выбрана библиотека *ReactJs*, *Bootstrap* и *HTML*, которая является одной из самых популярных библиотек для создания пользовательских интерфейсов в веб-приложениях. Эта библиотека разработана компанией *Facebook* и предназначена для создания масштабируемых и удобных интерфейсов.

HTML (*HyperText Markup Language*) – это язык разметки, который используется для создания структуры и отображения содержимого веб-страниц. *HTML* является основным языком для создания веб-страниц и состоит из серии тегов, которые определяют различные элементы и их свойства на странице.

Bootstrap – это популярный фреймворк для разработки пользовательских интерфейсов (*UI*) веб-приложений. Он предоставляет набор готовых компонентов, стилей и *JavaScript*-плагинов, которые значительно упрощают создание современных и отзывчивых веб-интерфейсов [19].

Для разработки серверной части был выбран *Node.js* и *Express.js*

Node.js – это среда выполнения *JavaScript* на стороне сервера. Она основана на движке *V8* от *Google*, который используется в браузере *Google Chrome* для выполнения JavaScript-кода. *Node.js* позволяет использовать *JavaScript* для разработки серверных приложений и инструментов командной строки [21].

Для хранения данных была выбрана база данных *MongoDB*. *MongoDB* является современной и гибкой базой данных, относящейся к категории *NoSQL* баз данных. Она отличается от традиционных реляционных баз данных тем, что использует документоориентированную модель данных [16].

Выше описанные структуры значительно облегчают процесс создания информационной системы и выявления недочетов и ошибок на стадии реализации информационной системы.

3 АНАЛИЗ РАСЧЕТА НАДЕЖНОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТУРИСТИЧЕСКОГО АГЕНТСТВА

3.1 Расчет надежности информационной системы по модели сложности

В ходе расчета модели сложности рассчитываются следующие метрики:

− метрики размера: объем программы *V*; потенциальный объем программы V^*;

− метрики сложности потока управления (по Джиблу): абсолютная сложность программы *CL*; относительная сложность программы *cl*; максимальный уровень вложенности оператора *CLI* [21];

− метрики сложности потока данных: метрика Чепина *Q*;

− метрики стилистики и понятности программ (по Холстеду): теоретическая длина программы $N^{\wedge}$; метрика корректности программы *L*; метрика корректности реальной программы $L^{\wedge}$; число элементарных решений, принятых при написании программы *E*;

− объектно-ориентированные метрики: суммарная сложность всех методов класса *WMC*; глубина дерева наследований *DIT*; количество потомков *NOC*; сцепление между классами *CBO*; мощность множества классов *RFC*; недостаток сцепления методов *LCOM*.

Метрики размера программного средства (ПС). В данные метрики включают следующее:

– объем программы *V*;

– потенциальный объем программы *V**.

Объем V определяется по формуле:

$$V = (N_1+N_2) \cdot log_2(n_1+n_2), \tag{3.1}$$

где $(n_1 + n_2)$ – словарь ПС; n_1 – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов); n_2 – число уникальных операндов программы (словарь операндов); $(N_1 + N_2)$ – длина ПС *N*; N_1 – общее число операторов в программе; N_2 – общее число операндов в программе.

Подставив значения в формулу (3.1) получим выражение

$$V = (5463+7812) \cdot log_2(4658 + 6647) = 13275 \cdot 13{,}465 = 178747{,}875$$

Потенциальный объем V^* рассчитывается по формуле:

$$V^* = n^* \cdot log_2(n^*), \quad (3.2)$$

где n^* – теоретический словарь ПС.

Подставив значения в формулу (3.2) получим выражение

$$V^* = 4132 \cdot log_2(4132) = 49637{,}716$$

Метрики сложности потока управления. Используются следующие метрики, предложенные Джиблом:

– абсолютная сложность управления *CL*, характеризующаяся количеством операторов условия;

– относительная сложность *cl*;

– насыщенность ПС операторами условия определяется отношением *CL* к общему числу операторов;

– максимальный уровень вложенности оператора условия *CLI* [22].

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Для этого все переменные, составляющие список ввода-вывода, разбивается на четыре функциональные группы:

1 *P* – вводимые переменные для расчетов и обеспечения вывода.

2 *M* – модифицируемые, или создаваемые внутри программы переменные.

3 *C* – переменные, участвующие в управлении работой программного модуля (управляющие переменные).

4 *T* – не используемые в программе («паразитные») переменные.

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Сложность потока данных *Q* определяется выражением:

$$Q = P + 2M + 3C + 0{,}5T \quad (3.3)$$

Подставив значения в формулу (3.3) получим следующее значение:

$$Q = 21 + 2 \cdot 415 + 3 \cdot 73 + 0{,}4 \cdot 0 = 1070$$

Метрики стилистики и понятности ПС. В данной группе используются три метрики:

– отклонение реальной N от теоретической $N^{\wedge}$ длины ПС ΔN (в %);

– уровень качества программирования L (уровень ПС);

– оценка интеллектуальных усилий на разработку ПС E.

Отклонение N от $N^{\wedge}$ определяется выражением:

$$\Delta N = \frac{N_1 + N_2}{N^{\wedge}} \cdot 100\%, \tag{3.4}$$

где $N^{\wedge}$ определяется выражением:

$$N^{\wedge} = n_1 \cdot log_2(n_1) + n_2 \cdot log_2(n_2), \tag{3.5}$$

где n_1 – словарь операторов; n_2–словарь операндов ПС.

Таким образом, теоретическая длина $N^{\wedge}$ информационной системы передачи информации составит:

$$N^{\wedge} = 4657 \cdot log_2(4657) + 6647 \cdot log_2(6647) = 141161.336$$

Отклонение N от $N^{\wedge}$ определим по формуле (3.4):

$$\Delta N = \frac{5463 + 7812}{141161.336} \cdot 100\% = 9{,}4\%$$

Уровень качества программирования L рассчитывается по формуле:

$$L = \frac{V^*}{V}, \tag{3.6}$$

Подставив численные значения получим:

$$L = \frac{49637{,}716}{178747.875} = 0{,}277$$

Исходным для введения этой характеристики является предположение о том, что при снижении стилистического качества программирования уменьшается содержательная нагрузка на каждый компонент программы и, как следствие, расширяется объем реализации исходного алгоритма.

Интеллектуальные усилия на разработку ПС E рассчитывается по формуле:

$$E = V \cdot \frac{V}{V^*}, \tag{3.7}$$

где V – объем программы; V^* – потенциальный объем программы.

Подставив численные значения в формулу (3.7) получим:

$$E = 178747{,}875 \cdot \frac{178747{,}875}{49637{,}716} = 643679{,}95$$

Объектно-ориентированные метрики. Рассчитаем метрики Мартина:

– центростремительное сцепление C_a – количество классов вне этой категории, которые зависят от классов внутри этой категории;

– центробежное сцепление C_e – количество классов внутри этой категории, которые зависят от классов вне этой категории;

– нестабильность I.

Нестабильность определяется по формуле:

$$I = \frac{C_e}{(C_e + C_a)}, \tag{3.8}$$

Подставив численные значения получим:

$$I = \frac{25}{25 + 23} = 0{,}52$$

Далее рассчитаем нормативный x_{min} и фактический $x_{\phi i}$ уровни для каждой из метрик по формулам:

$$x_{min} = \frac{a_{min}}{a_{max}}, \tag{3.9}$$

$$x_{\phi i} = \frac{a_i}{a_{\max}} \tag{3.10}$$

где a_i – значение метрики, рассчитанное для конкретного ПС; a_{min} – минимально возможное значение этой метрики для данного типа ПС; a_{max} – максимально возможное значение этой метрики для данного ПС.

Также рассчитываем дискриминант d_i для каждой метрики по формуле:

$$d_i = \frac{x_{\min}(1 - x_{\phi i})}{x_{\phi i}(1 - x_{\min})}, \tag{3.11}$$

Результаты вычислений приведены в таблице 3.1.

Таблица 3.1 – Нормативные пределы метрик, результаты вычислений нормативных и фактических уровней метрик, дискриминантов метрик

Метрики	Значение a_i	a_{min}	a_{max}	x_{min}	$x_{\phi i}$	d_i
V	178747,875	3100	240 000	0,129167	0,744	0,0508
*V**	49637,716	508	69 000	0,007362	0,71	0,0028
Q	1070	57	3425	0,016642	0,312	0,0372
ΔN	9,4	4	35	0,114286	0,2685	0,3514
L	0,277	0,003	3,1	0,000968	0,089	0,00987
E	643679,95	23000	991000	0,023209	0,6495	0,0128
C_a	23	11	141	0,078014	0,1631	0,434
C_e	25	11	210	0,052381	0,1190	0,409
I	0,52	0	2	0	0,26	0

Рассчитываем риск снижения надежности работы программного средства *R* по формуле:

$$R = 1 - \prod_{i=1}^{N}(1 - d_i)^{\lambda_i} \tag{3.12}$$

где λ_i – весовые коэффициенты для конкретных метрик. Отражает на сколько та или иная метрика имеет больший вес для надежности ПС и должна удовлетворять условию:

$$\sum_{i=1}^{N} \lambda_i = 1, \lambda_i \geq 0 \tag{3.13}$$

где *N* – количество метрик, используемое при расчете риска снижения надежности.

Для упрощения задачи примем равнозначным вклад каждой метрики в результат расчета риска снижения надежности.

Таким образом, по формуле (3.12) для данного программного продукта риск снижения надежности равен:

$$R = 0{,}056$$

Следовательно, надежность равна:

$$N = 1 - 0{,}056 = 0{,}9432$$

3.2 Расчет надежности информационной системы по модели Джелинского - Моранды

Модель Джелинского – Моранды строится на основе следующих допущений:

1 Интенсивность обнаружения ошибок $\lambda(t)$ пропорциональна текущему числу ошибок в программном средстве, т. е. числу оставшихся ошибок.

2 Все ошибки одинаково вероятны, и их появления независимы.

3 Каждая ошибка имеет один и тот же порядок серьезности.

4 Время до следующего отказа (ошибки) распределено экспоненциально.

5 Программное средство функционирует в среде, близкой к реальной.

6 Ошибки постоянно корректируются без внесения в программное средство новых.

7 $\lambda(t) = const$ в интервале между двумя соседними ошибками.

В соответствии с этими допущениями интенсивность возникновения ошибок в программном средстве можно представить в виде

$$\lambda(t) = K[B - (i - 1)] \tag{3.14}$$

где t – произвольное время между обнаружением (i-1) и i-й ошибок; K – неизвестный коэффициент; B – неизвестное общее число ошибок в программном средстве.

А вероятность безотказной работы программного средства рассчитывается согласно выражению:

$$P(t) = e^{-\lambda(t)} \tag{3.15}$$

Для нахождения B используют выражения

$$f_n(B+1) = g_n(B+1, A), \tag{3.16}$$

$$f_n(m) = \sum_{i=1}^{n} \frac{1}{m-i}, \tag{3.17}$$

$$g_n(m, A) = \frac{n}{m - A}, \tag{3.18}$$

$$A = \frac{\sum_{i=1}^{n} i \cdot X_i}{\sum_{i=1}^{n} X_i} \tag{3.19}$$

$$m = B + 1, \tag{3.20}$$

где $m \geq n+1$ – число прогнозируемых ошибок; n – количество выявленных ошибок.

Неизвестный коэффициент K рассчитывают по формуле

$$K = \frac{n}{(B+1) \cdot \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} iX_i}. \tag{3.21}$$

На этапе отладки программного средства за 179 день было выявлено 10 ошибок. Исходные данные сведены в таблицу 3.2 в виде интервалов времени X_i (дни) между соседними ошибками (i – номер ошибки). Необходимо найти вероятность безотказной работы программного средства.

Таблица 3.2 – Исходные данные для расчетов

i	1	2	3	4	5	6	7	8	9	10
X_i, день	2	9	11	14	16	18	22	27	33	39

Подставив численные значения в формулу (3.19) получим A=7,1712

Осуществим расчеты по формулам (3.17), (3.18) и сведем их в таблицу 3.3.

Таблица 3.3 – Результаты вычислений

i	X_i	$i \cdot X_i$	m	$g_n(m, A)$	$f_n(m)$	$\lvert f_n(m) - g_n(m, A) \rvert$
1	2	2	11	2,553191	2,928968254	0,375777
2	9	18	12	2,033898	2,019877345	0,01402
3	11	33	13	1,690141	1,603210678	0,08693
4	14	56	14	1,445783	1,346800422	0,09898
5	16	80	15	1,263158	1,168228993	0,09493
6	19	114	16	1,121495	1,03489566	0,0866
7	22	154	17	1,008403	0,930728993	0,07767
8	27	216	18	0,916031	0,84669538	0,06934
9	33	297	19	0,839161	0,777250935	0,06191
10	39	390	20	0,774194	0,718771403	0,05542

Из таблицы 3.3 видно, что наилучшим решением для уравнения (3.16) является m=12, в таком случае согласно выражению (3.20) B = 14.

Подставив в формулу (1.29) рассчитанные значения получим

$$K = \frac{10}{12 \cdot 179\text{-}1360} = 0.012690.$$

Далее согласно с формулой (1.22) рассчитаем интенсивность возникновения ошибок после того, как обнаружена (i – 1) ошибка.

$$\lambda(t) = 0.012690 \cdot [14\text{-}9] = 0{,}06345.$$

По формуле (3.14) вероятность безотказной работы программного средства $P(t)$ = 0,94.

3.3 Расчет надежности информационной системы по модели Муса и выводы

В модели Муса надежность ПС на этапе эксплуатации оценивается по результатам тестирования [7].

Средняя наработка до отказа после тестирования определяется по формуле:

$$\tau = \tau_0 \, exp\left(\frac{CT}{M\tau_0}\right), \tag{3.22}$$

где τ_0 – средняя наработка до отказа до начала тестирования; C – коэффициент, учитывающий уплотнение тестового времени по сравнению с временем реальной эксплуатации.

Неизвестный параметр τ_0 можно оценить из следующего соотношения:

$$\tau_0 = \frac{1}{fKN}, \tag{3.23}$$

Средняя скорость исполнения одного оператора ПС рассчитывается по формуле:

$$f = \frac{A}{B}, \tag{3.24}$$

Общее время тестирования $T + \Delta T$ должно удовлетворять соотношению:

$$\tau = \tau_0 \, exp\left(\frac{C(T+\Delta T)}{M\tau_0}\right), \tag{3.25}$$

Надежность ПС для периода эксплуатации t определяются по формуле:

$$p(t) = e^{-\frac{t}{\tau}}, \tag{3.26}$$

Для данного программного модуля длительности этапов тестирования составляют t_1 = 19 часов, t_2 = 23 часов, t_3 = 28 часов. Число отказов на первом этапе m_1 = 3, на втором – m_2 = 5, на третьем – m_3 = 1. Средняя скорость исполнения ПС $A = 10^4$ операторов/час, количество операторов в ПС B = 740. Период эксплуатации t = 189 часов.

Найдем среднюю частоту выполнения одного оператора с помощью формулы (3.24):

$$f = \frac{10^4}{740} = 13{,}513 \text{ час}^{-1}.$$

Первоначальное количество ошибок в программном средстве (из модели Шумана) N = 4. Коэффициент проявления ошибок K для данного программного средства $1{,}5 \cdot 10^{-6}$, в таком случае τ_0 можно рассчитать по формуле (3.23):

$$\tau_0 = \frac{1}{13{,}513 \cdot 1{,}6 \cdot 10^{-6} \cdot 4} = 1156{,}29 \text{ часа}.$$

Один час тестирования соответствует 6 часам работы, следовательно, значение коэффициента C = 5. Тогда средняя наработка до отказа после тестирования на этапе эксплуатации ПО согласно с формулой (3.22) равна:

$$\tau = 1156{,}29 \cdot exp\left(\frac{5 \cdot (19 + 23 + 28)}{(3+5+1) \cdot 1156{,}29}\right) = 1195{,}80 \text{ часа}$$

Найдем надежность программного средства для периода эксплуатации t = 189 часов по формуле (3.26):

$$p(189) = e^{-\frac{189}{1195{,}80}} = 0{,}85.$$

Вывод по проведенных анализах расчета надежности информационной системы туристическое агентство, использующее модель сложности, модель Джелински-Моранда и модель Мусы.

Модель сложности позволила определить количество потенциальных ошибок, которые могут возникнуть в приложении. Результаты анализа показали, что надежность равна 0,9432.

С использованием модели Джелинского-Моранды была проведена оценка надежности приложения на основе анализа предыдущих сбоев. По полученным результатам установлено, что вероятность безотказной работы программного средства равна 0,94.

Для оценки надежности системы в целом была применена модель Муса. Анализ результатов показал, что вероятность отказа приложения для 189 часов эксплуатации составляет 0,85. Это свидетельствует о высокой надежности и стабильности приложения. Таким образом, можно сказать, что приложение обладает достаточной надежностью и высокой степени стабильности.

Однако важно понимать, что моделирование надежности является всего лишь математической моделью и не дает абсолютных гарантий. Возможны непредвиденные обстоятельства и потенциальные уязвимости, которые не были учтены при расчетах. В связи с этим анализ расчета надежности информационной системы туристического агентства с использованием модели сложности, модели Джелинского-Моранды и модели Мусы позволил выявить потенциальные уязвимости и принять меры по их устранению. Кроме того, результаты подтвердили высокую надежность и стабильность работы приложения.

4 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТУРИСТИЧЕСКОГО АГЕНТСТВА

В этом разделе необходимо протестировать информационную систему туристического агентства. на соответствие заданным требованиям и ожиданиям для тестовых сценарии был выбран функциональное тестирование

Основная цель раздела тестирования – убедиться, что информационная система работают корректно и соответствуют требованиям [23, 24].

Для реализации тестирование был разработан ряд тест-кейсов, в котором элементы программы были разделены на модули:

- главная страница;
- страница бронирование;
- страница туров;
- страница билетов.

Таблица 4.1– Тест-кейсы основных функции информационной системы.

Приоритет	Требование	Модуль	Подмодуль	Описание Тест- кейса	Ожидаемые результаты	Тест пройден
1	2	3	4	5	6	7
High	Корректной отображения главной страницы	Главная страница	Основная часть	Графический пользовательский интерфейс	Вывод корректно отображает графически интерфейс главной страницы	Да
High	Корректной отображения страницы «О нас»	Главная страница	Основная часть	Переход на страницу «О нас» Корректное отображение информации о кампании	Вывод Переход на страницу «О нас»	DR-1
High	Регистрация пользователя	Главная страница	Форма регистрация полей для ввода данных	Заполнение Формы регистрации почту пользователя, содержащее некорректные данные	Получено оповещение о некорректном вводе почты	Да

Продолжения таблица 4.1.

1	2	3	4	5	6	7
High	Регистрация пользователя	Главная страница	Форма регистрация полей для ввода данных	Заполнение Формы регистрации пользователя, содержащее корректные данные	Пользователь зарегистрирован и на почту пользователя отправлен ссылка для активации аккаунта	Да
High	Авторизация в системе	Главная страница	Форма авторизация полей для ввода данных	Заполнить форму корректные данные пользователя	В ход в личный кабинет	Да
Medium	Страница пользователя	Главная страница	Основная часть	Вывод на экран информацию о пользователя	Вывод Информация пользователя выводится на экран	Да
Medium	Поиск тура	Главная страница	Форма ввода поиска тура	Заполнить форму поиска и нажать на кнопку «Найти тур»	Вывод Исходя из имеющих туров форма поиска выдает соответствующий ответ	Да
High	Переход на страницу туров	Страница туров	Переход на страницу туров	Правильны переход на страницу туров	При переходе на страницу туров должна отображается список туров	Да
High	Добавить новый тур в каталог	Страница туров	Форма заполнения данных тура	Заполнить форму для добавления тура	Вывод отображения в каталоге туров	Да
High	Удалить тур в каталог	Страница туров	Выбор тура	Выбрать какой тур должен быть удален	Вывод Тур успешно удоляется	Да
High	Корректное отображения туров	Страница туро	Каталог всех туров	Отображения каталог туров	Отображения каталог туров на странице туров	Да
High	Корректно отобразить подробную информацию о туре	Страница туров	Информация о конкретном туре	Вывести информацию о туре для удобной восприятии информации	Отображено корректное информация о туре	Да

Продолжения таблица 4.1.

1	2	3	4	5	6	7
High	Правильное расположение элементов бронирование	Страница бронирование	Страница бронирование	Корректное отображение элементов страницы бронирование	Вывод Корректное отображение элементов на странице бронирование	DR-2
High	Переход на страницу бронирование	Страница бронирование	Основная часть	Переход на страницу бронирование	Правелный переходе на страницу бронирование	Да
High	Выявить о дефектах заполненных данных в форме, для оплаты части тура бронирование	Страница бронирование	Модальное окно форма заполнения данных броня для оплаты	Ввести некорректные данные в форму для бронирование тура	Вывод оповещения о некорректности введенных данных	Да
High	Список забронированных туров у пользователя	Страница бронирование	Таблица забронированных туров	При переходе на страницу забронированных туров пользователя должно отображаться список забронированных туров	Вывод список забронированных туров отображаться	Да
High	При запросе на сервер должен отправить список забронированных туров	Страница бронирование	Список забронированных туров пользователей	Админ отправляет запроса на сервер о забронированных туров сервер должен отправить список всех забронированных туров	Вывод Список забронированных туров	Да

Продолжение таблицы 4.1

1	2	3	4	5	6	7
High	Уведомления пользователя о не правленой заполнен ии контактных данных для бро нирование тура	Страница бронирова ние	Форма заполнение контактных данных	Проверить на дефекты при заполнения формы	Вывод оповеще ния о некоррек тном заполнен ии формы	DR-3
High	Отмена забронированн ого тура	Страница бронирова ние	Список забронирова нных туров пользователя	Пользовател выбирает из списка бронь и отменяет забронирова нны тур на сервере система автомотичес кий удоляет из базы данных забронирова нный тур пользователя	Вывод тур успешно отменен	Да
High	Переъход на страницу билетов	Страница билетов	Страница билетов	Переход на старницу билетов	Правель ный переход на старниц у билетов	Да
High	Корректное расположения элементов админа	Страница билетов	Основная часть	Панел администрат ора должны должен быть удобным	Вывод при переходе не должны сежать элемент ы панель админа	DR-4
High	Список билетов	Страница билетов	Список билетов пользователе й	Админ отправляет запроса на сервер о билетах сервер должен отправить список билетов	Вывод Список всех билетов на понели администр тартора	Да

Продолжения таблица 4.1.

1	2	3	4	5	6	7
High	Создание билета	Страница билетов	Билет пользователя	При оплате остальной части забронированного тура должен создастся билет и сервер автоматический удаляет из БД, затем создается билет и отправляется на почту пользователя	Вывод Удаляется из списка забронированных туров, создается билет и отправляется на почту созданный билет	Да
High	Создание билета не авторизованного пользователя	Страница билетов	Билет пользователя	Пользователь приобретает билет не регистрируясь в системе, сервер создает билет и высылает на почту созданный билет	Вывод Пользовател не регистрируюсь может приобрести тур	Да

На основе проеденных тест-кейсов была составлена [26] таблица 4.2 с оформленным описанием дефекта.

Таблица 4.2 – Описания найденных дефекта

№	Название дефекта	Важность	Алгоритм воспроизведения	Фактический результат	Ожидаемый результат	Приложение
1	2	3	4	5	6	7
1	Клиентская часть: Страница «О нас»: неправильная гиперссылка на страницу «О нас»:	*Major*	1. Вход на веб-сайт 2. Переход на страницу «О нас» 3. Переходит на страницу «Каталогов тура»	Правильный переход на страницу «О нас»	При переходе на страницу «О нас» должна отобразится информация о компании и команде	Рисунок 4.1 Рисунок 4.2 Рисунок 4.3

Продолжения таблица 4.2.

1	2	3	4	5	6	7
2	Клиентская часть: Отсутствие позиционирование кнопок страницы бронирование	*Major*	1. Вход на веб-сайт 2. Проходим авторизацию: client/client 3.Переходим на страницу туров 4. Выбираем тур 5. Переходим на страницу бронирование 6. Элементы расположены неправильно	Все элементы находятся на месте	Кнопка «кто поедет» должна находится в левом крае кнопка «Бронирование тура» Должна находиться в центре	Рисунок 4.4 Рисунок 4.5 Рисунок 4.6
3	Форма заполнение контактных данных: При заполнении пользовательской информации Отсутствует вариация	*Critical*	1. Вход на веб-сайт 2. Проходим авторизацию: client/client 3.Переходим на страницу туров 4. Выбираем тур 5. Переходим на страницу бронирование 6. Заполняем форму некорректными данными пользователя для бронирования тура 7. Уведомления о некорректности введенных данных 8 Повторные заполнения формы	Вывод Уведомляющий действия исчезают корректно	Вывод После заполнение и отправки формы корректными данными должны исчезнуть уведомляющие действия	Рисунок 4.7 Рисунок 4.8 Рисунок 4.9

Продолжения таблица 4.2.

1	2	3	4	5	6	7
4	Элементы админ панели: Страница администратора: При переходе в раздел билетов	*Major*	1. Вход на веб-сайт 2. Проходим авторизацию: admin/admin 3. Переходим на страницу admin 4. Переходим в раздел билетов	Все элементы админ панели отображаются правильно	Правильное отображения при переходе на разделы панели администратора	Рисунок 4.11 Рисунок 4.12

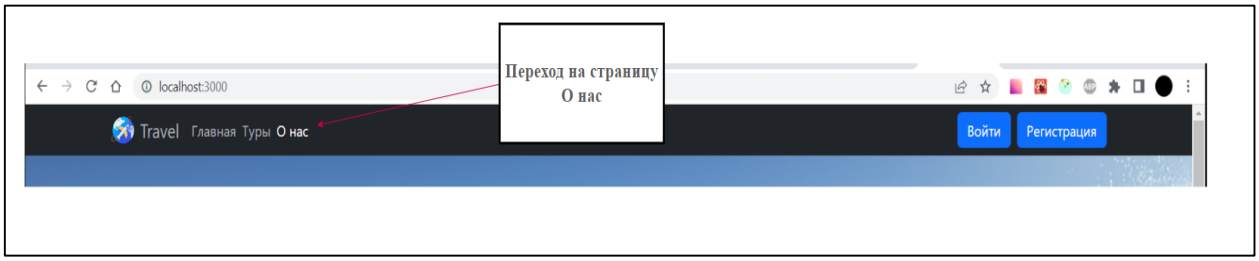

Рисунок 4.1 – Переход на страницу «О нас»

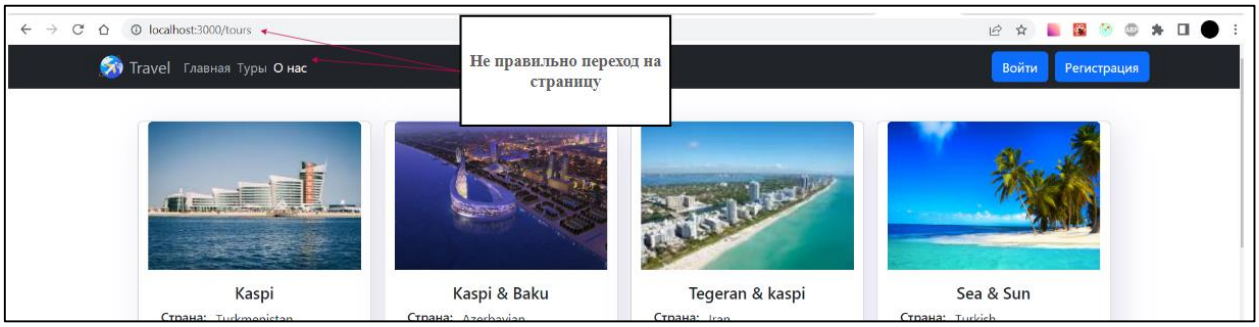

Рисунок 4.2 – Неправильная гиперссылка на страницу «О нас»

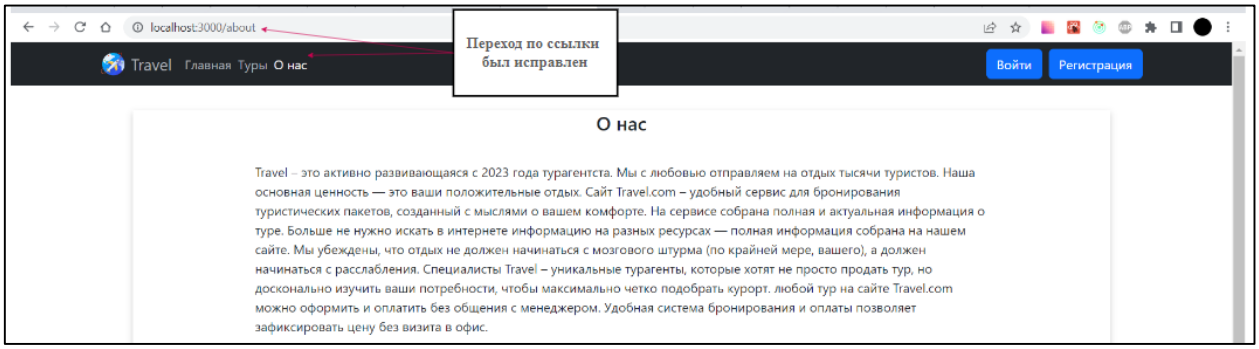

Рисунок 4.3 – Страница «О нас»

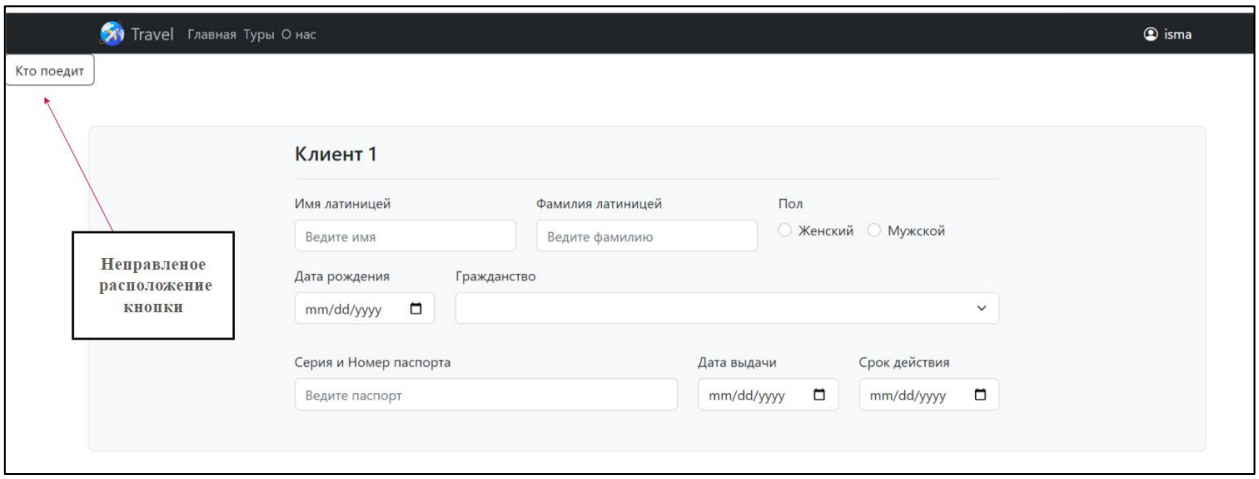

Рисунок 4.4 – Кнопка «Кто едет»

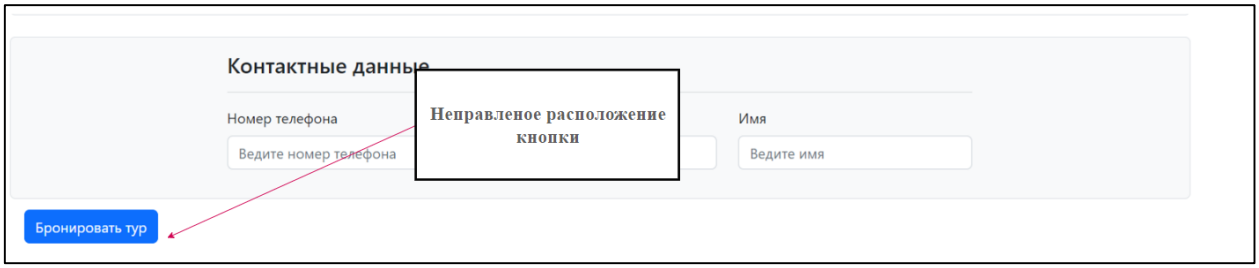

Рисунок 4.5 – Кнопка «Бронировать тур»

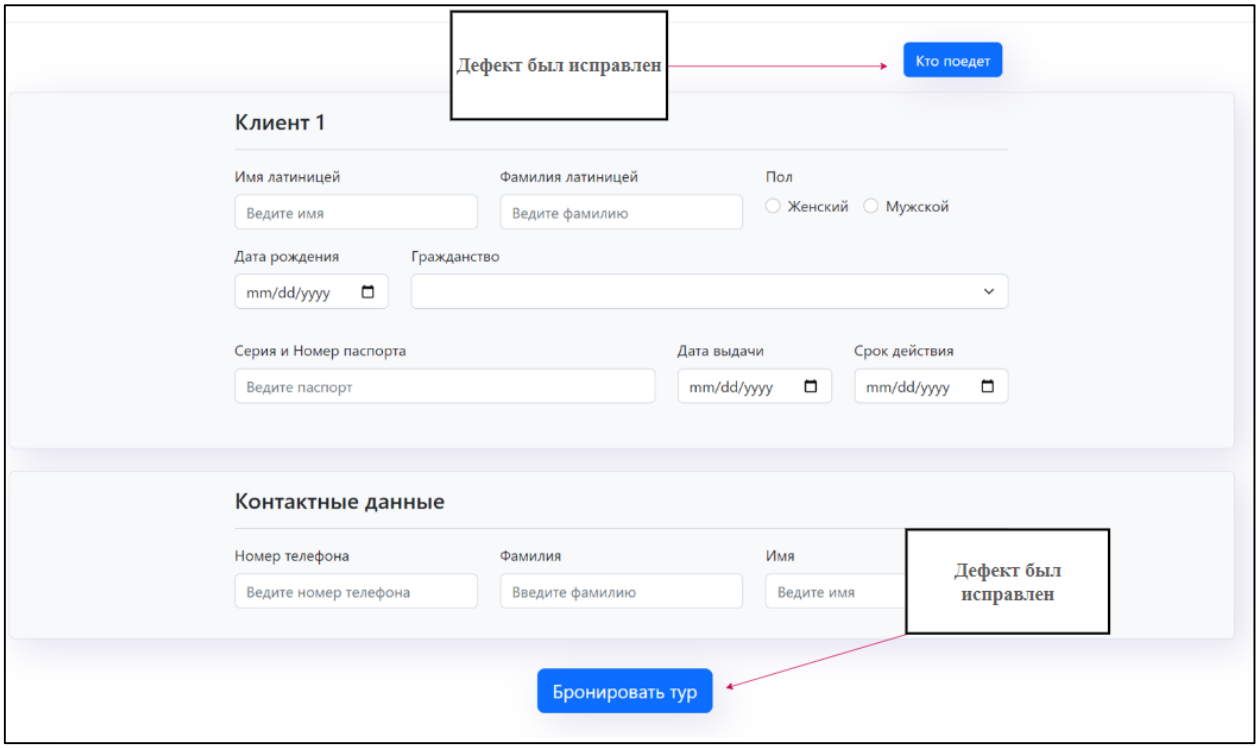

Рисунок 4.6 – Кнопка «Кто едет и Бронирование»

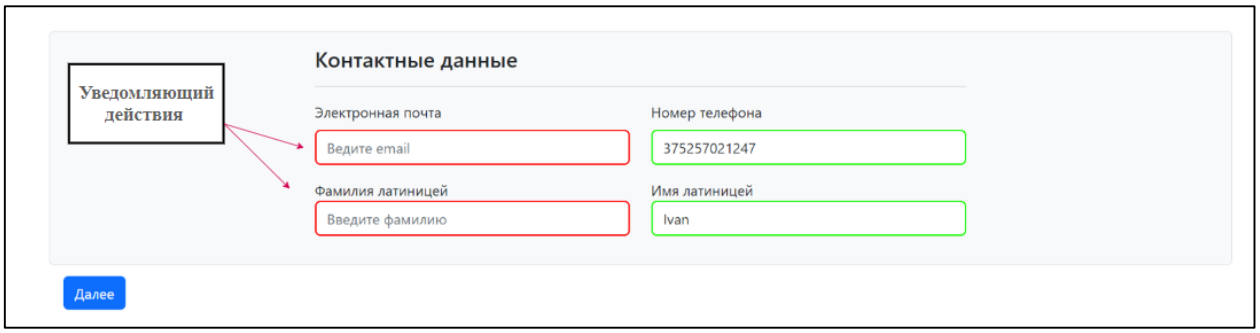

Рисунок 4.7 – Заполнение формы Некорректное данными

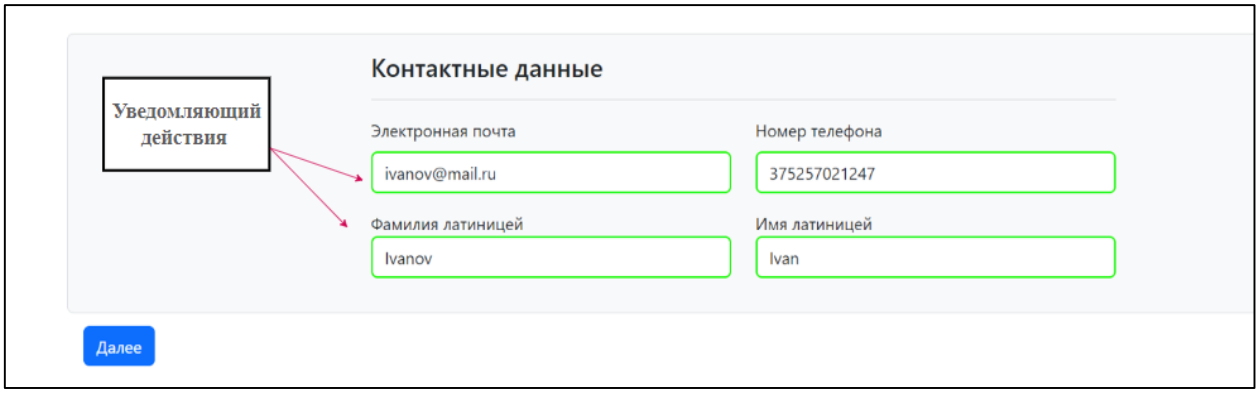

Рисунок 4.8 – Заполнение формы Корректное заполнение формы

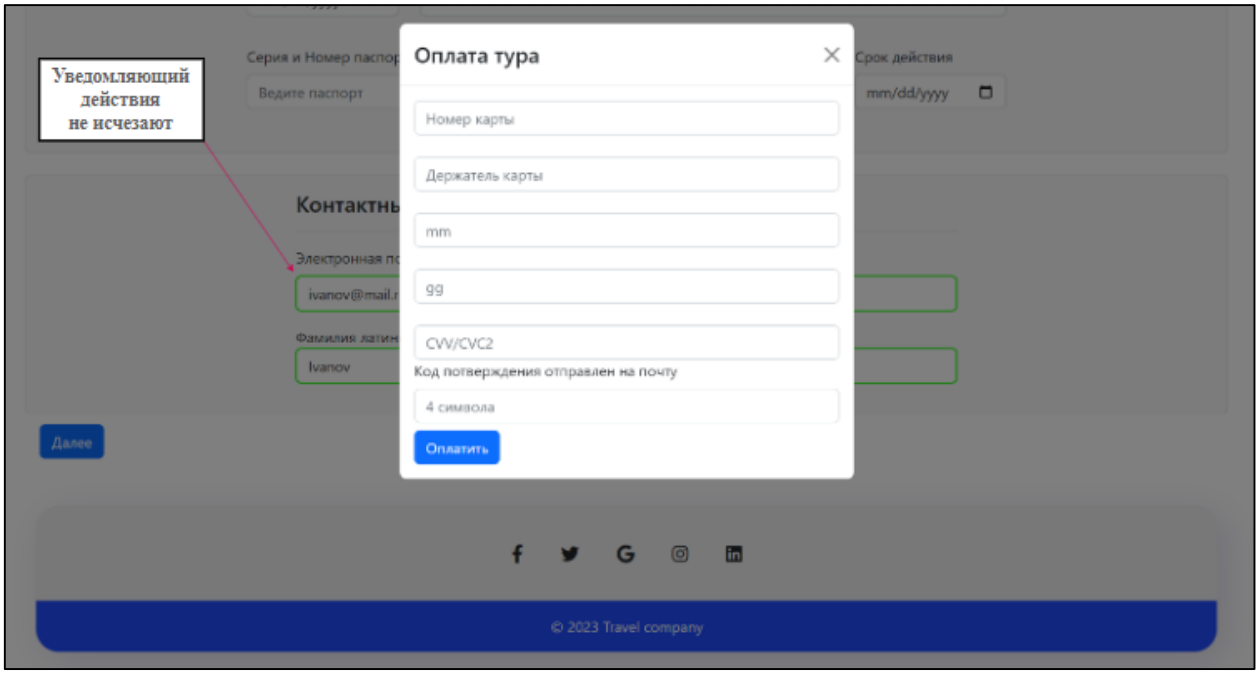

Рисунок 4.9 – Уведомляемые действия не исчезают

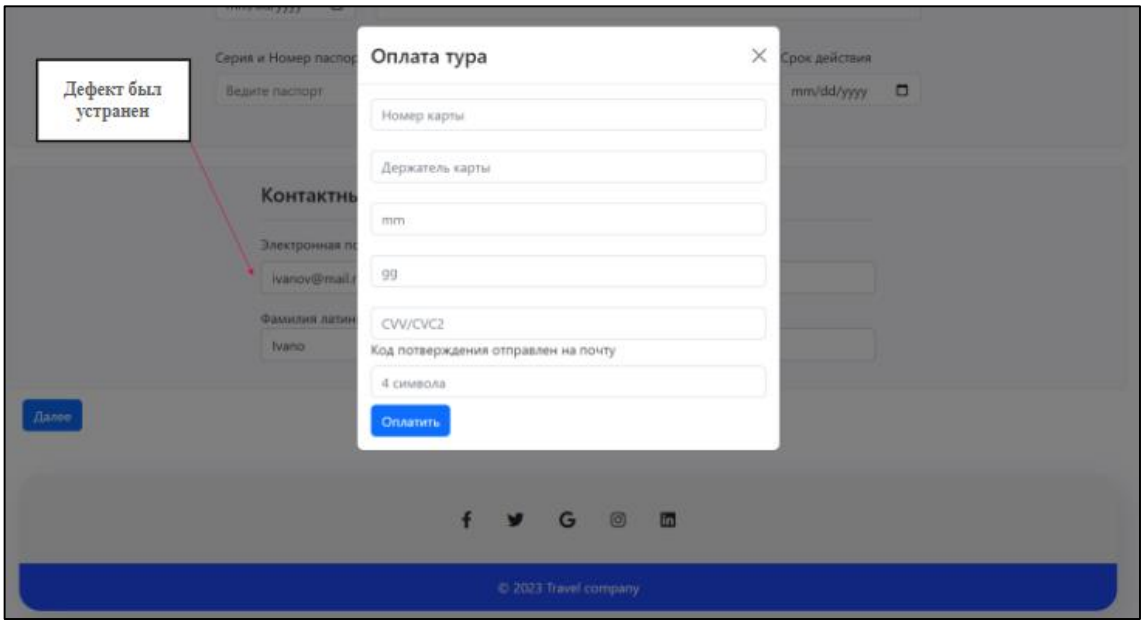

Рисунок 4.10 – Уведомляемые действия исчезают

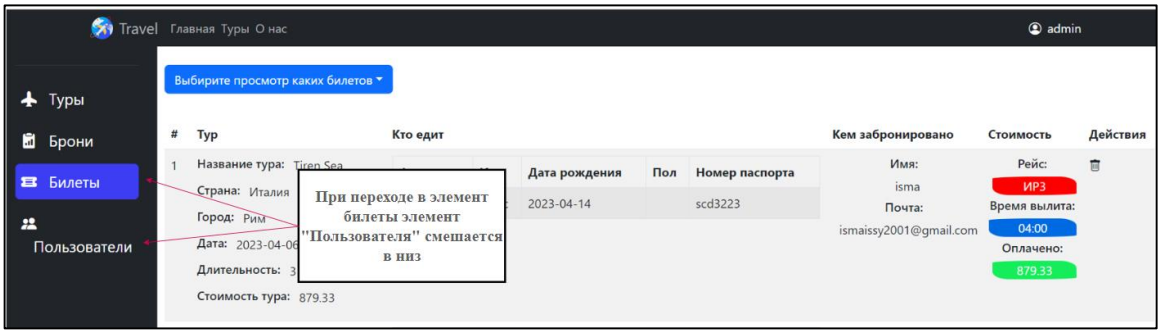

Рисунок 4.11 – Панель администратора «Билеты» смешения название «Пользователя»

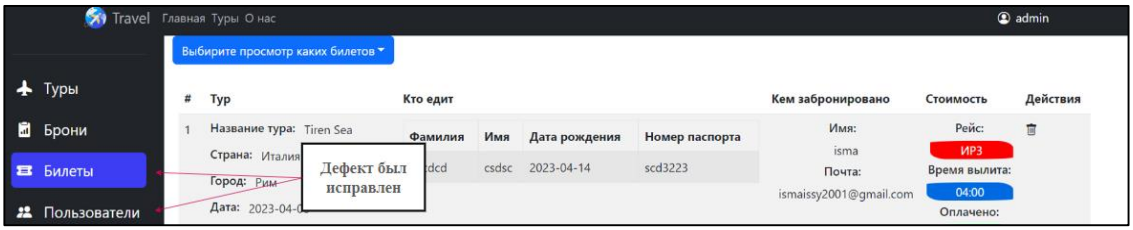

Рисунок 4.12 – Панель администратора «Билеты»

На основе тест-кейсов и дефект репорта был составлен отчет [25]. Предоставлен на рисунок 4.13 – Результат отчета тестирование

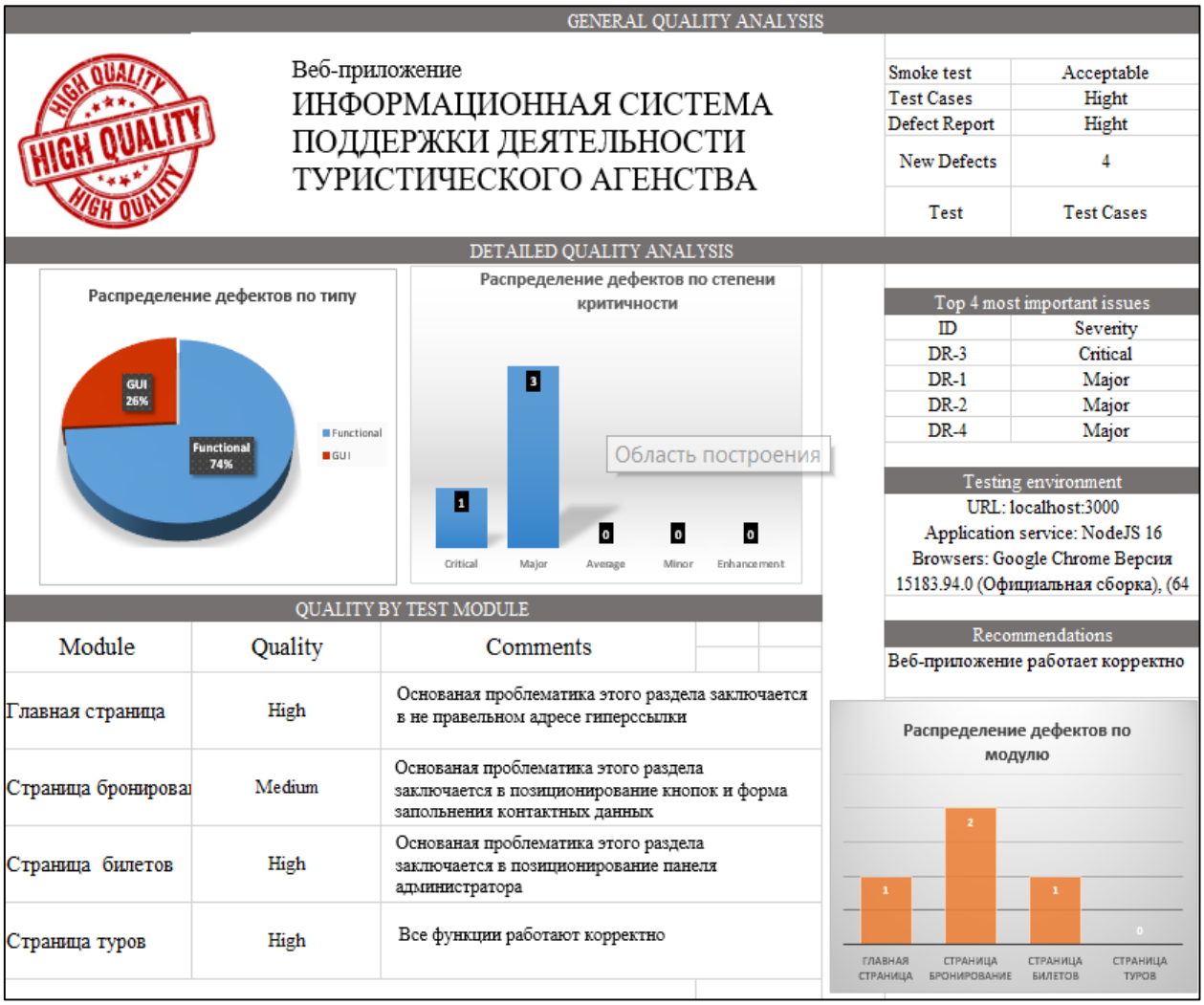

Рисунок 4.13 – Результат отчета тестирование

Вывод по разделу тестирование были выполнены все поставленные задач, а именно выполнения функциональной тестирование в модулях – главная страница, страница бронирование, страница туров, страница билетов.

Также были выявлены незначительные дефекты, а именно две функциональных и две пользовательского интерфейсных дефектов и были исправлены. По результату тестирование, программное средство работает корректно и соответствует поставленным требованиям в разделе 1.3. Программное средство готово к работе.

5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТУРИСТИЧЕСКОГО АГЕНТСТВА

5.1 Характеристика информационной системы

Целью разработки является создание информационной системы (ИС) для туристического агентства, которое позволит автоматизировать процессы продажи туров, управления бронирования туров. Разработанное программное средство будет использоваться в организации для улучшения эффективности работы туристического агентства и удовлетворения потребностей клиентов.

Актуальная потребность в разработке программного средства обусловлена увеличением объемов продаж и необходимостью улучшения качества обслуживания клиентов. Введение информационной системы позволит сократить время на обработку, уменьшить количество ошибок при бронировании тура. Также информационной системы позволит клиентам совершать онлайн покупки тура, что значительно расширит аудиторию информационной системы и повысит его конкурентоспособность.

Дополнительным фактором, обусловившим необходимость разработки программного средства, является рост конкуренции в отрасли отдыха и туризма. Согласно статистике, на 2021 год, более 2 миллиардов человек совершают покупки онлайн, что составляет около 27% населения мира. Также отмечается рост количества покупателей онлайн на 10-15% ежегодно, что свидетельствует о необходимости наличия у турагентство эффективной онлайн-платформы.

В результате внедрения программного средства, турагентство сможет значительно повысить эффективность работы и удовлетворить потребности клиентов. Ожидается увеличение объемов продаж на 20% в первый год использования системы и увеличение числа клиентов на 30%.

Экономическая целесообразность инвестиций в разработку программного продукта определяется на основе расчета и оценки следующих показателей [27]:

– чистая дисконтированная стоимость (ЧДД);

– срок окупаемости инвестиций (ТОК);

– рентабельность инвестиций ($Р_и$).

5.2 Расчет инвестиций в разработку программного средства

Часовая заработная плата определяется путем деления месячной заработной платы на количество рабочих часов в месяце. Среднемесячная расчетная норма рабочего времени на 202 год составляет 21 дня.

Расчет основной заработной платы ($З_о$) исполнителей:

$$З_о = К_{пр} \sum_{i=1}^{n} З_{чi} \cdot t_i, \quad (5.1)$$

где n – количество исполнителей, занятых разработкой программных средств;

$К_{пр}$ – коэффициент премий;

$З_{чi}$ – часовая заработная плата исполнителя (руб);

t_i – трудоемкость работ, выполняемых исполнителем i-й категории, определяется исходя из сложности разработки программного обеспечения и объема выполняемых им функций (час)

Список категорий исполнителей, их месячная и часовая заработная плата, а также трудоемкость работ представлены в таблице 5.1.

Согласно формуле 5.1 был произведен расчет основной заработной платы с учетом коэффициента премий, равного 1,5.

Таблица 5.1 – Расчет затрат на основную заработную плату команды разработчиков

Категория исполнителя	Месячная заработная плата, р.	Часовая заработная плата, р.	Трудоемкость работ, ч.	Итого, р.
Руководитель проекта	1650	9,82	65	638,3
Программист	1850	11,01	120	1321,2
Тестировщик	1000	5,95	35	208,25
Дизайнер	1200	7,14	57	406,98
Итого				2574,73
Премия				1287,36
Всего затраты на основную заработную плату разработчиков				3862,09

Дополнительная заработная плата разработчиков определяется по формуле:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (5.2)$$

где $Н_д$ – норматив дополнительной заработной платы, (10%).

После подстановки значений в формулу (5.2) дополнительная заработная плата составит:

$$З_д = \frac{3862,09 \cdot 10}{100} = 386,20 \text{ р.}$$

Для определения отчислений на социальные нужды воспользуемся формулой:

$$Р_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (5.3)$$

где $Н_{соц}$ – ставка отчислений в ФСЗН и Белгосстрах.

На момент написания проведения расчетов по формуле (5.3), в Республике Беларусь ставка отчислений в ФСЗН и Белгосстрах составляла 34,6%. Согласно формуле (5.3) размер отчислений в фонд социальной защиты населения и на обязательное страхование составит:

$$Р_{соц} = \frac{(3862,09+386,20) \cdot 34,6}{100} = 1469,90$$

Затраты по статье «Прочие расходы» ($Р_{пр}$), связанные с необходимостью содержания аппарата управления, а также расходами на общехозяйственные нужды, рассчитываются по формуле:

$$Р_{пр} = \frac{З_о \cdot Н_{пр}}{100}, \quad (5.4)$$

где $Н_{пр}$ – норматив накладных расходов (30%).

Расходы по статье «Накладные расходы» равны:

$$Р_{пр} = \frac{3862,09 \cdot 30}{100} = 1158,62 \text{ р.}$$

Общая сумма инвестиций (затрат) на разработку программного продукта рассчитывается по формуле:

$$З_р = З_о + З_д + Р_{соц} + Р_{пр} \quad (5.5)$$

Общая сумма инвестиций (затрат) на разработку:

$$З_р = 3862,09+386,20+1469,90+1158,62 = 6876,81 \text{ р.}$$

Таблица. 5.2 Результат расчета затрат

Наименование статьи затрат	Формула/таблица для расчета	Сумма, р.
1. Основная заработная плата разработчиков	Табл. 5.1	3862,09
2. Дополнительная заработная плата разработчиков	Формула (5.2)	386,20
3. Отчисления на социальные нужды	Формула (5.3)	1469,90
4. Прочие расходы	Формула (5.4)	1158,62
5. Общая сумма инвестиций (затрат) на разработку	Формула (5.5)	6876,81

5.3 Расчет экономического эффекта от использования программного средства

Экономическим эффектом в результате использования программного средства является прирост чистой прибыли, полученный за счет:

– экономии затрат на заработную плату с начислениями на заработную плату служащих в связи с сокращением их численности;

– экономии материальных затрат, электроэнергии, затрат на оплату труда и пр. в результате снижения брака, технологических потерь;

– снижения себестоимости продукции (работ, услуг) в результате роста производительности труда;

– снижения затрат на заработную плату с начислениями на заработную плату основных производственных рабочих;

– снижения материальных затрат на производство продукции (работ, услуг) и т. п.

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ [27]:

$$Э_{з.п.} = К_{пр} \cdot (t_р^{без\ п.с.} - t_р^{с\ п.с.}) \cdot Т_ч \cdot N_п \cdot (1 + \frac{Н_д}{100}) \cdot (1 + \frac{Н_{соц}}{100}), \quad (5.6)$$

где $К_{пр}$ – коэффициент премий;

$t_р^{без\ п.с.}$, $t_р^{с\ п.с.}$ – трудоемкость выполнения работ сотрудниками до и после внедрения программного средства, ч;

$Т_ч$ – часовой оклад (часовая тарифная ставка) сотрудника, использующего программное средство, р.;

$N_п$ – плановый объем работ, выполняемых сотрудником;

$Н_д$ – норматив дополнительной заработной платы (10 %);

$Н_{соц}$ – ставка отчислений от заработной платы, включаемых в себестоимость (34,6 %).

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ:

$$Э_{з.п.} = 1 \cdot (8 - 6{,}5) \cdot 8{,}63 \cdot 21 \cdot \left(1 + \frac{10}{100}\right) \cdot \left(1 + \frac{34{,}6}{100}\right) = 402{,}49$$

Экономическим эффектом при использовании программного средства является прирост чистой прибыли, полученной за счет экономии на текущих затратах предприятия [27], который рассчитывается по формуле:

$$\Delta П_ч = (Э_{тек} - \Delta Э_{тек}^{п.с})(1 - \frac{Н_п}{100}), \quad (5.7)$$

где $Э_{тек}$=$Э_{з.п.}$ – экономия на текущих затратах при использовании программного средства, р.;

$\Delta Э_{тек}^{п.с}$ – прирост текущих затрат, связанных с использованием программного средства (затраты на сопровождение программного средства, затраты на интернет-трафик и т. п.), р.;

$Н_п$ – ставка налога на прибыль согласно действующему законодательству (18 %).

Экономическим эффектом при использовании программного средства

$$\Delta П_ч = (402{,}49 - 140)\left(1 - \frac{18}{100}\right) = 215{,}24$$

5.4 Расчет показателей экономической эффективности разработки и использования программного средства

Приведение доходов и затрат к настоящему моменту времени осуществляется посредством так называемого дисконтирования, т. е. путем их умножения на коэффициент дисконтирования, который определяется по формуле:

$$\alpha = \frac{1}{(1+d)^{t-t_p}}, \quad (5.8)$$

где d – требуемая норма дисконта, которая по своему смыслу соответствует устанавливаемому инвестором желаемому уровню рентабельности инвестиций, доли единицы; t – порядковый номер года, доходы и затраты которого

приводятся к расчетному году; t_p – расчетный год, к которому приводятся доходы и инвестиционные затраты (t_p=1).

$$\alpha_1 = \frac{1}{(1+0{,}11)^0} = 1.$$

$$\alpha_2 = \frac{1}{(1+0{,}11)^1} = 0{,}90.$$

$$\alpha_3 = \frac{1}{(1+0{,}11)^2} = 0{,}81.$$

$$\alpha_4 = \frac{1}{(1+0{,}11)^3} = 0{,}73.$$

Чистый дисконтированный доход определяется по формуле:

$$\text{ЧДД} = \sum_{t=1}^{n} \Delta\text{П}_{\text{чт}} \cdot \alpha_t - \sum_{t=1}^{n} З_t \cdot \alpha_t, \qquad (5.9)$$

Результаты расчета показателей эффективности сведены в таблице 5.3.

Таблица 5.3 – Расчет эффективности инвестиций (затрат) в реализацию проектного решения

Показатель	Расчетный период			
	2024	2025	2026	2027
1. Прирост чистой прибыли, р.	2584,08	2868,34	3183,85	3534,08
2. Дисконтированный результат, р.	2584,08	2581,50	3534,07	3922,82
3. Инвестиции (затраты) в реализацию проектного решения, р.	6876,81			
4. Дисконтированные инвестиции, р.	6876,81			
5. Чистый дисконтированный доход по годам, р.	-4292,73	-1711,23	1822,84	3922,82
6. Чистый дисконтированный доход нарастающим итогом, р.	-4290,93	-2579,7	-756,86	3165,96
7. Коэффициент дисконтирования, доли единицы	1	0,90	0,81	0,73

Для определения экономической целесообразности инвестиций в разработку и использование необходимо определить чистый дисконтированный доход, срок окупаемости инвестиций и их рентабельность.

Рассчитаем среднюю норму прибыли/рентабельности инвестиций ($Р_и$) по формуле:

$$Р_и = \frac{\frac{1}{n}\sum_{t=1}^{n} ΔП_{чt}}{\sum_{t=1}^{n} З_t} \cdot 100\%, \quad (5.10)$$

где n – расчетный период, лет;

$З_t$ – затраты (инвестиции) в году t, так как затраты уходят сразу на разработку то независимо затраты будут равны $З_р$;

$ΔП_{чt}$ – прирост чистой прибыли в году t в результате реализации проекта, р.

Средняя норма прибыли/рентабельности инвестиций:

$$Р_и = \frac{\frac{1}{4} \cdot 2584{,}08 + 2868.34 + 3183.58 + 3534.08}{6876{,}81} \cdot 100\% = 44{,}2\%$$

Рассчитаем оценку экономической эффективности разработки ($T_{ок}$) по формуле:

$$T_{ок} = \frac{\sum_{t=1}^{n} З_t}{\frac{1}{n}\sum_{t=1}^{n} ΔП_{чt}}, \quad (5.11)$$

Время окупаемости инвестиции при внедрении программного средства:

$$T_{ок} = \frac{6876{,}8}{\frac{1}{4} \cdot 12170{,}08} = 2{,}26$$

В результате технико-экономического обоснования разработки программного продукта были получены следующие значения показателей эффективности:

– чистый дисконтированный доход (ЧДД) за четыре года составил 3165,96 р;

– окупаемость инвестиций наступает на четвертый год;

– рентабельность инвестиций по проекту составляет 44,2%.

Таким образом, разработка и продажа программного продукта является эффективным и инвестиции в его разработку целесообразно осуществлять.

6 ОХРАНА ТРУДА. РАЗРАБОТКА МЕРОПРИЯТИЙ ПО ПОВЫШЕНИЮ РАБОТОСПОСОБНОСТИ И ПРОИЗВОДИТЕЛЬНОСТИ ТРУДА РАЗРАБОТЧИКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ПОДДЕРЖКИ ДЕЯТЕЛЬНОСТИ ТУРИСТИЧЕСКОГО АГЕНТСТВА

Целью дипломного проекта явилась разработка информационной системы (ИС) [28], для туристического агентства. Веб-приложение позволяет автоматизировать определенные задачи в заданной области информационной системы. Приложение включает в себя различные компоненты, такие как серверная часть, клиентская часть и базы данных, которые взаимодействуют друг с другом для реализации необходимых функций.

Актуальность разработки данной информационной системы заключается в упрощении процесса оформления туристических путевок и повышении эффективности работы турагентства. Для достижения этой цели были учтены основные принципы удобства использования и быстроты доступа к необходимой информации, что обеспечивает удобное и продуктивное использование данной информационной системы.

Во время разработки ИС [29], условия труда играют важную роль в производительности разработчика ИС. Рабочее место, здоровье и настроение работника, доступность необходимых ресурсов и технологий - все эти факторы могут повлиять на качество и скорость работы разработчика.

Физические условия рабочего места могут включать такие факторы, как, неудобное расположение оборудования, шум и плохое освещение. Неудобная мебель и оборудование могут вызвать напряжение в мышцах, боли в шее, спине и руках, а также усталость глаз. Шум может вызвать раздражение и отвлечение, что может снизить концентрацию и эффективность работы. Плохое освещение может привести к ухудшению зрения и быстрой усталости глаз. Работающие в таких условиях, могут столкнуться с уменьшением производительности, ухудшением здоровья и настроения.

В процессе разработки ИС программист работает в кабинете, где расположено его рабочее место за компьютером. Трудовая деятельность в основном связана с использованием монитора, компьютера, мыши и клавиатуры, поэтому эти инструменты являются основными элементами рабочего места.

Время разработки ИС велась три месяца, все рабочие дни недели. Начала рабочего дня начиналась с 9:00 до 18:00 обеденное перерыв входит в это время, работа за компьютером составляло 8 часов в день.

Условия труда на конкретном рабочем месте могут оказывать существенное влияние на производительность работника, особенно если речь идет о разработчике информационной системы. Следует уделить большое внимание комфорту рабочего места. Рабочее место программиста и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. При организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Важным фактором является положение тела и рабочая поза человека. Неудобная поза может привести к чрезмерным статическим нагрузкам на позвоночник, невротическим состояниям и патологическим нарушениям со стороны опорно-двигательного аппарата и внутренних органов. Также важно организовать рабочую зону таким образом, чтобы центр тяжести тела человека был расположен в пределах площади его опоры. Неустойчивое положение тела может привести к быстрому утомлению, травмам и заболеваниям опорно-двигательного аппарата, что также может отрицательно сказаться на производительности работника. Поэтому важно учитывать эти факторы при организации рабочего места для разработчика информационной системы.

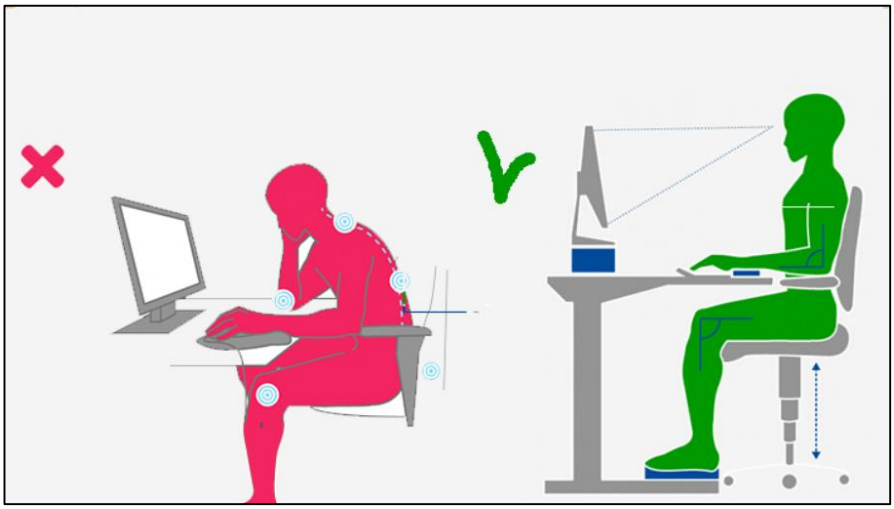

Рисунок 6.1 – Схема биомеханического анализа рабочей позы при неустойчивом (а), и устойчивом положениях (б)

Выбор оптимальной рабочей позы (сидя, стоя, сидя-стоя) зависит от физической нагрузки на работника. При работе с небольшими усилиями (не более 50 Н) рекомендуется использовать рабочую позу сидя. При работе с усилиями от 50 до 100 Н, рабочую позу можно выбрать на свое усмотрение, так как эффект на организм при сидении или стоянии будет примерно одинаковым. Однако, при работе с усилиями, превышающими 100 Н, рекомендуется работать стоя [30].

Работа стоя может быть эффективной для наладки оборудования, но увеличивает нагрузку на нижние конечности и требует больше энергии. Работа в позе сидя более рациональна и менее утомительна, но может приводить к застою в органах таза. Смена позы помогает перераспределить нагрузку и улучшить кровообращение, поэтому рекомендуется предусматривать возможность для работников менять положение тела. Для профессионалов в области разработки ИС важно оптимизировать условия труда, чтобы избежать заболеваний и повысить продуктивность.

Кресло является неотъемлемой частью рабочего места, и его удобство имеет большое значение. Для комфортной работы сотрудник должен иметь кресло с удобным сиденьем, поддержкой спины и подлокотниками, который позволит ему сидеть в течение длительного времени [31].

Существует множество различных типов кресла, которые могут помочь уменьшить нагрузку на спину и шею. Некоторые кресла позволяют регулировать высоту, наклон спинки и сиденья, что позволяет сотруднику настроить их под свои индивидуальные потребности.

Рисунок 6.2 – Кресло

Одним из ключевых элементов правильного рабочего места является стол. Стол должен иметь достаточную площадь, чтобы разместить компьютер,

клавиатуру, мышь, документы и другие необходимые материалы. Он должен быть достаточно прочным и стабильным, чтобы не дрожал при нажатии на клавиши клавиатуры или движении мыши. Высота стола должна соответствовать росту сотрудника, чтобы предотвратить излишнее напряжение в шее, плечах и спине. Регулируемая высота стола может помочь удовлетворить потребности сотрудников с разным ростом. На рисунке 6.3 показан пример стола для организации рабочего места.

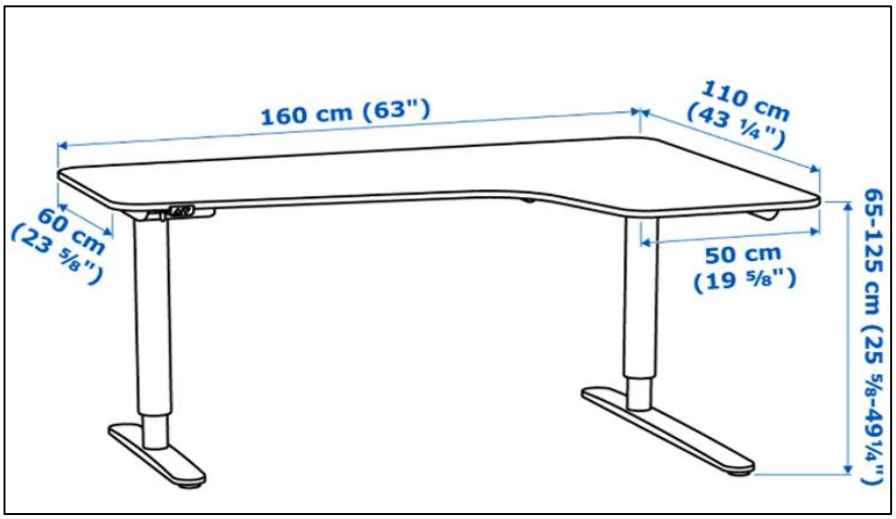

Рисунок 6.3 – Пример стола для организации рабочего места

Немаловажное значение имеет положение экрана монитора при работе программиста. Также должна также предусматриваться возможность регулирования экрана по высоте и наклону. Во время пользования компьютером советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже. Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются слезной жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости [31].

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет 60-80 см, то высота знака должна быть не менее 3 мм, оптимальное соотношение ширины и высоты знака составляет 3:4,

а расстояние между знаками - 15-20% их высоты. Соотношение яркости фона экрана и символов - от 1:2 до 1:15.

Для улучшения условий труда и повышения работоспособности разработчика информационной системы на его рабочем месте, можно применить следующие предложения и рекомендации:

1 Эргономичная организация рабочего места. Рабочее место должно быть организовано таким образом, чтобы соответствовать физиологическим особенностям разработчика. Например, стол и кресла должны иметь оптимальные размеры, высота стола должна соответствовать росту разработчика, кресла должен иметь удобное сиденье с поддержкой спины и подлокотниками.

2 Регулярные перерывы. Длительное время в одной и той же позе может привести к усталости и боли в мышцах и суставах. Поэтому необходимо делать регулярные перерывы в течение рабочего дня, чтобы размяться и расслабиться. Рекомендуется делать перерыв каждые 45-60 минут.

3 Установка правильной освещенности. Освещение должно быть достаточным, равномерным и не вызывать ослепления. Разработчик должен иметь возможность регулировать яркость света на своем рабочем месте.

4 Регулярные упражнения для разминки. Регулярные упражнения для разминки помогут снять напряжение с мышц и напряженности суставов, уменьшить усталость и улучшить кровообращение.

5 Использование специализированного оборудования. Существует большое количество специального оборудования, такого как эргономические клавиатуры и мыши, которые могут помочь уменьшить нагрузку на суставы и мышцы рук и предотвратить синдром запястного канала.

В целом, проблемы, связанные с условиями труда [8], могут значительно повлиять на производительность и качество работы разработчика информационной системы. Решение этих проблем может повысить производительность и эффективность работы, а также улучшить качество жизни разработчика и снизить риск заболеваний и стресса.

Разработанные предложения и рекомендации по повышению работоспособности и производительности труда разработчика информационной системы, оптимизация условий труда для разработчиков ИС является важным фактором для повышения эффективности работы и предотвращения заболеваний. Для улучшения условий труда на конкретном рабочем месте необходимо уделить внимание выбору эргономичной мебели и оборудования, оптимизации освещения, а также организации рабочего пространства и разработке эффективных рабочих процессов.

ЗАКЛЮЧЕНИЕ

В результате дипломного проектирование проведен анализ предметной области, рассмотрены аналоги разрабатываемого программного средства, на основе выявленных в них достоинств и недостатков составлены требования по разработке структуры приложения и алгоритм работы программы, разработан алгоритм работы пользователя, разработан расчет надежности информационной системы туристического агентства, произведено тестирование веб-приложения, сделано технико-экономическое обоснование разработки.

В программе реализован множественный функционал, в разной степени доступный пользователям в зависимости от их прав: поиск информации о турах; заказ и бронирование туров; покупка билетов и услуг; внесение и редактирование информации о турах и клиентах; управление процессами бронирования и оплаты туров. Данный продукт может быть использован, как сотрудниками, так и простыми пользователями. В программе предусмотрена обработка ошибок, которые могут возникнуть в процессе работы. Безопасность обеспечивается за счет разграничения прав доступа, системы авторизации. Применение базы данных в качестве хранилища информации позволяет оптимально и эффективно хранить информацию, ее структурировать

Помимо бизнес-логики также были определены требования к работе, внешнему виду и содержанию системы, а также были определены основные сущности системы для организации моделей. В процессе разработки, приложение было протестировано, в результате не было выявлено серьезных ошибок, а все выявленные незначительные ошибки были устранены.

Произведен расчет надежности информационной системы. по моделям. Надёжность веб-ресурса в соответствии с моделями сложности, Муса, Джелинского-Моранда составляет более 80%, что говорит о высокой надежности данной программного средства.

В результате технико-экономического обоснования разработки программного продукта рассчитан чистый дисконтированный доход. Рентабельность инвестиций по проекту составляет 44,2%, окупаемость инвестиций наступает на четвертый год.

Разработаны рекомендации по повышению работоспособности и оптимизации производительности труда разработчика.

Разработанная информационная система позволит автоматизировать такие бизнес-процессы туристической компании как управление заказами, учет финансовых операций и учет клиентов, что способствует повышению производительности и качества работы сотрудников туристического агентства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Национальная стратегия устойчивого развития Республики Беларусь до 2035 года [Электронный ресурс]. – Режим доступа: NSUR-2035-1.pdf (economy.gov.by). – Дата доступа: 02.04.2023.

[2] Миронова Н.А. Туристская отрасль в контексте цифровой экономики [Элекронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/turistskaya-otrasl-v-kontekste-tsifrovoy-ekonomiki. – Дата доступа: 05.03.2023.

[3] Костюченко Е.П. Возможности цифровизации в туриндустрии // Интеллектуальные ресурсы – региональному развитию. 2022. № 1. С. 249-255.

[4] Григорьева Н.С., Колычева Ж.Я., Дынник Д.И. Цифровая трансформация системы управления туристским комплексом // Наука и образование: хозяйство и экономика; предпринимательство; право и управление. – 2021. – № 1 (128). – С. 40-44.

[5] Развития туризма в Туркменистане [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/potentsial-osobennosti-razvitiya-perspektivy-turizma-v-turkmenistane – Дата доступа: 02.04.2023.

[6] Иконников, В. Ф. Информационные технологии в индустрии туризма : учеб.-метод. пособие /В. Ф. Иконников, М. Н. Садовская. – Минск : РИПО, 2014. – 78 с.

[7] Туризм и туристические ресурсы в Республике Беларусь : буклет / под ред. И. В. Медведевой. – Минск : Национальный статистический комитет Республики Беларусь, 2021. – 31 с.

[8] Козлов М.В., Брыксин В.Е., Немчинова Е.Е. Факторы влияния цифровизации на туристский бизнес // Вестник Алтайской академии экономики и права. – 2023. – № 2-1. – С. 52-56.

[9] Toursoyuz.by [Электронный ресурс]. – Режим доступа: https://www.toursoyuz.by/2021/12/20/rejtingi-belorusskih-turoperatorov-s-kem-sotrudnichali-agenty/. – Дата доступа: 01.04.2023.

[10] Viapol.by [Электронный ресурс]. – Режим доступа: https://viapol.by/index.htm. – Дата доступа: 01.04.2023.

[11] Tez-tour [Электронный ресурс]. – Режим доступа: https://www.tez-tour.com/ – Дата доступа: 01.04.2023.

[12] T-v.by [Электронный ресурс]. – Режим доступа: https://t-v.by/about-company/about-company/. – Дата доступа: 01.04.2023.

[13] Азади [Электронный ресурс]. – Режим доступа: http://www.azady.co.tm/turizm_ru.html – Дата доступа: 05.04.2023.

[14] Понятие информационных систем [Электронный ресурс]. – Режим доступа: https://www.vsavm.by/knigi/kniga3/220.html– Дата доступа: 06.04.2023.

[15] MERN [Электронный ресурс]. – Режим доступа: https://blog.logrocket.com/mern-stack-tutorial/ – Дата доступа: 24.02.2023.

[16] База данных mongodb [Электронный ресурс]. – Режим доступа: https://www.mongodb.com/ – Дата доступа: 13.03.2023.

[17] Express фреймворк [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs – Дата доступа: 06.03.2023.

[18] React [Электронный ресурс]. – Режим доступа: https://ru.legacy.reactjs.org/ – Дата доступа: 06.03.2023.

[19] Bootstrap [Электронный ресурс]. – Режим доступа: https://getbootstrap.com/ – Дата доступа: 22.03.2023.

[20] Redux [Электронный ресурс]. – Режим доступа: https://redux.js.org/ – Дата доступа: 05.03.2023.

[21] Nodejs [Электронный ресурс]. – Режим доступа: https://metanit.com/web/nodejs/1.1.php – Дата доступа: 05.03.2023.

[22] Правила и этапы построения надежного программного обеспечения [Электронный ресурс]. – Режим доступа: https://www.kaznu.kz/content/files/news/folder22810/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%20-08_2020.pdf/ – Дата доступа: 07.05.2023.

[23] Меженная М. М. [и др.] Юзабилити-тестирование программного обеспечения: пособие / М. М. Меженная. – Минск : БГУИР, 2017. – 72 с.

[24] Стотлемайер Д. Тестирование Web-приложений / Д. Стотлемайер. – М. : Кудиц-образ. 2003. – 240 с.

[25] Куликов, С. С. Тестирование программного обеспечения. Базовый курс : практ. пособие. / С. С. Куликов. – Минск: Четыре четверти, 2015. – 294 с.

[26] О.С. Медведев, М.М. Меженная, Т.В. Гордейчук, М.М. Борисик, И.Ф. Киринович. Юзабилити-тестирование программного обеспечения: пособие [Электронный ресурс]. – Режим доступа: https://bit.ly/3Wgai0O – Дата доступа: 10.05.2023.

[27] Экономика проектных решений: методические указания по экономическому обоснованию дипломных проектов : учебно-методическое пособие / Горовой В. Г. [и др.]. – Минск : БГУИР, 2021. – 107 с.

[28] Михнюк Т.Ф. Охрана труда. Учебное пособие для вузов / Т.Ф. Михнюк. – Минск: Вышэйшая школа, 2007. – 335 с.

[29] Аттестация рабочих мест по условиям труда //Библиотека журнала «Ахова працы». – 2003. – № 10.

[30] Шупейко, И. Г. Эргономическое проектирование систем «человек – машина»: пособие / И. Г. Шупейко. – Минск: БГУИР, 2017. – 80 с.

[31] Валетко О. Рекомендации по проведению оценки интеллектуальных и сенсорных нагрузок при проведении аттестации рабочих мест по условиям труда [Электронный ресурс]. – Режим доступа: https://www.nitt.by/izdaniya/nitt/rekomendatsii-po-provedeniyu-otsenki-int – Дата доступа: 3.05.2023.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программы

Файл – bookingModel.js

```
const { Schema, model } = require("mongoose");
const BookingSchema = new Schema({
  user: { type: Schema.Types.ObjectId, ref: "User" },
  tour: { type: Schema.Types.ObjectId, ref: "Tour" },
  pay: { type: Number, required: true },
  needToPay: { type: Number, required: true },
  price: { type: Number, required: true },
  customers: { type: Array, required: true },
});
module.exports = model("Booking", BookingSchema);
```

Файл - ticketModel.js

```
const { Schema, model } = require("mongoose");
const TicketSchema = new Schema({
  userId: { type: Schema.Types.ObjectId, ref: "User" },
  tourId: { type: Schema.Types.ObjectId, ref: "Tour" },
  email: { type: String, required: true },
  cardNumber: { type: String, required: true },
  cardId: { type: String, required: true },
  paid: { type: Number, required: true },
  clock: { type: String, required: true },
  flight: { type: String, required: true },
  pathName: { type: String, required: true },
  customers: { type: Array, required: true },
  flag: { type: Boolean, required: true, default: true },
});
module.exports = model("Ticket", TicketSchema);
```

Файл – ticketNoUser.js

```
const { Schema, model } = require("mongoose");
const TicketNoUser = new Schema({
  tourId: { type: Schema.Types.ObjectId, ref: "Tour" },
  firstName: { type: String, required: true },
  lastName: { type: String, required: true },
  email: { type: String, required: true },
  phoneNumber: { type: String, required: true },
  cardNumber: { type: String, required: true },
  cardId: { type: String, required: true },
  paid: { type: Number, required: true },
  clock: { type: String, required: true },
  flight: { type: String, required: true },
  pathName: { type: String, required: true },
```

```
  customers: { type: Array, required: true },
  flag: { type: Boolean, required: true, default: true },
});
module.exports = model("TicketNoUser", TicketNoUser);
```

Файл – tokenModel.js

```
const { Schema, model } = require("mongoose");
const TokenSchema = new Schema({
  user: { type: Schema.Types.ObjectId, ref: "User" },
  refreshToken: { type: String, required: true },
});
module.exports = model("Token", TokenSchema);
```

Файл – tourModel.js

```
const { Schema, model } = require("mongoose");
const TourSchema = new Schema({
  name: { type: String, required: true },
  type: { type: String, required: true },
  date: { type: String, required: true },
  country: { type: String, required: true },
  city: { type: String, required: true },
  price: { type: Number, required: true },
  duration: { type: Number, required: true },
  flag: { type: Boolean, default: true },
  linkPhoto: { type: String },
});
module.exports = model("Tour", TourSchema);
```

Файл – userModel.js

```
const { Schema, model } = require("mongoose");
const UserSchema = new Schema({
  email: { type: String, unique: true, required: true },
  password: { type: String, required: true },
  isActivated: { type: Boolean, default: false },
  activationLink: { type: String },
  role: { type: String, default: "USER" },
  nickName: { type: String, required: true },});
module.exports = model("User", UserSchema);
```

Файл – validCardId.js

```
const { Schema, model } = require("mongoose");
const CardValidationId = new Schema({
  validateCard: { type: String, required: true },
  email: { type: String, required: true },
});
module.exports = model("CardValidation", CardValidationId);
```

Файл – index.js

```
require("dotenv").config();
const express = require("express");
const bodyParser = require('body-parser');
const mongoose = require("mongoose");
const userRouter = require("./routes/user/useRoutes");
const tourRouter = require("./routes/tour/tourRoutes");
const cors = require("cors");
const cookieParser = require("cookie-parser");
const PORT = process.env.PORT || 5011;
const app = express();
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json());
app.use(express.json());
app.use(cookieParser());
app.use(cors({ credentials: true,
origin: process.env.CLIENT_URL}));
app.use("/api", userRouter);
app.use("/", tourRouter);

const start = async () => {
  try {
    await mongoose.connect(process.env.DB_URL);
    app.listen(PORT, () => console.log(`server started on
                port ${PORT}`));
  } catch (e) {
    console.log(e);
  }
};
start();
```

Файл - tourRoutes.js

```
const Router = require("express");
const controller = require("./tourController");
const router = new Router();
router.post("/create", controller.createTour);
router.get("/tour", controller.getTours);
router.delete("/remove/:id", controller.removeTour);
router.post("/api/buy/tour", controller.buyTour);
router.post("/api/buy/validate", controller.validateCardId);
router.post("/api/booking/tour", controller.bookingTour);
router.get("/api/get/booking/tour/:id",
controller.getBookingTour);
router.post("/api/pay/booking/tour/user",
controller.buyTourValidationUser);
router.post("/api/cancel/booking/tour/user",
controller.cancelTour);
router.get("/api/get/ticket/valid/user/:id",
controller.getTicketValidUser);
```

```
router.get("/api/get/tickets", controller.getTickets);
router.get("/api/get/reservations",
controller.getReservations);
router.get("/api/get/users", controller.getUsers);
module.exports = router;
```

Файл - tourController.js

```
const tourService = require("../../service/tour/
tourService");
const decrypted = require("../../crypto/cryptography");
const mailService = require("../../
service/mail/mailService");
const { json } = require("body-parser");

class TourController {
  async getTours(req, res, next) {
    try {
      const arrTours = await tourService.getAllTours();
      res.json(arrTours);
    } catch (e) {
      next();
    }
  }

  async createTour(req, res, next) {
    try {
      const { name, type, date, country, city,
     price, duration, linkPhoto } = req.body;
      const ss = await tourService.createTour(
        name, type, date, country,
   city, price, duration, linkPhoto);
      res.json({ elem: "add db " });
    } catch (e) {
      next();
    }
  }

  async removeTour(req, res, next) {
    try {
       const removedTour = await tourService.de-
lete(req.params.id);
       res.json({ removedTour });
    } catch (e) {
      res.json({ removedTour: "tour do not deleted" });
    }
  }

  async validateCardId(req, res, next) {
    try {
      const { email } = req.body;
      const Id = await tourService.
```

```
      validateIdCard(email.email);
            if (!Id) {
              Id = "error";
            }
res.json({ Code: "код потверждения отправлен на почту" });
           } catch (e) {
            next();
          }
        }

        async buyTour(req, res, next) {
          try {
            const { clientInfoBooking } = req.body;
            const buyTourData = await tourService.buyTour(
              JSON.parse(decrypted(clientInfoBooking)));
            res.json({ elem: buyTourData });
          } catch (e) {
            next();
          }
        }

        async bookingTour(req, res, next) {
          try {
            const { objectBooking } = req.body;
            const bookingTourData = await tourService.
      reservationTour(JSON.parse(decrypted(objectBooking)));
            res.json(bookingTourData);
          } catch (e) {
            next();
          }
        }

        async getBookingTour(req, res, next) {
          try {
            const user = JSON.parse(req.params.id);
            const getBookings = await tourService.
            getReservationTour(user);
             return res.json(getBookings);
          } catch (e) {
            next();
          }
        }

        async buyTourValidationUser(req, res, next) {
          try {
            const { bookingInfoUser } = req.body;
            const payTour = await tourService.buyTourValidUser(
              JSON.parse(decrypted(bookingInfoUser)));
            res.json(payTour);
          } catch (e) {
            res.json(
      { elem: "не получилось купить забронированный тур" });
```

```
    next();
  }
}

async cancelTour(req, res, next) {
  try {
    const { cancelTourObj } = req.body;
    const cancel = await tourService.
    cancelBookingTour(cancelTourObj);
    console.log("cancel ", cancel);
    res.json(cancel);
  } catch (e) {
    res.json({ elem: "не получилось отменить тур" });
    next();
  }
}

async getTicketValidUser(req, res, next) {
  try {
    const ticketInfoUser = JSON.parse(req.params.id);
    const ticket = await tourService.
    getValidUserTicket(ticketInfoUser);
    res.json(ticket);
  } catch (e) {
    res.json({ elem: "не получилось взять билеты" });
    next();
  }
}

async getTickets(req, res, next) {
  try {
    const tickets = await tourService.getTickets();
    res.json(tickets);
  } catch (e) {
    next();
  }
}

async getReservations(req, res, next) {
  try {
    const reservations = await tourService.
    getReservations();
    res.json(reservations);
  } catch (e) {
    next();
  }
}

async getUsers(req, res, next) {
  try {
    const users = await tourService.getUsers();
    res.json(users);
```

```
    } catch (e) {
      next();
    }
  }
}
module.exports = new TourController();
```

Файл – tourService.js

```
const { ObjectId } = require("mongodb");
const TourModel = require("../../models/tour/tourModel");
const { onlineTicket } = require("../pdf/pdfTicket");
const mailService = require("../mail/mailService");
const { nanoid } = require("nanoid");
const CardValidationId = require("../../models/
validCardId/validateCardId");
const BookingModel = require("../../models/booking/
bookingModel");
const UserModel = require("../../models/user/userModel");
const TicketNoUserModel = require("../../mod-
els/ticketNoUser/ticketNoUser");
const TicketModel = require("../../models/ticket/
ticketModel");
const bookingModel = require("../../models/booking/
bookingModel");

class TourService {
  async getAllTours() {
    const tours = await TourModel.find({ flag: true });
    return tours;
  }

  async createTour(name,type, date, country, city, price,
                   duration, linkPhoto) {
    const tour = await TourModel.create({
          name, type, date, country, city, price,
          duration, flag: true, linkPhoto, });
    return tour;
  }

  async delete(id) {
    const removedTour = await TourModel.updateOne(
      { _id: id }, { $set: { flag: false } } );
    console.log("removedTour ", removedTour);
    return removedTour;
  }

  async validateIdCard(email) {
    let ID = nanoid(4);
    const cardValid = await CardValidationId.create({
      validateCard: ID, email: email, });
```

```
    mailService.sendValidateCard(email, ID);
    return cardValid;
  }

  async buyTour(objInfoBuyTour) {
    const { customers, mainClient, tour } = objInfoBuyTour;
    const cardValidation = await CardValidationId.findOne({
      email: mainClient.email,
      validateCard: mainClient.codeId,
    });

    if (!cardValidation) {
      return { elem: "not found code or email" };
    }
    const totalCost = customers.length * tour.price;
    const reys = tour.country[0] + tour.city[0] +
                 tour.duration;
    const time = "03:00";
    const pathName = `C:/Users/admin/
                        Desktop/diplom/server/pdf/
     ${mainClient.firstName}.pdf`;
    let items = [];
    for (let i = 0; i < customers.length; i++) {
      items.push({
        item: customers[i].firstName + " " +
              customers[i].lastName,
        description: customers[i].passportSeriesAndNumber,
        quantity: time,
        amount: reys,
      });
    }

    const invoice = {
      shipping: {
        name: tour.name,
          address: tour.duration + " " + tour.type + " " +
          tour.date,
        city: tour.city, state: tour.country, country: " ",
        postal_code: "94111",
      },
      items, subtotal: "8000", paid: totalCost,
           invoice_nr: totalCost,};

    onlineTicket(invoice, pathName);
    try {
      const createTicketNoUser = await TicketNoUserModel.
create({
        tourId: tour._id,
        firstName: mainClient.firstName,
        lastName: mainClient.lastName,
        email: mainClient.email,
        phoneNumber: mainClient.phoneNumber,
```

```
            cardNumber: mainClient.cardNumber,
            cardId: mainClient.codeId,
            paid: totalCost,
            clock: time,
            flight: reys,
            pathName,
            customers,
            flag: true,
          });
        } catch (error) {
          console.log(error);
        }

    mailService.sendTicket(pathName, mainClient.firstName,main-
Client.email);
    const cardValidationDelete = await CardValidationId.
    deleteOne({ email: mainClient.email, validateCard: mainCli-
ent.codeId, });
        return { elem: "vse super " };
      }

      /** Booking Service */
      async reservationTour(objInfoBuyTour) {
const { customers, mainClient, tour, card } = objInfoBuyTour;

        const tourModel = await TourModel.findOne({
          _id: tour._id,  country: tour.country, });

        if (!tourModel) {return { elem: "нету такого тура" };}
        const userModel = await UserModel.findOne({
          _id: mainClient.id, email: mainClient.email, });

        if (!userModel) {
          return { elem: "нету такого пользователя" };
        }

const cardValidationModel = await CardValidationId.findOne({
         email: mainClient.email, validateCard: card.codeId, });

    if (!cardValidationModel) {
          return { elem: "Неверный код подтверждение" };}
        const cost = tour.price * customers.length;
        const pay = 30 * (cost / 100);
        const needToPay = cost - pay;

        const btm = await BookingModel.create({
          user: userModel._id, tour: tourModel._id,
          pay, needToPay, price: cost, customers,});

        const cardValidationRemove = await CardValidationId
         .deleteOne({email: mainClient.email, validateCard:
    card.codeId, });
```

```
        return { booking: "Тур успешно забронирован" };
      }

      async getReservationTour(user) {
        const userModel = await UserModel.findOne({
          _id: user.id, email: user.email, });
        if (!userModel) {
          return { elem: "Нету такого пользователя" };
        }

        if (user.role === "USER") {
      console.log(userModel);
      const id = userModel._id;
      const getBooking1 = await BookingModel.find({ user: id });
          let arr = [];
          for (let i = 0; i < getBooking1.length; i++) {
          const tourInfo = await TourModel.findOne({
          _id: getBooking1[i].tour});
            const bookingInfo = getBooking1[i];
            arr.push({ bookingInfo, tourInfo,
                        userInfo: userModel });
          }
          return arr;
        }

        const getBooking = await BookingModel.find();
        let arrAdmin = [];
        for (let i = 0; i < getBooking.length; i++) {
          const tourInfo = await TourModel.findOne({
                               _id: getBooking[i].tour });
          const userInfo = await UserModel.findOne({
                         _id: getBooking[i].user });
          const bookingInfo = getBooking[i];
          arrAdmin.push({ bookingInfo, tourInfo, userInfo });
        }
        return arrAdmin;
      }

      async buyTourValidUser(bookingInfoValidUser) {
        const invoice = {
          shipping: {  name: "John Doe",
            address: "1234 Main Street",  city: "San Francisco",
            state: "CA", country: "US", postal_code: 94111,1 },
          items: [
            {
              item: "TC 100", description: "Toner Cartridge",
quantity: 2,
          amount: 6000, },
            {
              item: "USB_EXT", description: "USB Cable Ex-
tender",
              quantity: 1,  amount: 2000,  }, ],
```

```
          subtotal: 8000,  paid: 0, invoice_nr: 1234, };

        const { bookingInfo, tourInfo, userInfo, card } =
     bookingInfoValidUser;
       const tourModel = await TourModel.findOne({ _id: tour-
Info._id });
        if (!tourModel) {
          return { elem: "нету такого тура" };
        }

        const userModel = await UserModel.findOne({
          _id: userInfo._id, email: userInfo.email, });
        if (!userModel) {
          return { elem: "нету такого пользователя" };
        }

        const bookingModelTicket = await BookingModel.findOne({
          _id: bookingInfo._id, });
        if (!bookingModelTicket) {
          return { elem: "нету такого забронированного тура" };
        }

        const cardValidId = await CardValidationId.findOne({
              email: userInfo.email, validateCard: card.codeId,
              });

        if (cardValidId !== null) {
          const cardValidationDelete = await CardValidationId
         .deleteOne({ email: userInfo.email, validateCard:
         card.codeId,});
        } else {
          return {
         elem: "неверный код валидация карты введеите
              то что отправили на почту",
          };    }

        const bookingModelDelete = await BookingModel
         .deleteOne({ _id: bookingInfo._id,});

        const folder = userInfo.nickName + Date.now();
        const reys = tourInfo.country[0] +
        tourInfo.city[0] + tourInfo.duration;
         const directory = `C:/Users/admin/Desk-
         top/diplom/server/pdf/${folder}.pdf`;
        const createTicket = await TicketModel
         .create({
          userId: userInfo._id,
          tourId: tourInfo._id,
          email: userInfo.email,
          cardNumber: card.cardNumber,
          cardId: card.codeId,
          paid: bookingInfo.price,
```

```
      clock: "04:00",
      flight: reys,
      pathName: directory,
      customers: bookingInfo.customers,
      flag: true,
    });

    onlineTicket(invoice, directory);
     mailService.sendTicket(directory, userInfo.nickName,
     userInfo.email);
    return "тур куплен";
  }

  async cancelBookingTour(cancelTourObj) {
    const { userId, tourId, bookingId } = cancelTourObj;
    const bookingModelDelete = await BookingModel.
     deleteOne({
      _id: bookingId,
      user: userId,
      tour: tourId,
    });
    return { elem: "тур успешно отменен" };
  }

  /** Ticket Service */

  async getValidUserTicket(objUserTicket) {
    const tickets = await TicketModel.find({
      userId: objUserTicket.userId,
      email: objUserTicket.email,
    });
    const toursId = tickets.map((elem) => elem.tourId);
const tours = await TourModel.
find({ _id: { $in: toursId } });

    const resultTickets = [];

    for (let i = 0; i < tickets.length; i++) {
      for (let j = 0; j < tours.length; j++) {
     if (JSON.stringify(tickets[i].tourId) ===
     JSON.stringify(tours[j]._id)) {
          tickets[i].tourId = tours[j];
          resultTickets.push(tickets[i]);
          break;
        }
      }
    }
    return resultTickets;
  }

  async getTickets() {
    let arrTicket = [];
```

```
    const ticketValidUser = await TicketModel.find()
      .populate("userId").populate("tourId")
      .then((objArr) => (arrTicket = { ...objArr }))
      .catch((err) => console.log(err));
    let arrNoUserTicket = [];
    const ticketNoUser = await TicketNoUserModel
.find()
.populate("tourId")
.then((objArrNoUser)=>(arrNoUserTicket ={...objArrNoUser }))
.catch((err) => console.log(err));
if(arrTicket.length !== 0 && arrNoUserTicket.length !== 0){
      return { arrTicket, arrNoUserTicket };
    } else {
      return "нету билетов";
    }
  }

  async getReservations() {
    let arr = [];
    const bookings = await BookingModel
      .find()
      .populate("tour").populate("user")
      .then((p) => (arr = { ...p }))
      .catch((error) => console.log(error));
    if (arr.length === 0) {
      return "Нету забронированных туров";
    }
    return arr;
  }
  async getUsers() {
    const users = await UserModel.find();
    return users;
  }
}
module.exports = new TourService();
```