Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет: Инженерно-экономический

Кафедра: Экономической информатики

Дисциплина: Средства и технологии анализа и разработки информационных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ РАБОТЫ ТУРФИРМЫ

БГУИР КП 1-40 01 02-02 06 ПЗ

Выполнил студент группы 372301

Иванова Л. В.

Проверил ассистент кафедры ЭИ

Ярмольчик В. В.

Минск 2016

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc450804277)

[1 Описание работы автоматизированной системы управления работы турфирмы 8](#_Toc450804278)

[1.1 Использование информационных технологий в туристической отрасли 8](#_Toc450804279)

[1.2 Системы бронирования 11](#_Toc450804280)

[1.3 Обзор современных программных и технических средств для решения подобных задач 12](#_Toc450804281)

[2 Описание основных процессов системы работы турфирмы 16](#_Toc450804282)

[3 Спецификация вариантов использования 19](#_Toc450804283)

[4 Информационная модель системы и ее описание 21](#_Toc450804284)

[5 Обоснование выбора компонентов и технологий для реализации курсового проекта 26](#_Toc450804285)

[5.1 Node.js 26](#_Toc450804286)

[5.2 Фреймворк Express.js 28](#_Toc450804287)

[5.3 MongoDB 29](#_Toc450804288)

[5.4 Диаграмма последовательности 31](#_Toc450804289)

[5.5 Диаграмма компонентов 33](#_Toc450804290)

[5.6 Диаграмма развертывания 34](#_Toc450804291)

[6 Модели представления системы и их описание 37](#_Toc450804292)

[6.1 Диаграмма состояний 37](#_Toc450804293)

[6.2 Обобщенный алгоритм программы 38](#_Toc450804294)

[6.3 Алгоритм бронирования туров 39](#_Toc450804295)

[7 Описание применения паттернов проектирования 42](#_Toc450804296)

[7.1 Паттерн проектирования MVC 42](#_Toc450804297)

[7.2 Паттерн проектирования Singleton 45](#_Toc450804298)

[7.3 Паттерн проектирования Observer 47](#_Toc450804299)

[8 Руководство по развертыванию системы 49](#_Toc450804300)

[8.1 Установка необходимых компонент 49](#_Toc450804301)

[8.2 Запуск системы 51](#_Toc450804302)

[9 Результаты тестирования разработанной системы и оценка выполнения задач 52](#_Toc450804303)

[Выводы и заключения 62](#_Toc450804304)

[Список используемой литературы 63](#_Toc450804312)

[Приложение А (обязательное) Функциональная модель 64](#_Toc450804313)

[Приложение Б (обязательное) Листинг основных элементов программы 67](#_Toc450804314)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 75](#_Toc450804315)

**ВВЕДЕНИЕ**

Понятия туризм и информация неразделимы. Это даёт основание утверждать, что каждый сотрудник, работающий в туристской отрасли, должен уметь работать с информацией, включая её сбор, обработку, и на этой основе принимать решения. Значительный объем информации, работающий в сфере туристского бизнеса, затрудняет ход производительного процесса. В связи с этим возникают задачи автоматизации производства, когда возможностей существующих коммуникаций становится недостаточно для эффективной работы. Таким образом, внедрение автоматизированных систем управления, основанных на использовании современных информационных технологий, крайне необходимо. Это позволяет туристской организации расширить диапазон услуг, предоставляемых клиенту, повысить оперативность работы и снизить издержки, более эффективно взаимодействовать с партнёрами, использовать мультимедийные рекламно-информационные продукты, а также получить доступ к обширным информационным ресурсам.

Современные компьютерные технологии активно внедряются в сферу туристского бизнеса, и их применение становится неотъемлемым условием повышения конкурентоспособности любого туристского предприятия. Индустрия туризма позволяет использовать все многообразие компьютерных технологий, начиная от специализированных программных продуктов управления отдельной туристской фирмой до применения глобальных компьютерных систем.

Специализированные автоматизированные системы управления дают возможность работать в режиме, предназначенном для формирования пакетов услуг и программ туров, готовых к реализации, калькуляции их стоимости, расчета прибыли. С помощью автоматизированных систем можно вести базы данных партнеров, что дает возможность организовать с ними персональную работу. С помощью комплексной системы автоматизации можно корректировать стратегию продаж. Использование специализированного программного обеспечение экономит время персонала, упрощает работу по оформлению туристов, обеспечивает печать всех необходимых документов, уменьшает вероятность ошибок в работе персонала.

Цель данного курсового проекта: сократить непроизводительные трудозатраты работников туристических фирм, за счет частичной автоматизации процесса предоставления услуг; привлечение новых клиентов, за счет разработки удобной и дружественной инфраструктуры, с помощью которой клиентам станет приятнее пользоваться услугами.

Для достижения поставленной цели следует выделить основные задачи:

* изучить основные принципы и особенности предоставления туристических услуг клиентам;
* исследовать процессы бронирования и оформления путевок, определить пути их улучшения путем автоматизированной системы
* разработать модель, которая полно описывает функционирование системы, посредством различных наглядных диаграмм;
* определить методы и технологии разработки программного продукта;
* разработка приложения в соответствии с ранее составленной моделью;
* тестирование разработанной системы в реальной среде, анализ полученных данных и оптимизация;

# **1 ОПИСАНИЕ РАБОТЫ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ РАБОТЫ ТУРФИРМЫ**

## **1.1 Использование информационных технологий в туристической отрасли**

Международный и внутренний туризм представляет собой мощную отрасль по торговле услугами.

Основу туристической индустрии составляют фирмы туроператоры и турагенты, занимающиеся туристическими поездками, продажей их в виде путевок и туров; предоставляющие услуги по размещению и питанию туристов (гостиницы, кемпинги и др.), их передвижению по стране, а также органы управления, информации, рекламы по исследованию туризма и подготовке для него кадров, предприятия по производству и продаже товаров туристского спроса. На туризм работают и другие отрасли, для которых обслуживание туристов не является основным видом деятельности (предприятия культуры, торговли и др.).

Туризм – информационно насыщенная деятельность. Существует немного других отраслей, в которых сбор, обработка, применение и передача информации были бы настолько же важны для ежедневного функционирования, как в туристической индустрии. Услуга в туризме не может быть выставлена и рассмотрена в пункте продажи, как потребительские или производственные товары. Ее обычно покупают заранее и вдали от места потребления. Таким образом, туризм на рынке почти полностью зависит от изображений, описаний, средств коммуникаций и передачи информации.

Турагент – физическое или юридическое лицо, выступающее посредником по продаже сформированных туроператором туров. Туроператор – туристическая организация, занимающаяся комплектацией туров.

Система информационных технологий, используемых в туризме, состоит из компьютерной системы резервирования, системы проведения телеконференций, видеосистем, компьютеров, информационных систем управления, электронных информационных систем авиалиний электронной пересылки денег, телефонных сетей, подвижных средств сообщения и т.д. При этом необходимо отметить, что эта система технологий развертывается не турагентами, гостиницами или авиакомпаниями каждым в отдельности, а всеми ими. Более того, использование каждым сегментом туризма системы информационных технологий имеет значение для всех остальных частей. Например, системы внутреннего управления гостиницей могут быть связаны с компьютерными глобальными сетями, которые обеспечивают, в свою очередь, основу для связи с гостиничными системами резервирования, которые, уже в обратном направлении, могут быть доступны турфирмам через их компьютеры.

Следовательно, мы имеем дело с интегрированной системой информационных технологий, которая распространяется в туризме.

Кроме того, отдельные компоненты туристической отрасли тесно взаимосвязаны друг с другом — ведь многие турпроизводители вертикально или горизонтально вовлечены в деятельность друг друга. Все это позволяет рассматривать туризм как высокоинтегрированную услугу, что делает его еще более восприимчивым для применения информационных технологий в организации и управлении.

Не секрет, что, когда клиент туристического агентства, приобретая тур, получает хорошо подготовленные документы и полную информацию о туре, он более комфортно себя чувствует и больше доверяет выбранному агентству. Также и турфирма, затратив немалые средства на привлечение данного клиента к себе, сохраняя полную информацию о нем, может в дальнейшем использовать ее для повторного привлечения старого клиента к себе. Известно, что затраты в этом случае в четыре раза ниже первоначальных.

Есть возможность сразу поставить единую систему учета, как оперативного, так и финансового, причем в двух разрезах - управленческого и бухгалтерского. С использованием гибкого разделения прав доступа и видимости информации менеджерам доступна только их часть системы, а бухгалтерам - их часть. Естественно, руководители компании видят всю информацию и могут наблюдать за работой менеджеров со своего компьютера. Так как оперативный ввод информации ведется в единой системе, то видно, кто и как работает: количество введенных заявок, подтвержденных, оплаченных клиентом, оплаченных поставщику, закрытых, доход по каждой заявке. Через отчеты системы руководитель видит эффективность работы каждого менеджера и подразделения. Детализацию можно установить по каждому туристу и каждой составляющей тура. По любому клиенту можно увидеть историю его обращений и оплат.

В единой системе, менеджеры могут тут же увидеть, оплачена их заявка или нет, сумму оплаты. Бухгалтерия сразу же получает исходную информацию и не тратит время на повторный ввод одних и тех же данных. Вся информация разделена по правилам доступа, и лишнюю информацию менеджер не увидит. Соответственно, нет необходимости в разделении баз данных, что неминуемо приводит к ошибкам ввода, неоперативности работы и вытекающих из этого проблем.

После того как решены основные задачи систематизированного управления турфирмой – управление работой с клиентами, с поставщиками и внутренней деятельности компании, можно приступать к расширению системы.

В конечном итоге вся необходимая информация в красивом виде выводится на печать. Клиентам будет выдаваться весь комплект необходимых документов, поставщикам – соответствующие отчеты, а для государства – печататься все отчеты установленной формы.

Автоматизация работы турагентств обычно сводится к оперативному учету заказанных туристами туров, оформленных платежей с клиентами и партнерами, печати всех необходимых документов.

Для автоматизации отдельного туристического агентства необходимо использовать программное обеспечение, основанное на применении базы данных, аккумулирующей все рабочие данные в едином хранилище. Это необходимо для получения отчетов о работе агентства в любой момент времени. Кроме базы данных необходимо также иметь программу с удобным пользовательским интерфейсом, приемлемой скоростью работы, конфигурированием параметров системы, относящихся к деятельности фирмы. К таким параметрам в первую очередь относятся текст и внешний вид печатных документов, формируемых на базе рабочих данных, а также возможность управления правами доступа к различным данным системы.

С автоматизацией сетевых агентств дело обстоит несколько сложнее. Сетевые агентства с юридической точки зрения могут иметь различные типы: одна фирма с несколькими офисами продаж, сеть отдельных агентств под общим логотипом (франчайзи): Руководство сетевых агентств периодически нуждается в оперативной отчетности о деятельности всей сети. Предоставить такую отчетность можно только при условии хранения всех данных в едином хранилище[1].

Техническая сторона автоматизации сетевых агентств должна в определенном смысле переплетаться с видом сети. Если отдельное агентство достаточно легко автоматизировать, подключив его к серверу единой базы данных центрального офиса через каналы интернета, то с франчайзи так поступить невозможно. Проблемы связи с сервером базы данных сразу же приведут к параличу работы отдельного агентства. В случае с франчайзи это нарушение работы одной компании из-за проблем другой компании. Решение возможно только с помощью применения локальной рабочей базы данных в отдельном офисе, что исключает зависимость работы агентства от работы канала передачи данных, с копированием в центральный офис всех введенных данных (канал передачи данных используется достаточно редко по сравнению с постоянным подключением – раз в час, раз в день, периодичность выбирается головной компанией). Но за преимуществами кроются и недостатки. Сервер центрального офиса хранит у себя единую базу данных, собранную от отдельных агентств. Центральный офис может осуществлять только пассивное наблюдение за работой агентства. Технически очень сложно реализовать активное поведение центрального офиса по отношению к дочернему.

Автоматизацию точек продаж одной компании гораздо лучше проводить с использованием центральной базы данных и каналов постоянной связи с сервером. Такое построение сети позволяет гораздо эффективнее управлять данными (заявками, платежами, правами доступа и т.д.), а отчетность о работе (если на это есть необходимые права доступа) получать в любом офисе.

## **1.2 Системы бронирования**

Бронирование – это предварительный заказ на предстоящее туристское обслуживание, предварительное закрепление объектов бронирования: мест (билетов, путевок), снаряжения, оборудования и т.д. за заказчиком [2]. Бронирование осуществляется различными информационными системами, которые резервируют, т.е. создают запасы объектов бронирования, а затем из созданного резерва закрепляют их за потребителем.

В настоящее время в сфере реализаций туристских услуг активно используется компьютерная техника. Сейчас уже трудно представить, как могло производиться бронирование отелей, железнодорожных или авиабилетов различными тур агентами и компаниями с учетом всевозможных скидок при отсутствии систем компьютерного бронирования.

Высокая экономическая эффективность использования систем компьютерного бронирования побудила их владельцев бороться за влияние на рынке туристских услуг. Для того чтобы привлечь тур агентов, компании расширили сферу применения систем бронирования, включив в их программу широкий круг услуг (бронирование гостиничных мест, прокат автомобилей, страхование авиапассажиров, оформление билетов, заграничных паспортов и даже заказов на цветы). Кроме того, программы предоставляют тур агентам возможность компьютерной обработки документации и бухгалтерского учета.

Функции агентства при самостоятельном бронировании клиентами через Интернет заключаются в проверке корректности выполненных броней, оптимизации выбранных маршрутов, выписке и доставке авиабилетов.

Итак, каждая компьютерная система бронирования создаёт свои собственные компьютерные программы, которые выполняют следующие функции:

* позволяют проводить поиск оптимальных ценовых вариантов туруслуги;
* позволяют работать по различным тарифам, в том числе, по конфиденциальным;
* позволяют турагенству автоматически распечатывать на матричном принтере билеты, предназначенные для ручной выписки, используя данные о бронировании;
* позволяют турагентствам использовать программу автоматического составления отчетов;
* позволяют вести различные виды статистики; предоставляют доступ к обширной справочной системе оперативной информации;
* позволяют контролировать все операции по взаиморасчетам с клиентами и поставщиками;
* позволяют создавать на сайте турагентства систему бронирования туруслуг для корпоративных и частных клиентов. Таким образом, посетитель сайта получает возможность самостоятельного выбора авиарейсов и отелей и их последующего бронирования [3].

Взаимодействие туроператора с компьютерными системами бронирования создает возможность:

* интегрироваться в мировой туристский рынок;
* оперативно управлять своей деятельностью;
* расширять рынок сбыта туристской продукции;
* планировать маркетинговую деятельность;
* предварительно формировать списки потребителей и заблаговременно обеспечивать загрузку;
* анализировать спрос на туристскую продукцию и многое другое.

## **1.3 Обзор современных программных и технических средств для решения подобных задач**

Выбор турфирмами программных продуктов и специфика их использования зависят от ряда факторов, в частности:

* от направления деятельности турфирмы, совокупности решаемых задач, исходной технологии, принятой на турфирме, суммарного объема продаж, финансового состояния турфирмы;
* от осведомленности руководства турфирмы о программном обеспечении, существующем на рынке информационных технологии управления в туризме, о его достоинствах и недостатках;
* от режима работы компьютеров (автономный или сетевой).

Существующие в настоящее время программы автоматизации работы туристских фирм можно условно разбить на три класса:

1  Программы автоматизации работ, которые обеспечивают внутреннюю деятельность турфирмы, без автоматизации поддержания внешних взаимосвязей. Предполагается создание локальной вычислительной сети, баз данных, которые располагаются на центральной мощной машине-сервере, куда обеспечивается сетевой доступ пользователей с их автоматизированных рабочих мест.

2  Программы автоматизации включают возможности программ 1-го класса, а также позволяют туроператору осуществлять модемную связь в пакетном режиме с другими турагентствами.

3  Программы автоматизации, базирующиеся на использовании глобальных телекоммуникационных сетей. В этом случае базы данных о турпродукте той или иной турфирмы размещаются в информационных узлах глобальной сети и для всех пользователей сети обеспечивают удаленный доступ в режиме online [4].

На диаграмме (рисунок 1.1) представлены ответы туристических агенств на вопрос: «Какие системы автоматизации турагенства вы знаете?».

По диаграмме можно сказать, что наиболее популярной системой является «САМО-турагент». Решение выбрать данное ПО является абсолютно обоснованным, так при анализе ряда подобных программ, «САМО-турагент» практически по всем рассматриваемым критериям получил положительный отзыв. Результаты анализа представлены на рисунках 1.2 и 1.3 [5].

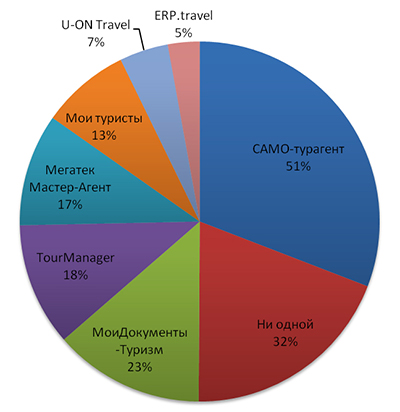
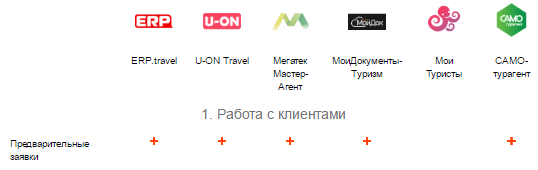


Рисунок 1.1 – Системы автоматизации, используемые турфирмами





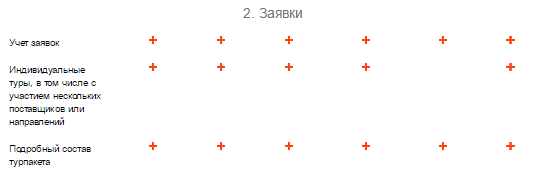


Рисунок 1.2 – Анализ систем по критериям









Рисунок 1.3 – Анализ систем по критериям (продолжение)

# **2 ОПИСАНИЕ ОСНОВНЫХ ПРОЦЕССОВ СИСТЕМЫ РАБОТЫ ТУРФИРМЫ**

Как, правило, первым этапом изучения любой системы является моделирование средствами IDEF0. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков - в терминах IDEF0)[6].

В проекте основным бизнес-процессом является предоставление туристических услуг клиентам. При разработке данного процесса было использовано CASE-средство ProcessModeler. На основе модели IDEF0 была построена функциональная модель данных «Предоставление туристической услуги клиенту». Общий вид процесса представлен на рисунке 2.1. На контекстной диаграмме верхнего уровня (Приложение А, рисунок А.1) представлена функциональная модель «Предоставления туристической услуги клиентам», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными. Как видно из диаграммы, на входе процесса — информация о туристических местах в мире, статистика продаж и желание клиентов воспользоваться турагентством. Механизмы исполнения — автоматизированная система, оператор, клиент и представители туристических мест в мире. Управление — методы и правила статистического анализа, курсы валют, регламенты по туристическим мероприятиям, образцы для заполнения, а также правила по организации работы. На выходе процесса — клиент с оформленной путевкой.

После описания контекстной диаграммы проводится функциональная декомпозиция — система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции).

На диаграмме декомпозиции первого уровня (Приложение А, рисунок А.2) выделено 3 основных процесса, а именно: сформировать тур, после которого на выходе появляется сформированные туры и начинается процесс предоставления клиенту забронировать тур. После чего на выходе получается либо желание клиента сразу оформить путевку, либо забронировать тур. В следствие чего начинается новы процесс предоставления клиенту путевки.

Процесс формирования туров достаточно нетривиален, поэтому составлена декомпозиция второго уровня (Приложение А, рисунок А.3). на данном уровне параллельно происходят два процесса: выявление туристических мест в мире под управлением регламентов по туристическим мероприятиям. В итоге процесса получается информация о доступных местах для туристов. Второй параллельный процесс – анализ продаж предыдущих периодов. Данный процесс получает статистику продаж и при помощи автоматизированной системы и оператора выявляется информация о продажах различных секторов туризма по периодам.



Рисунок 2.1 – Уровни функциональной модели бизнес-процесса

Два полученных результата нужны для дальнейшего выявления наиболее актуальных туров. Для этого следует учитывать условия актуальности в различные сезоны и периоды. В результате по полученным данным может составляться список предоставляемых услуг, который будет наиболее актуальным и популярным среди клиентов. Однако для составления туров необходимы правила оформления туров в организации, текущие курсы валют и образцы для заполнения.

Когда туристические услуги сформированы, можно предоставить клиенту забронировать тур – декомпозиция второго уровня (Приложение А, рисунок А.4). первым этапом в данном процессе выступает ознакомление клиентов со списком предоставляемых туров, для это нужны сформированные туры от предыдущего процесса и желание клиента воспользоваться услугами турагентства. Процесс происходит по правилам обслуживания клиентов.

При правильном выполнении процесса ознакомления клиента у последнего должно возникнуть желание подобрать подходящий тур именно в данном турагентстве. После подбора тура по предпочтениям клиента может возникнуть два варианта результата. Первый: клиент сразу желает оформить путевку по соответствующим документам. Второй: клиент желает сначала забронировать тур. Во втором случае оператор предоставляет клиенту форму для заполнение контактных данных и полученные результаты отправляет на оформление брони. На этом этапе процесс предоставления клиенту забронировать тур завершается.

После оформления брони можно начинать процесс предоставления клиенту путевки. Данный процесс является составным, поэтому на рисунке А.5 Приложения А изображена декомпозиция процесса.

Первый этап в данном процессе – это получения у клиента подтверждения брони, что он не передумал и не решил отказаться от бронирования тура. Если все прошло успешно, то оформляется путевка. Данный документ передается клиенту, а затем сотрудник турфирмы по мере надобности консультирует клиента по приобретенному туру.

В результате, как и планировалось, - клиент с оформленной путевкой.

В конечно процессе очень вскользь упоминается процесс оформления путевки, хотя его возможно декомпозировать на еще один уровень (Приложение А, рисунок А.6). в самом начале стоит собрать у клиента все необходимые данные и документы, необходимые при оформлении билетов/путевки. По полученным документам, относящимся к оформлению билетов, оператор выкупает все необходимые по туру билеты (например, такие как на транспортировку или включенные билеты на экскурсии). Выкупленные билеты и документы, не относящиеся к оплате (например, страховка) формируются в конечную путевку.

# **3 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ**

Диаграмма вариантов использования обеспечивает высокоуровневое описание того, что система в состоянии сделать и с кем (или чем) она будет взаимодействовать. Это называется методом определения функциональных требований.

С помощью схемы вариантов использования можно обсудить и передать следующие сведения:

* сценарии, в которых система или приложение взаимодействуют с людьми, организациями или внешними системами;
* цели, которых она помогает добиться этим субъектам;
* область действия системы.

На схеме вариантов использования не отображаются подробности о вариантах использования: на ней представлены только некоторые отношения между вариантами использования, субъектами и системами. В частности, схема не показывает порядок выполнения шагов для достижения цели каждого из вариантов использования. Эти сведения можно описать в других схемах и документах, которые можно связать с каждым вариантом использования.

Диаграмма вариантов использования автоматизированной системы управления работы турфирмы представлена на рисунке 3.1.



Рисунок 3.1 – Диаграмма вариантов использования автоматизированной системы работы турфирмы

Как видно из диаграммы, актерами являются клиент и оператор. Клиент – непосредственный пользователь системы, а оператор – представитель турфирмы, выполняющий ряд операций, необходимых при предоставлении туристической услуги клиенту.

Каждый клиент имеет возможность найти интересующий его тур, воспользовавшись поиском. Чаще всего, клиенты просто ищут наиболее популярные туры, или просто по различным критериям, не относящимся к конкретике тура. Здесь важным моментом, со стороны турагентства выступает правильное описание каждого тура, чтобы пользователь беспрепятственно вводил свои предпочтения в строку поиска, а система отображала интересующие его результаты.

Бывают моменты, когда клиент почти определенно знает, что хочет найти, правда в абстрактных понятиях. В этом случае, целесообразным будет воспользоваться фильтрации. Данная операция предоставляет клиенту ряд критериев, заполнив которые, клиент имеет возможность отсеять не интересующие его туры (например, вывести все возможные туры по определенной стране). Такой функционал достаточно удобен и сохраняет клиентам нервы при поиске подходящего тура.

Бронирование тура – одна из самых важных операций в процедуре предоставления туристических услуг. Клиенты не всегда имеют возможность моментальной оплаты услуги или не в состоянии в данный момент времени быть в турфирме. операция, которая выполняется, когда клиент определился с туром и хочет забронировать места.

Оформление брони – фактическое продолжение операции бронирования, но уже со стороны оператора. Оформление забронированных туров происходит после оставления клиентом заявления на бронь. В большинстве случаев заявка на бронь поступает к турфирме онлайн, тогда оператор связывается с клиентом по оставленным контактным данным в заявлении. В таком случает процесс оформления несколько дольше, нежели при личной встрече с клиентом.

Важной частью предоставления туристических услуг является оформление документов. Здесь важно определить какие именно документы необходимы, чтобы клиент беспрепятственно прибыл по месту назначения и воспользовался всеми мероприятиями, включенными в тур. В большинстве случаев в каждой турфирме уже существует ряд готовых шаблонов для оформления документов, поэтому оператору необходимо только их грамотно заполнить, поставить подписи и официальные печати

Ни одна туристическая фирма не может обойтись без статистического анализа как своих продаж, так и конкурентов. А также, без регулярного мониторинга популярных туристических мест. Так как это позволяет агентству быть в «топе» среди конкурентов, и продолжать обслуживать как постоянных клиентов, так и впервые обратившихся.

# **4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ**

Ни одну область деятельности человека, поддерживаемую информационными технологиями, невозможно представить себе без использования баз данных, помогающих получить быстрый доступ к информации, увеличивая тем самым продуктивность работы. Клиент-серверные приложения, получившие в последнее время широкое распространение, построены на основе баз данных; приложения Internet и intranet могут получать доступ к базам данных, открывая широкие возможности для публикации информации, необходимой широкому кругу пользователей.

Информационный объект – это описание некоторой сущности предметной области, т.е. реального объекта, процесса, явления или события. База данных (БД) – это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

Традиционно процедуру проектирования базы данных (далее БД) разбивают на три этапа, каждый из которых завершается созданием соответствующей информационной модели: концептуальное, логическое и физическое проектирование.

В настоящее время для проектирования БД активно используются CASE-средства, в основном ориентированные на использование ERD (Entity – Relationship Diagrams, диаграммы «сущность–связь»). С их помощью определяются важные для предметной области объекты (сущности), отношения друг с другом (связи) и их свойства (атрибуты).

Сущность (таблица, в реляционной базе данных (далее РБД) – отношение) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению. Если выражаться точнее, то это не объект, а набор объектов (класс) с одинаковыми свойствами. Примеры сущностей: работник, деталь, ведомость, результаты сдачи экзамена и т. д.

Экземпляр сущности (запись, строка, в РБД – кортеж) – уникально идентифицируемый объект.

Связь – некоторая ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Примерами связей могут являться родственные отношения «отец–сын», производственные – «начальник-подчиненный» или произвольные – «иметь в собственности», «обладать свойством».

Атрибут (столбец, поле) – свойство сущности или связи [7].

В приложении А представлена полная информационная модель разрабатываемо автоматизированной системы. Она состоит из 6-и сущностей.

Сущность users (рисунок 4.1) хранит в себе информацию о личных данных операторов, которые нужны для входа в систему. Она включает в себя следующие атрибуты:

* id. Идентификационный номер оператора, является уникальным первичным ключем. Тип поля: ObjectID. Данный тип позволяет автоматически генерировать ключи, которые представлены 24 байтным шестнадцатиричным числом, приведенном к строке, что гарантирует стопроцентную уникальность;
* login. Уникальный логин пользователя типа String, является обязательным;
* password. Поле для хранения пароля пользователя типа String, является обязательным;
* firstname. Имя оператора, представлено типом String;
* lastname. Фамилия оператора, представлено типом String;

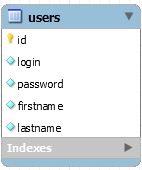


Рисунок 4.1 – Сущность users

Сущность addresses (рисунок 4.2) хранит в себе адреса всех туристических мест, предоставляемых турфирмой. Имеет следующие атрибуты:

* id\_addresses. Обязательный уникальный первичный ключ типа ObjectID;
* country. Обязательное поле типа String, хранящее в себе наименования стран;
* city. Обязательное поле типа String для наименований городов;
* postcode. Индекс места, представленный типом Number.

Сущность hotels (рисунок 4.3) предназначена для хранения информации о различных отелях и гостиницах. Состоит из следующих атрибутов:

* id\_hotel. Поле, хранящее в себе уникальный ключ. Является обязательным. Как и предыдущие id-поля, имеет тип ObjectID;



Рисунок 4.2 – Сущность addresses

* stars. Информацией в этом поле является количество звезд отеля. Обязательное поле типа Number;
* cold\_table. Поле типа Boolean, которое определяет, включен ли шведский стол в данном отеле или нет. По умолчанию – true.
* image. Поле со ссылкой на изображение отеля. Имеет тип String;
* id\_addresses. Поле определяет адрес отеля и служит для связи с сущностью addresses.



Рисунок 4.3 – Сущность hotels

Сущность tours (рисунок 4.4) обобщает всю информацию о туре. Именно с ней осуществляется основная работа по поиску нужных туров. Состоит из следующих полей:

* id\_tour. Обязательный уникальный номер тура, является типом ObjectID;
* id\_hotel. Поле для связи отеля с туром;
* transport. Информация в данном поле описывает способ транспортировки в место тура из данной сраны. Имеет тип String;
* name\_tour. Название тура. Обязательное поле типа String;
* price. Поля типа Object, что означает, что данные представляют собой объект с полями: usd, eur, rub. Показывают стоимость тура на одного человека в разных валютах;

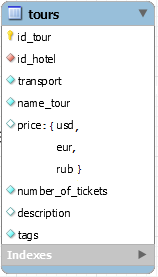


Рисунок 4.4 – Сущность tours

* number\_of\_tickets. В данном поле содержится информация об оставшемся количестве билетов на данный тур. Поле типа Number;
* description. Поле типа String, которое хранит в себе описание тура, с помощью которого турфирма может привлечь клиента воспользоваться данным туром;
* tags. Поле, содержащее теги, предназначенные для оптимального поиска, типа String.

Сущность reservations (рисунок 4.5) служит для хранении информации о поступивших заявление на бронирование туров. Состоит из следующих полей:

* id\_reservation. Обязательный уникальный номер заявления, является типом ObjectID;
* tour\_id. Здесь хранится идентификационный номер тура (сущность tours), для определения, какой тур оформляет клиент;
* fullname. ФИО клиента, записанное в формате String;
* email. Обязательное поле типа String, хранящее электронную почту клиента для связи;
* phone. Обязательное поле, содержащее контактные телефоны клиентов. Как и email, типа String;
* tickets. Информация в данном поле записывается как тип Number. Описывает, сколько билетов заказывает клиент на определенный тур;
* online\_status. Поле содержит информацию типа Boolean. Определяет, каким образом совершен заказ: true – клиент сам, с помощью сети Интернет, забронировал тур; false – клиент обратился в турагенство, где ему оформили заявление на бронь;
* execution\_status. Как и предыдущее поле, является типом Boolean. Показывает в каком состоянии находится заявление, а именно: true – заявление оформлено; false – заявление находится на рассмотрении.

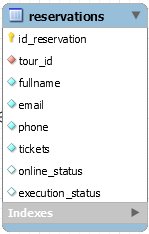


Рисунок 4.5 – Сущность reservations

Сущность archive (рисунок 4.6) является архивом заявлений. С помощью нее составляются статистический анализ по продажам и популярным турам. Состоит из следующих полей:

* id\_ tour. Обязательный уникальный номер тура в архиве, является типом ObjectID;
* tickets. Поле типа Number, хранящее количество оформленных путевок на данный тур. В дальнейшем служит для определения популярности тура.



Рисунок 4.6 – Сущность archive

# **5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ ДЛЯ РЕАЛИЗАЦИИ КУРСОВОГО ПРОЕКТА**

## **5.1 Node.js**

В условиях почти экспоненциального развития технологических инноваций многие новые идеи имеют практический смысл с момента своего появления. Одна из таких идей — JavaScript на стороне сервера. Воплощением этой полезной идеи является Node.js — управляемая событиями инфраструктура ввода/вывода для JavaScript-механизма V8, предназначенная для написания масштабируемых сетевых программ, таких как веб-серверы.

Вместо того чтобы преодолевать трудности с языком JavaScript, каркас Node.js использует его в качестве полного стека средств разработки — от кода на стороне сервера до браузера. Node.js воплощает еще одну инновационную идею: параллельная модель с асинхронным вводом/выводом, реализованным посредством обратных вызовов.

Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и Mac OS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

По сравнению с традиционными вариантами веб-сервисов (рисунок 5.1.1), где каждое соединение (запрос) порождает новый поток, нагружая оперативную память системы и, в конце концов, разбирая эту память без остатка, Node.js гораздо экономичнее: он работает в единственном потоке, при вызовах использует неблокирующий ввод/вывод, позволяет поддерживать десятки тысяч конкурентных соединений (которые существуют в цикле событий).

Несмотря на то, что JavaScript не поддерживает модули на уровне языка, сообществом были созданы впечатляющие решения для их реализации [8]. Два наиболее популярных (но, к сожалению, несовместимых) стандарта:

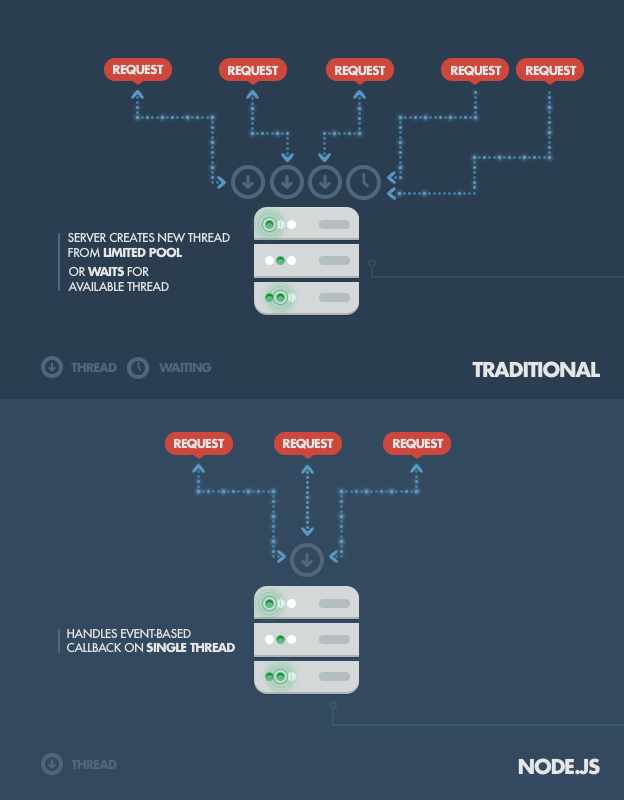


Рисунок 5.1.1 – Сравнение работы Node.js с тридиционными веб-сервисами

1  CommonJS (CJS): главное воплощение этого стандарта — модульная система Node.js. Характеристики:

* компактный синтаксис;
* предназначен для синхронной загрузки;
* преимущественно используется на стороне сервера.

2  Asynchronous Module Definition (AMD): наиболее популярной реализацией этого стандарта стал RequireJS. Характеристики:

* синтаксис немного сложнее, что позволяет AMD работать без eval() или этапа компиляции;
* предназначен для асинхронной загрузки;
* преимущественно используется на стороне клиента.

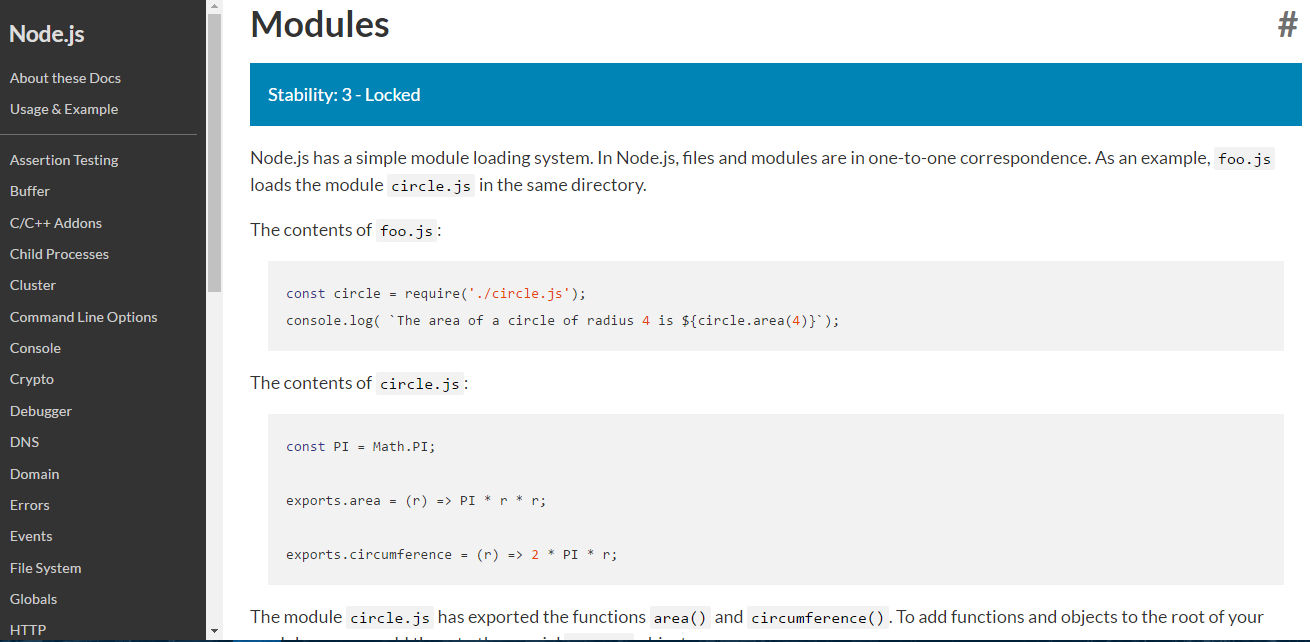


Рисунок 5.1.2 – Страница документации по модулям в Node.js

## **5.2 Фреймворк Express.js**

Express.js – это серверный веб-фреймворк для приложений Node.js, предназначенный для построения одностраничных, многостраничных и гибридных веб-приложения. [9]. Автором express является TJ Holowaychuk. Вся информация об особенностях фреймворка, новых версиях и документация находится на официальном сайте (рисунок 5.2.1).

Ниже приведены некоторые основные особенности структуры Express:

* позволяет настроить промежуточные обработчики для ответа на HTTP-запросы;
* определяет таблицу маршрутизации, которая используется для выполнения различных действий на основе метода HTTP и URL;
* позволяет динамически перерисовывать HTML-страницы на основе передачи аргументов в шаблоны.

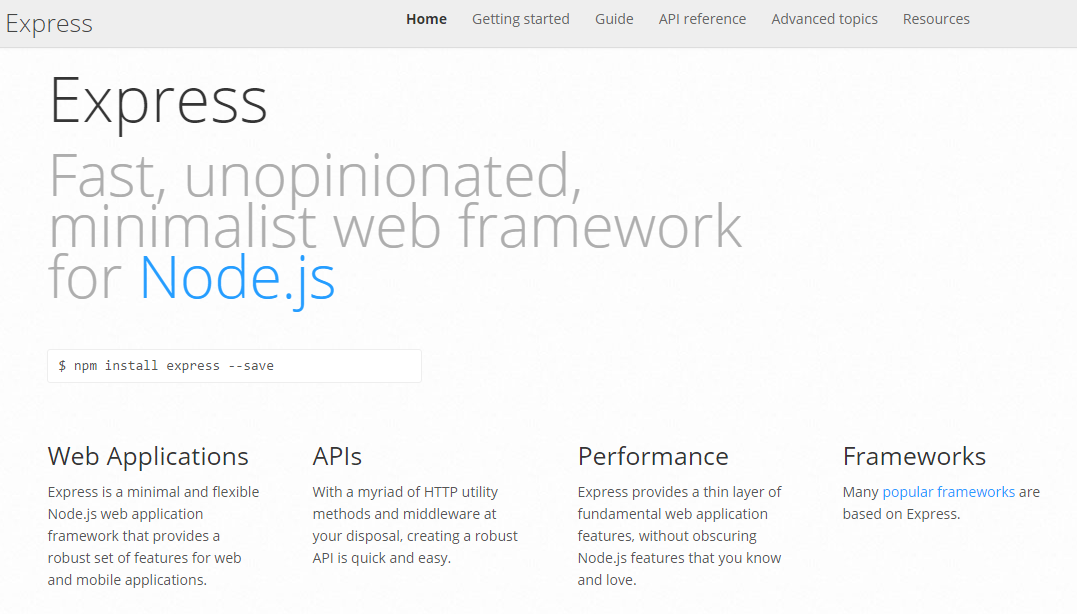


Рисунок 5.2.1 – Домашняя страница фреймфорка Express.js

## **5.3 MongoDB**

MongoDB — документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++.

Основные концепции выбранной БД:

1  MongoDB — концептуально то же самое, что обычная база данных (или в терминологии Oracle — схема). Внутри MongoDB может быть ноль или более баз данных, каждая из которых является контейнером для прочих сущностей.

2  База данных может иметь ноль или более «коллекций». Коллекция настолько похожа на традиционную «таблицу», что можно смело считать их одним и тем же.

3  Коллекции состоят из нуля или более «документов». Документ можно рассматривать как «строку».

4  Документ состоит из одного или более «полей», которые подобны «колонкам».

5  «Индексы» в MongoDB почти идентичны таковым в реляционных базах данных.

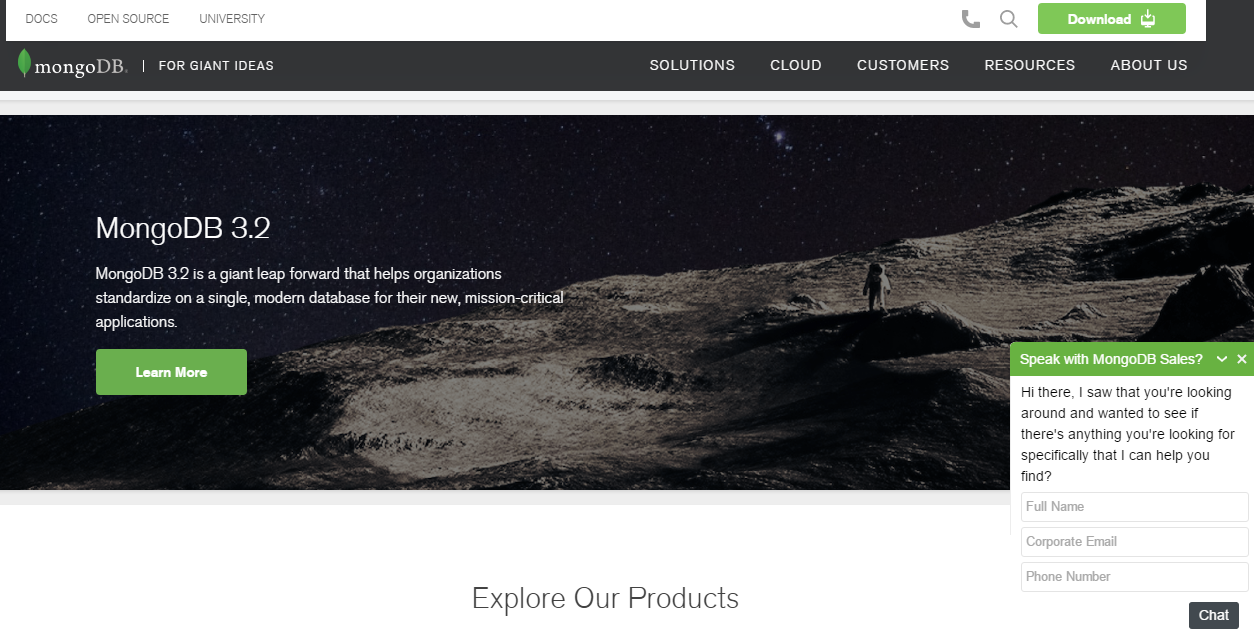


Рисунок 5.3.1 – Главная страница MongoDB

6  «Курсоры» отличаются от предыдущих пяти концепций, но они очень важны. Когда происходит запрос на какие-либо данные у MongoDB, то она возвращает курсор, с которым можно делать все что угодно — подсчитывать, пропускать определённое число предшествующих записей — при этом не загружая сами данные.

Подводя итог, MongoDB состоит из «баз данных», которые состоят из «коллекций». «Коллекции» состоят из «документов». Каждый «документ» состоит из «полей». «Коллекции» могут быть проиндексированы, что улучшает производительность выборки и сортировки. И наконец, получение данных из MongoDB сводится к получению «курсора», который отдаёт эти данные по мере надобности [10].

На рисунке 5.3.2 представлено сравнение реляционной модели и документо-ориентированной. Как видно из рисунка, хранение данных в виде документов, которые хранятся в коллекциях намного проще, понятнее и удобнее.

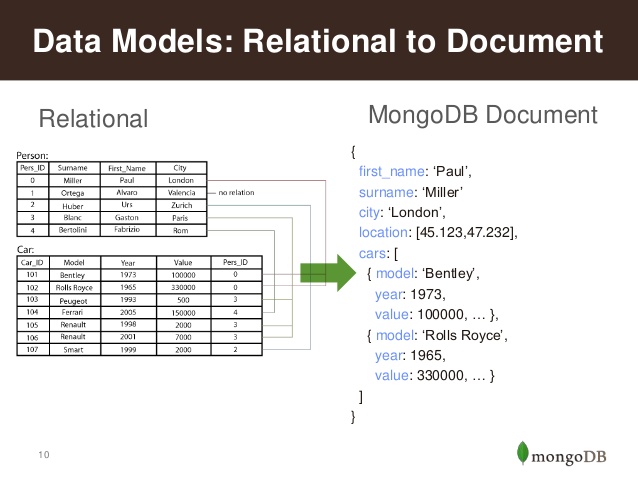


Рисунок 5.3.2 – Представление данных в MongoDB в разрезе РБД

## **5.4 Диаграмма последовательности**

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности. Взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений. Другими словами, хотя сообщение и имеет информационное содержание, оно приобретает дополнительное свойство оказывать направленное влияние на своего получателя [11].

На рисунке 5.4.1 изображена диаграмма последовательности оформления тура, при соответствующем обращении клиента в турфирму. Оператор, получив запрос на оформление тура, в первую очередь, заходит в свой личный кабинет посредством ввода логина и пароля. Сервер системы отправляет запрос на проверку данных в базе данных, и, если все верно, оператору открывается окно поиска тура.



Рисунок 5.4.1 – Диаграмма последовательности

На странице поиска оператор вводит предпочтения клиента, после чего сервер запрашивает у базы данных наиболее подходящие туры в соответствии с полученными предпочтениями клиента. Из полученного списка туров оператор предлагает клиенту определиться с конкретным туром, после введения которого осуществляется переход на страницу бронирования.

На странице бронирования тура оператору предлагается заполнить личные данные клиента, которые будут хранится в базе данных для дальнейшего оформления и статистики продаж.

После оформления бронирования оператор может оформить документы для окончательного составления тура. Для этого в окне оформления документов сначала автоматически генерируются все нужные шаблоны документов, которые оператор заполняет и сохраняет в базе данных.

Сохраненные документы следует распечатать. Далее оператор подписывает все необходимое и передает документы клиенту.

## **5.5 Диаграмма компонентов**

Полный проект программной системы представляет собой совокупность моделей логического и физического уровней, которые должны быть согласованы между собой. В языке UML для физического представления моделей систем используются диаграммы реализации (implementation diagrams), которые включают в себя диаграмму компонентов и диаграмму развертывания.

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

* визуализации общей структуры исходного кода программной системы;
* спецификации исполняемого варианта программной системы;
* обеспечения многократного использования отдельных фрагментов программного кода;
* представления концептуальной и физической схем баз данных.

На рисунке 5.5.1 изображена диаграмма компонентов автоматизированной системы турфирмы. В системе существует 6 взаимосвязанных компонент.

Через клиентское приложение пользователи, а именно: оператор и клиент, имеют возможность оставить заявления на бронирование тура, а также благодаря интерфейсу программы пользователи могут воспользоваться не основными операциями турфирмы, которые предложены в меню программы.

Клиентское приложение само по себе абсолютно бесполезно без определенной связки с сервером. Сервер от клиентской части принимаю различные запросы, обрабатывает и, если нужно, перенаправляет.

Основными операциями, запрашиваемыми на сервере, являются операции по работе с БД.

База данных выступает в качестве отдельного компонента, так как БД имеет свой сервер, который, в свою очередь, обрабатывает запросы самой БД.



Рисунок 5.5.1 – Диаграмма компонентов системы

Вся программная часть системы не имеет смысла без человеческого вмешательства. Клиент либо сам регистрирует заявление на бронь посредством сети Интернет, либо обращается в турфирму, в данном случае – к оператору, с целью помочь найти тур.

Компонент услуги возникает в результате оформления всех необходимых документов оператором, которые генерируются автоматизированной системой. Клиент, в данном случае, оплачивает услугу в соответствии с документами.

## **5.6 Диаграмма развертывания**

Диаграмма развертывания (deployment diagram) - диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений - маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются [11].

На рисунке 5.6.1 изображена диаграмма развертывания разрабатываемого проекта.



Рисунок 5.6.1 – Диаграмма развертывания системы

Данная диаграмма включает в себя три основных узла: «DatabaseStation», «ServerStation», «ClientStation». «DatabaseStation» - узел, предназначенный для базы данных со средой выполнения MongoDB. Этот узел включает в себя компонент самой базы данных, а уже внутри этой базы данных хранятся компоненты сущностей, необходимые для работы станции. Для успешного функционирования узел станции БД имеет файл для запуска базы данных, имеющий расширение exe, и файл конфигурации – mongodb.js.

Серверная часть автоматизированной системы представляется узлом «ServerStation». Для сервера используется платформа Node.js, а следовательно, на сервере необходимы все подключаемые модули, которые собраны в компоненте node\_modules. Файл package.json хранит в себе все зависимости, которые нужны для запуска сервера. Он используется при установке модулей. Для запуска сервера используется файл app.js, посредством которого сервер связывается как с сервером базы данный, так и с клиентской частью программного продукта посредством HTTP соединения.

Для клиентской части используется узел «ClientStation», который является самым простым в использовании, так как включает в себя практически только веб-браузер, в который вводится адрес сервера.

Помимо основных трех узлов системы существует еще два, которые относятся к разрабатываемой системе лишь косвенно, так как они помогают визуализировать все интерфейсы узлов. Принтер принадлежит серверной части, однако в реальной среде он может принадлежать как серверу, так и клиентскому узлу.

# **6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

## **6.1 Диаграмма состояний**

Главное предназначение этой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий [11].

Диаграмма состояний системы изображена на рисунке 6.1.1



Рисунок 6.1.1 – Диаграмма состояний программы

При включении программы она переходит в режим ожидания. Далее пользователь может просто выполнять определенные операции, не авторизируясь. В данном случае пользователь будет считаться клиентом, и в следствии чего, программа будет предоставлять различные услуги по составлению тура. Если был выбран пункт входа, система перейдет в состояние авторизации. Пользователь должен ввести данные логина и пароля, и если пароль или логин неверны, то происходит повтор ввода данных. В случае успешной авторизации, оператором считается пользователь, который зашел в кабинет. Система переходит в состояние выполнения операций администратора, и выходит из данного состояния при условии, что пользователь выбрал операцию выхода из личного кабинета. После выхода из кабинета оператор становится обычным пользователем, а следовательно, система переходит в режим выполнения пользовательских операций до тех пор, пока не будет выполнено закрытие программы/браузера.

## **6.2 Обобщенный алгоритм программы**

Наиболее яркое представление о работе и выполнении программы могут дать схемы алгоритмов. Объемные проекты, как правило, следует разбить на несколько алгоритмов, каждый из которых отвечает за какую-то свою определенную функциональность.

Начинать рассмотрение процесса работы программы лучше всего с обобщенного алгоритма, который представлен на рисунке 6.2.1.

Из рисунка видно, что в самом начале алгоритма происходит проверка на авторизацию, то есть если пользователь не авторизировался, то ему предоставляется функционал клиента, а именно: поиск тура, бронирование тура и отправка письма. Все выше перечисленные функции являются составными, что означает, что каждая из них также работает по определенному алгоритму.

Если пользователь является работником турфирмы, то для работы с системой следует произвести авторизацию. Как видно из алгоритма (рисунок 6.2.1), в этом случае главное меню для пользователя будет выглядеть несколько иначе. Оператору предоставляется базовая возможность зарегистрировать нового сотрудника, тем самым создать ему личный кабинет, которым в дальнейшем новый сотрудник может использовать самостоятельно. Кроме этого все операторы имеют возможность найти тур, в случае если клиент обратился в турфирму лично, чтобы тур ему подобрали сотрудники компании, исходя из его предпочтений.

Одной из главных операций в меню администратора является оформление заявлений на бронирование тура. В идеальном случае оператор обязан связаться с клиентом для идентификации как его личности, так и его адекватности.

Оформление документов на путевку – ключевая обязанность сотрудников. Автоматизированная система упрощает процесс составления документов и их предоставление клиентам. В следствие чего, данная операция включена в функционал работника турфирмы.

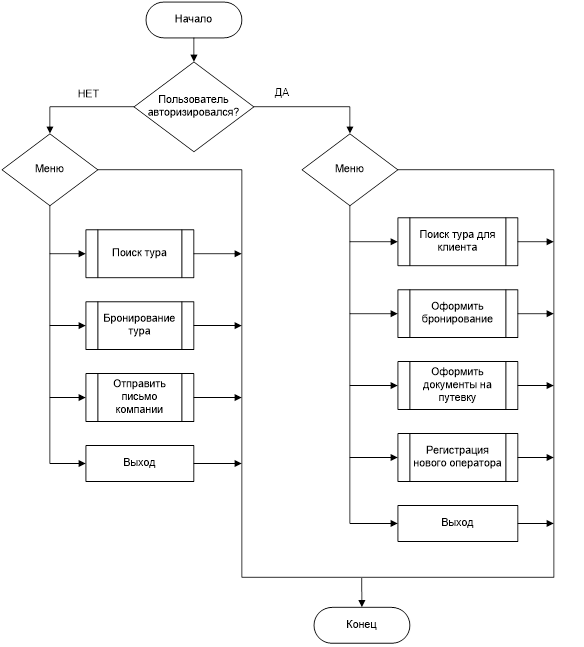


Рисунок 6.2.1 – Обобщенный алгоритм работы программы

## **6.3 Алгоритм бронирования туров**

Бронирование тура является наиболее интересной частью со стороны разработки программного кода. Существуют отдельные программные обеспечения по бронированию не только туров, но и различных видов отелей, билетов на виды транспорта и так далее.

В разрабатываемой системе тур включает в себя полную оплату все видов транспортировки по месту назначения, видов экскурсий, включенных в тур и многое другое в зависимости от тура. Следовательно, алгоритм для бронирования в автоматизированной системе (рисунок 6.2.2) не является настолько обширным, чтобы выделять его в отдельное программное обеспечение, однако он заслуживает внимания.

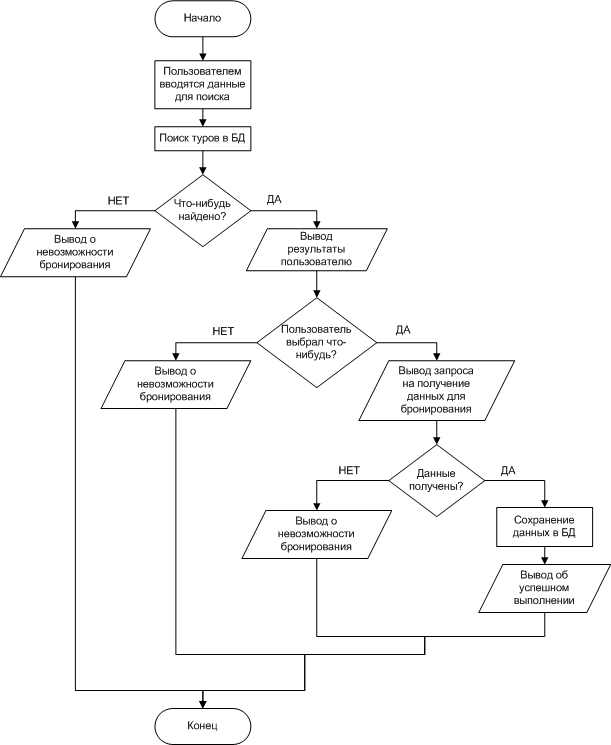


Рисунок 6.2.2 – Алгоритм бронирования тура

Первое, что происходит при бронировании тура – это поиск самого тура, на который нужно повесить бронь. Пользователь вводит в строку поиска интересующиего его данные, либо отфильтровывает существующие посредством введения ограничений на вывод туров. Полученные данные система обрабатывает, а именно: находит наиболее подходящие запросу туры в базе данных, и в соответствии с ними выводит пользователю интересующий его список.

Далее происходит проверка, выбрал ли пользователь конкретный тур, и если да, то программа запрашивает у клиента данные для бронирования, например, такие как контактные данные. Это нужно для дальнейшего оформления бронирования со стороны оператора, чтобы он мог связаться с клиентом.

Если пользователь заполнил и отправил форму с контактными данными, то сервер приложения отправляет полученные данные на сохранение в базу данных.

Более детально алгоритмы представлено в Приложении Б с листингом кода программы.

# **7 ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ**

## **7.1 Паттерн проектирования MVC**

Шаблон проектирования или паттерн (англ. design pattern) в разработке программного обеспечения — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться [12].

Для разработки более структурированного кода и приложения в целом можно использовать специально предназначенный для этого паттерн.

MVC (Model-View-Controller) — это фундаментальный паттерн, который нашел применение во многих технологиях, дал развитие новым технологиям и каждый день облегчает жизнь разработчикам (рисунок 7.1.1).

Впервые паттерн MVC появился в языке SmallTalk. Разработчики должны были придумать архитектурное решение, которое позволяло бы отделить графический интерфейс от бизнес-логики, а бизнес логику от данных. Таким образом, в классическом варианте, MVC состоит из трех частей, которые и дали ему название:

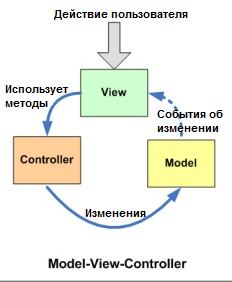


Рисунок 7.1.1 – Паттерн MVC

Под Моделью, обычно понимается часть, содержащая в себе функциональную бизнес-логику приложения. Модель должна быть полностью независима от остальных частей продукта. Модельный слой ничего не должен знать об элементах дизайна, и каким образом он будет отображаться. Достигается результат, позволяющий менять представление данных, то как они отображаются, не трогая саму Модель.

Модель обладает следующими признаками:

* модель — это бизнес-логика приложения;
* модель обладает знаниями о себе самой и не знает о контроллерах и представлениях;
* для некоторых проектов модель — это просто слой данных (DAO, база данных, XML-файл);
* для других проектов модель — это менеджер базы данных, набор объектов или просто логика приложения.

Представление (View). В обязанности Представления входит отображение данных полученных от Модели. Однако, представление не может напрямую влиять на модель. Можно говорить, что представление обладает доступом «только на чтение» к данным.

Представление обладает следующими признаками:

* в представлении реализуется отображение данных, которые получаются от модели любым способом;
* в некоторых случаях, представление может иметь код, который реализует некоторую бизнес-логику;
* для некоторых проектов модель — это просто слой данных (DAO, база данных, XML-файл);

Основная идея этого паттерна в том, что и контроллер, и представление зависят от модели, но модель никак не зависит от этих двух компонент.

Признаки контроллера:

* контроллер определяет, какие представление должно быть отображено в данный момент;
* события представления могут повлиять только на контроллер, контроллер может повлиять на модель и определить другое представление;
* возможно несколько представлений только для одного контроллера;

Реализация:

Контроллер перехватывает событие извне и в соответствии с заложенной в него логикой, реагирует на это событие изменяя Модель, посредством вызова соответствующего метода. После изменения Модель использует событие о том, что она изменилась, и все подписанные на это события Представления, получив его, обращаются к Модели за обновленными данными, после чего их и отображают [13].

В наше время, шаблон MVC был применен для разнообразных языков программирования, включая JavaScript, на котором написана платформа Node.js. Однако его реализация на JavaScript несколько отличается от классической.

На рисунке 7.1.2 изображена часть MVC – модель.

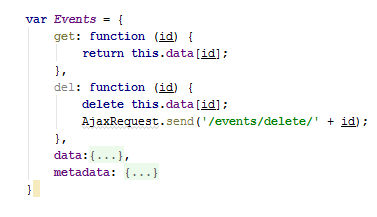


Рисунок 7.1.2 – Модель паттерна MVC

Из рисунка видно, что объект Модель содержит метаданные, определяющие допустимые значений полей события.

В паттерне MVC, вид получает данные и определяет, как их отобразить. Для этого он может использовать существующий HTML, запрашивать HTML блок у сервера, или создавать его с помощью DOM. Вид не беспокоится о том, откуда и как получить данные, он только отображает те данные, которые ему передали (рисунок 7.1.3). Чтобы контроллер мог управлять видом, не беспокоясь о его внутренней реализации, добавляются методы open и close.

Главное правило: вид не должен вызывать свои методы, например, диалог не должен открывать или закрывать себя, это работа контроллера (рисунок 7.1.4). Он активируется, каким-либо событием, это может быть загрузка страницы или действие пользователя, для этого обработчик события связывается с методом Контроллера[14].

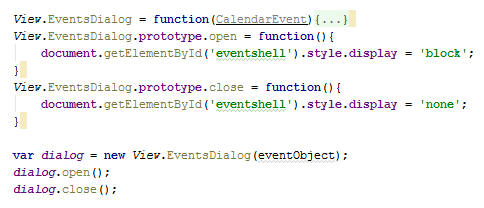


Рисунок 7.1.3 – Вид паттерна MVC

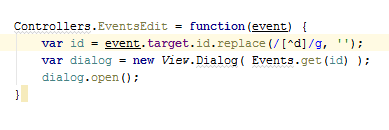


Рисунок 7.1.4 – Один из методов Контроллера паттерна MVC

## **7.2 Паттерн проектирования Singleton**

Часто в системе могут существовать сущности только в единственном экземпляре, например, система ведения системного журнала сообщений или драйвер дисплея. В таких случаях необходимо уметь создавать единственный экземпляр некоторого типа, предоставлять к нему доступ извне и запрещать создание нескольких экземпляров того же типа.

Паттерн Singleton предоставляет такие возможности.

Архитектура паттерна Singleton основана на идее использования глобальной переменной, имеющей следующие важные свойства:

* такая переменная доступна всегда. Время жизни глобальной переменной - от запуска программы до ее завершения;
* предоставляет глобальный доступ, то есть, такая переменная может быть доступна из любой части программы;

Однако, использовать глобальную переменную некоторого типа непосредственно невозможно, так как существует проблема обеспечения единственности экземпляра, а именно, возможно создание нескольких переменных того же самого типа (например, стековых).

Для решения этой проблемы паттерн Singleton возлагает контроль над созданием единственного объекта на сам класс. Доступ к этому объекту осуществляется через статическую функцию-член класса, которая возвращает указатель или ссылку на него. Этот объект будет создан только при первом обращении к методу, а все последующие вызовы просто возвращают его адрес. Для обеспечения уникальности объекта, конструкторы и оператор присваивания объявляются закрытыми (рисунок 7.2.1).

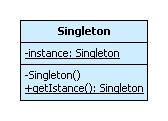


Рисунок 7.2.1 – UML-диаграмма паттерна Singleton

Реализация Singleton на JavaScript имеет ряд трудностей из-за особенностей языка: нет никакой инкапсуляции данных, ссылку на объект необходимо явно прокидывать во все области видимости, где подразумевается его использование; объект конструируется в любом случае, даже если конструктор не был ни разу вызван, невозможность использования цепочки прототипов, некорректный конструктор объекта, возвращаемого «Singletone».

Решение Singleton в реализации JavaScript избегая всех вышесказанных ошибок представлено на рисунке 7.2.2.

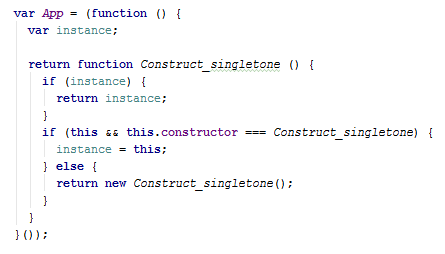


Рисунок 7.2.2 – Реализация Singleton

## **7.3 Паттерн проектирования Observer**

Идея паттерна Observer заключается в создании зависимости типа один ко многим. При изменении состояния одного объекта(субъекта), зависящие от него объекты (наблюдатели) об этом оповещаются и обновляются. Это нужно для согласования состояния взаимосвязанных объектов без их жесткой связанности.

Например, при изменении данных A, B, C диаграмма и таблица, отвечающие за представление, должны измениться. При этом неизвестно, сколько имеется представлений.

Паттерн наблюдатель описывает, как реализовать такое отношение. В основе лежат объекты Subject и Observer. Субъект изменяется и уведомляет о своих изменениях зависимым от него Наблюдателям. Наблюдатели синхронизируют свои данных с Субъектом. Также это отношение называют издатель-подписчик. Субъект (издатель) рассылает уведомления своим наблюдателям (подписчикам), даже не зная о том, какие объекты ими являются. При этом количество подписчиков не ограничено.

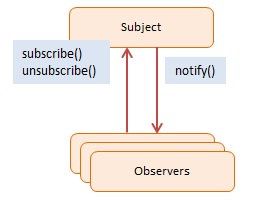


Рисунок 7.3.1 – Паттерн Observer

Одним из ярких примеров реализации паттерна Observerна языке JavaScript является библиотека jQuery. Рисунок 7.3.2 изображает пример использования данного паттерна в рамках jQuery. На элементы DOM ставятся «наблюдатели», которые будут запускать функции (callback) при изменении элемента DOM, а в конкретном примере – при возникновении события нажатия на элемент[14].



Рисунок 7.3.2 – Пример использования паттерна Observer в jQuery

В рамках разрабатываемой автоматизированной системы используется объект наблюдателя, который имитирует собой классическое представление класса Observer. На рисунке 7.3.3 представлена часть данного объекта-класса.

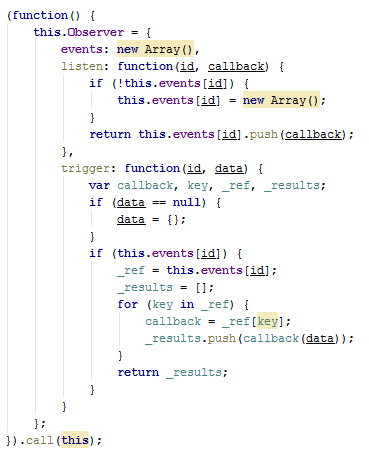


Рисунок 7.3.3 – Пример реализации паттерна Observer в проекте

# **8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ**

Настройка и запуск разработанной системы достаточно прост. Для этого достаточно выполнять ряд простых действий.

## **8.1 Установка необходимых компонент**

Первым делом следует установить базу данных, в которой будет хранится вся необходимая информация. В курсовом проекте использовалась база данных MongoDB, которую нужно скачать с официального сайта MongoDB: <https://www.mongodb.com/download-center#community> (рисунок 8.1).

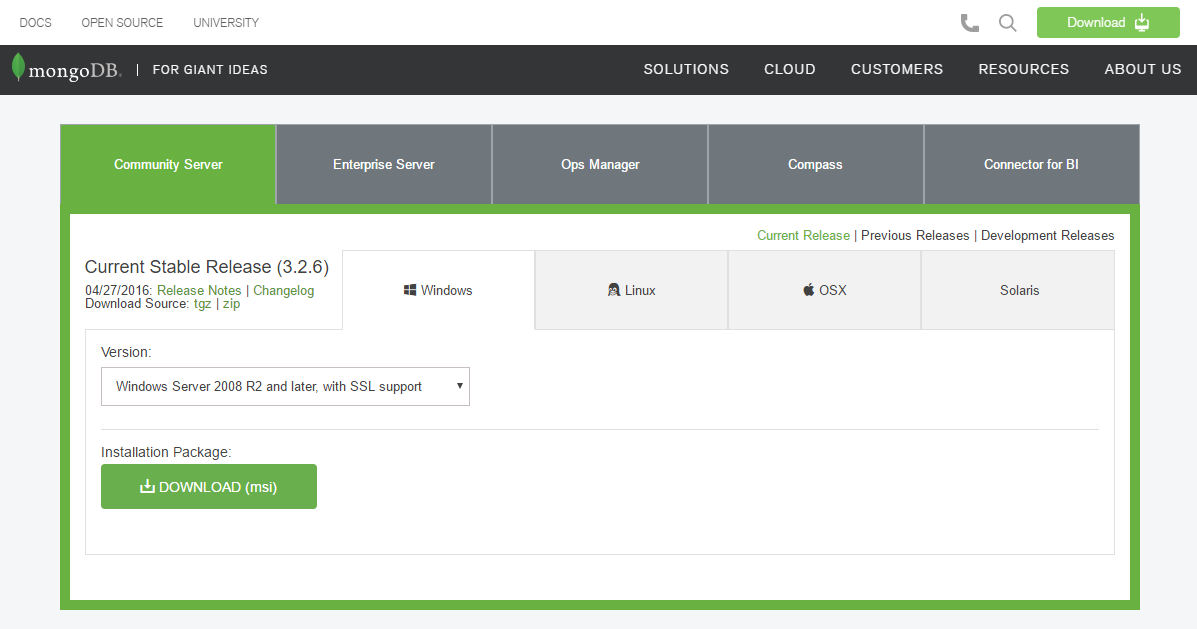


Рисунок 8.1.1 – Страница для скачивания БД MongoDB

На данной странице следует выбрать операционную систему в соответствии с той, которая используется. Далее, следуя всем инструкциям, нужно базу данных установить.

Так как программный продукт написан на платформе Node.js, то в первую очередь нужно установить Node.js. Для этого с официального сайта Node.js: https://nodejs.org/en/download (рисунок 8.1.2) скачивается и устанавливается та версия продукта, которая соответствует операционной системе.

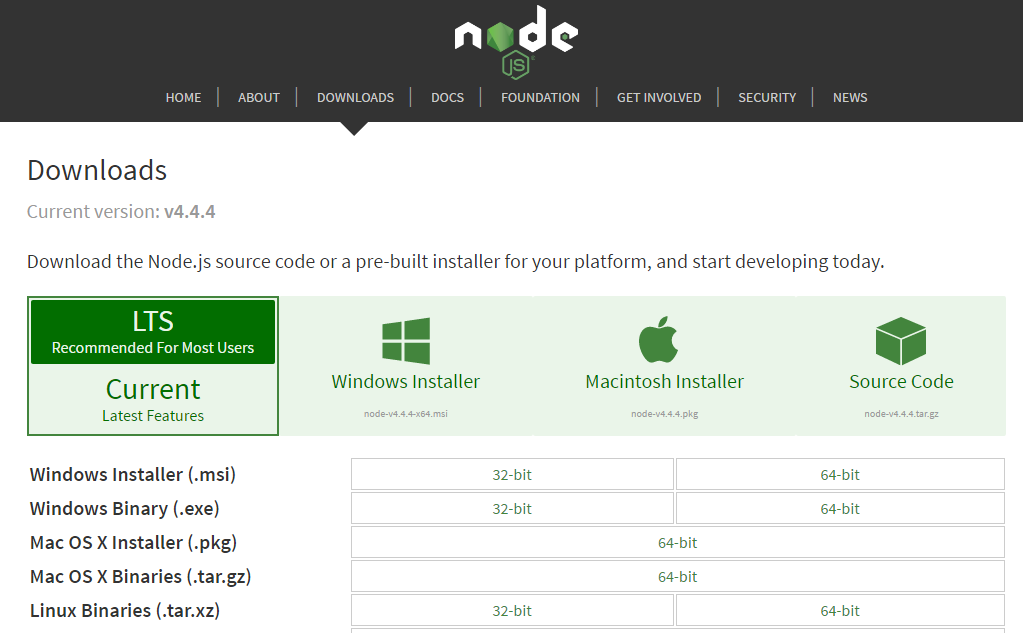


Рисунок 8.1.2 – Страница для скачивания Node.js

Вместе с Node.js устанавливается пакетный менеджер npm. С его помощью можно управлять модулями и зависимостями.

Все зависимости проекта прописаны в файле package,json, расположенном в корне проекта, которые будут использоваться при установке модулей. Сама установка модулей требует открытия командной строки в корневой директории проекта и выполнения команды «npm install» (рисунок 8.1.3). При успешной установке результат вывода в консоль будет примерно такой, как представлен на рисунке 8.1.4.

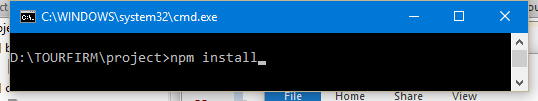


Рисунок 8.1.3 – Команда установки необходимых модулей

Также после выполнения команды установки модулей, в директории проекта появится папка «node\_modules». Предупреждение: нежелательно изменять файлы, хранящиеся в данной папке, это может привести к некорректной работе программного продукта.

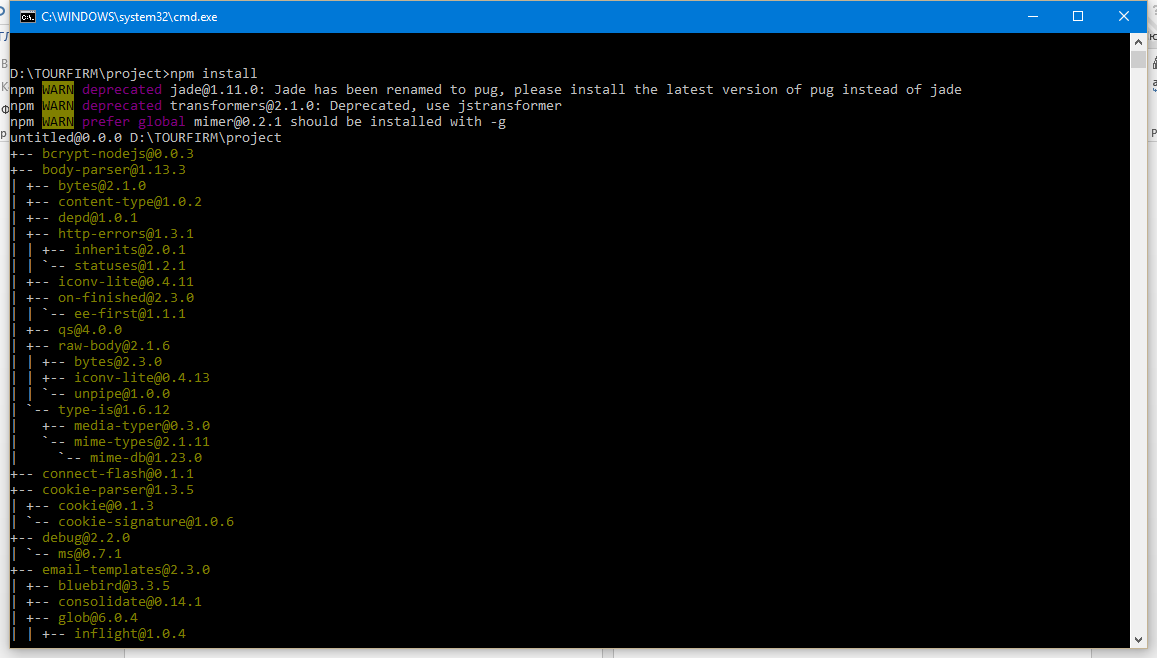


Рисунок 8.1.4 – Результат успешной установки модулей в командной строке

## **8.2 Запуск системы**

После установки всех необходимых компонент становится возможным запуск системы. В первую очередь следует запустить сервер базы данных. Он запускается следующим образом:

* перейти в директорию, куда установлена MongoDB и открыть папку «bin»;
* запустить файл mongod.exe – сервер базы данных;

Примечание: в каталоге bin есть так же файл mongo.exe, который запускает консоль для работы с базой данных без сторонних программ с пользовательским интерфейсом.

Когда сервер базы данных запущен, можно запустить сам проект. В корневом каталоге следует открыть командную строку и выполнить следующую команду: «node bin\www» (рисунок 8.2.1).

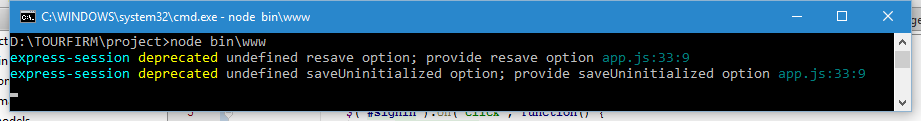


Рисунок 8.2.1 – Запуск сервера системы

Таким образом сервер запущен и все готово к работе. Последним шагом остается запуск клиентской части. Для это нужно запустить web-браузер и в адресной строке ввести «localhost:3001» (рисунок 8.2.2).

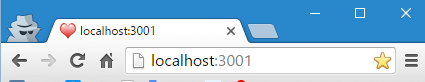


Рисунок 8.2.2 – Запуск клиентской части системы

# **9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ И ОЦЕНКА ВЫПОЛНЕНИЯ ЗАДАЧ**

Разработанная система является приложением, которое может использоваться как клиентами, в качестве удобного использования туристических услуг, так и сотрудниками турфирмы, в качестве организации работы.

При запуске программы, открывается стартовая страница (рисунок 9.1). В верхней часть экрана расположено меню, которое будет прослеживаться на всех страницах приложения, то есть в любой момент можно выбрать любой пункт меню. В качестве клиента на странице можно воспользоваться следующими пунктами меню:

* поиск тура;
* бронирование тура;
* отправить письмо.

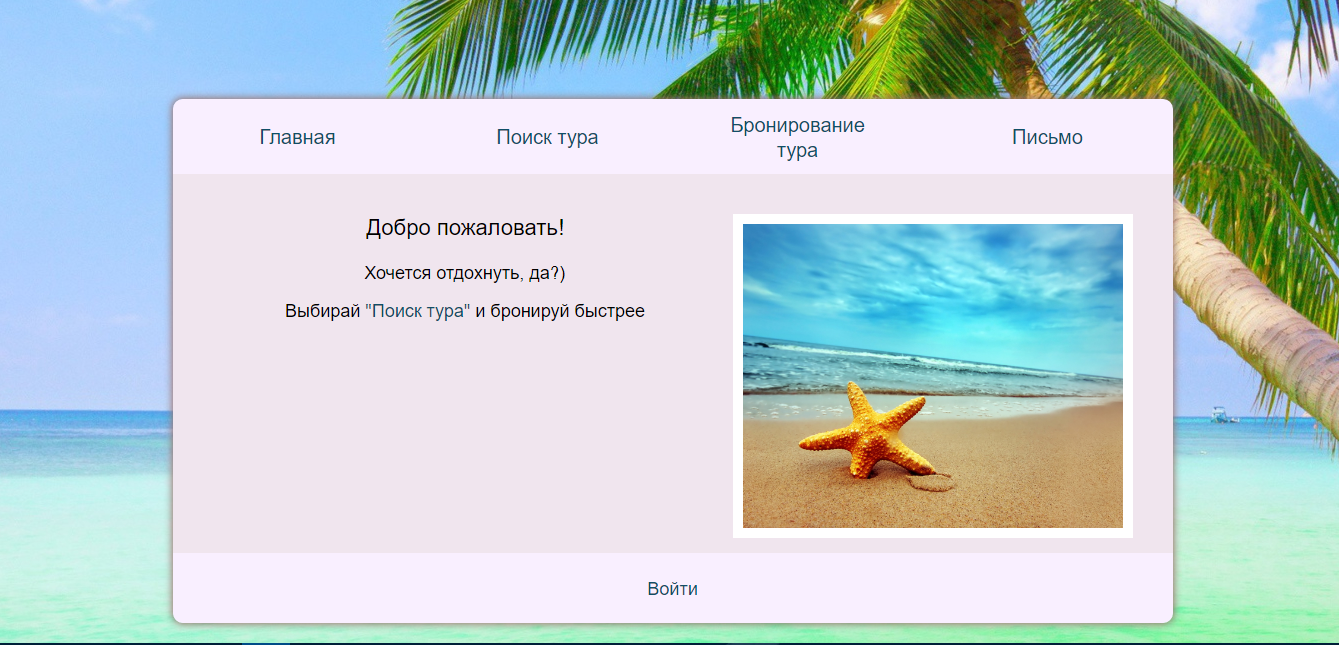


Рисунок 9.1 – Стартовая страница

При переходе на страницу поиска туров появляется несколько полей для ввода (рисунок 9.2). в главной части экрана расположен поиск без фильтрации, то есть в поле можно ввести какое-нибудь предпочтение в туре и нажать на кнопку «Поиск» или клавишу «Enter» на клавиатуре. Если в системе есть туры, удовлетворяющие критерию, то они будут отображены под строкой поиска (рисунок 9.3).

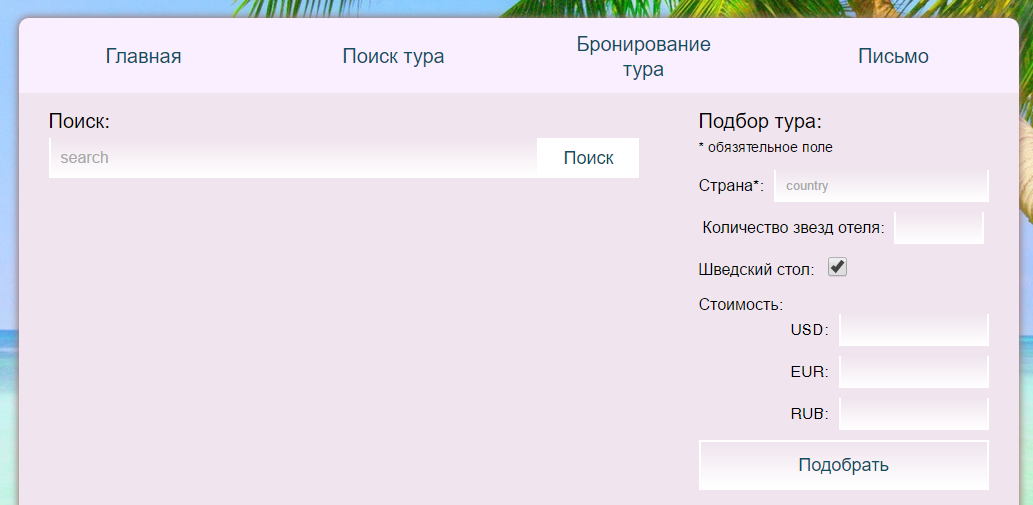


Рисунок 9.2 – Страница поиска туров

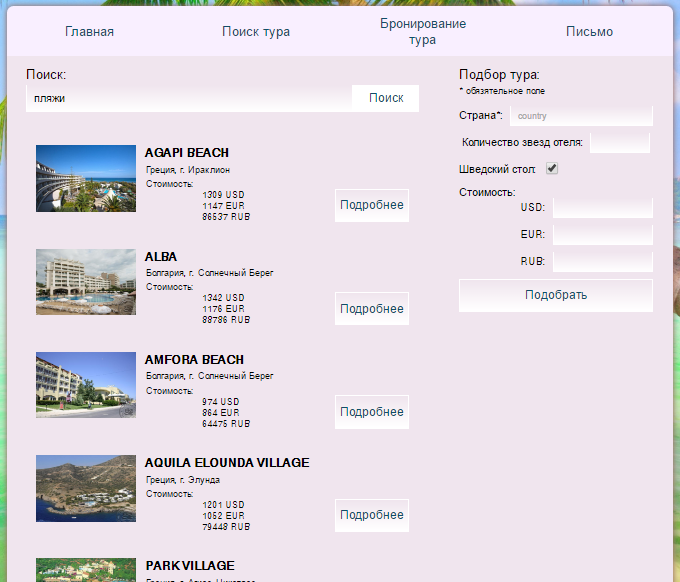


Рисунок 9.3 – Страница с результатом общего поиска

С правой стороны страницы находятся поля для фильтрации результатов, например, если клиент желает отдыхать только в определенной стране и за примерную сумму, указанную в любой валюте (на выбор). Результат такого поиска представлен на рисунке 9.4.

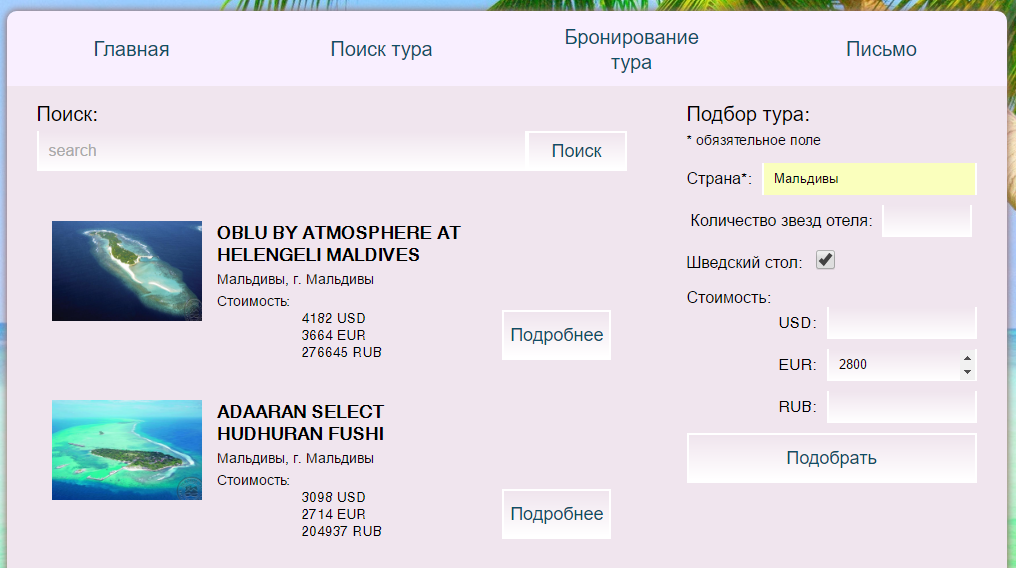


Рисунок 9.4 – страница с результатом фильтрации данных

В результатах поиска выводится минимальная информация о туре, такая как стоимость и местоположение, однако рядом с каждым результатом находится кнопка «Подробнее», нажав на которую, можно перейти на страницу определенно тура с более подробной информацией (рисунок 9.5)

На данной странице можно увидеть более крупное изображение места, развернутое описание тура и места. Также каким образом можно добраться до места назначения и стоимость тура на одного человека, представленная в трех наиболее известных валютах: USD, EUR, RUB.

Если клиент останавливает свой выбор на данном туре, то для этого случая в нижней части экрана расположена кнопка «Оставить заказ на бронь». Это еще не конкретное бронирование тура, так как при заказе через сеть Интернет, оператор должен убедиться в достоверности отправленных клиентом данных, и в случае возникновения вопросов, связаться с клиентом. Поэтому при нажатии на данную клавишу будет выведена страница с формой (рисунок 9.6), которую следует заполнить клиенту для заказа брони.

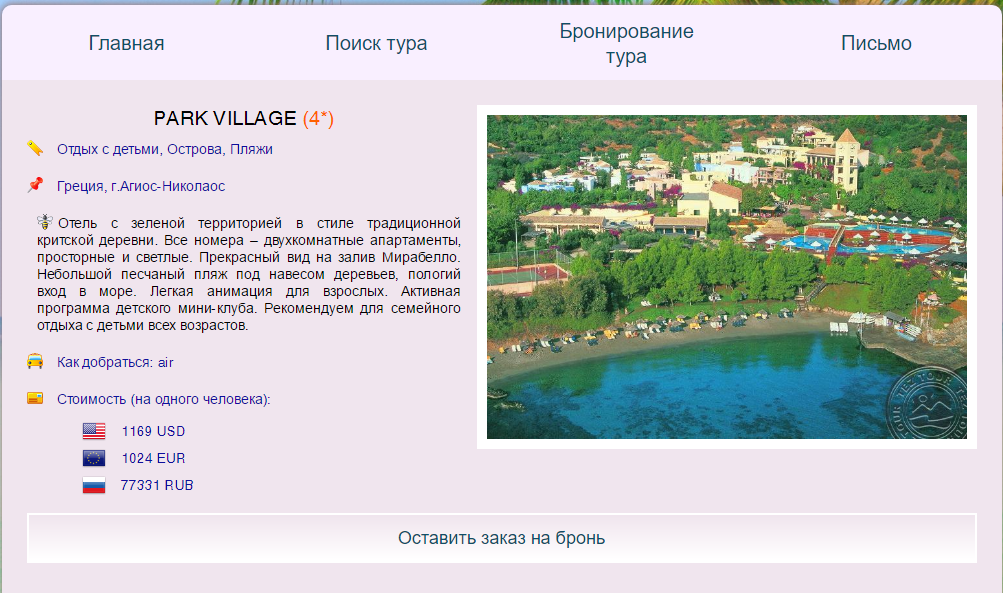


Рисунок 9.5 – Страница информации о туре

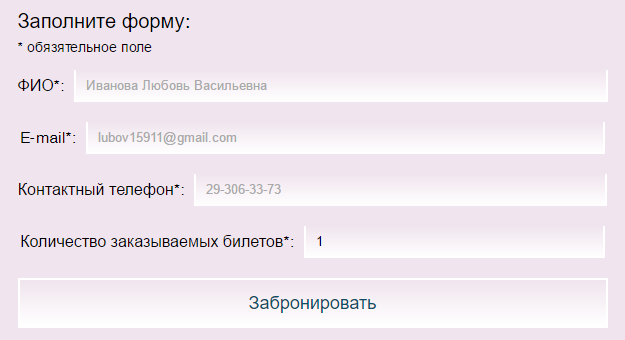


Рисунок 9.6 – Форма для заполнение контактной информации для резервирования брони

После ввода контактной информации следует нажать на кнопку «Забронировать», после чего произойдет отправка данных на сервер.

Если в главном меню, которое расположено в верхней части экрана, выбрать пункт «Бронирование тура», то сначала произойдет перенаправление на страницу поиска туров, так как перед тем, как бронировать, необходимо выбрать что именно нужно бронировать.

Еще одной функциональностью, предоставляемой клиентам, является обратная связь с турфирмой. То есть клиент имеет возможность задать интересующие его вопросы или оставить отзыв о компании или что-либо другое в письме.

Чтобы отправить письмо, нужно перейти на страницу отправки письма (рисунок 9.7). В предложенные поля нужно ввести необходимую информацию и после чего отправить письмо посредством нажатия кнопки «Отправить». После отправления письма клиент перенаправится на домашнюю страницу.

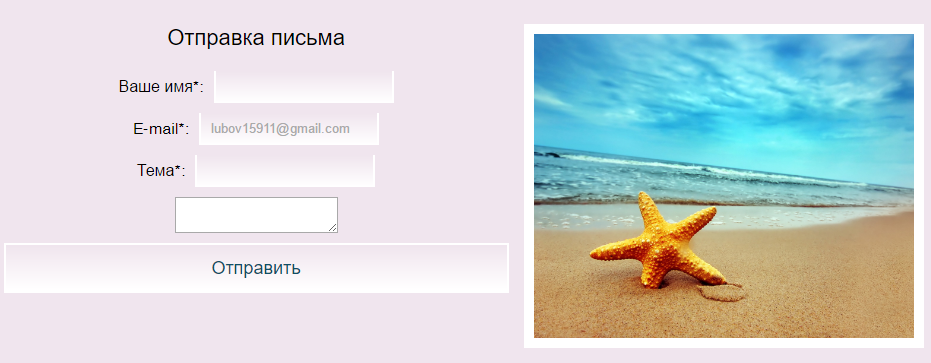


Рисунок 9.7 – Страница отправки письма

В нижней части страницы всегда находится меню для операторов. При начальном запуске приложения, когда сотрудник еще не авторизировался, в этом меню будет находится только один пункт: Вход (рисунок 9.8)

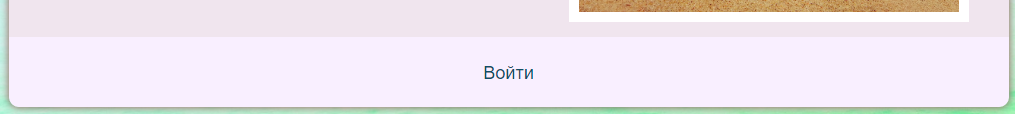


Рисунок 9.8 – Меню не авторизированного сотрудника фирмы

Для входа в личный кабинет необходимо выбрать этот пункт меню, после чего будет предложено вести логин и пароль (рисунок 9.9).

При попытке несанкционированного доступа будет выведено сообщение о том, что пользователя с введенными данными не существует (рисунок 9.10).

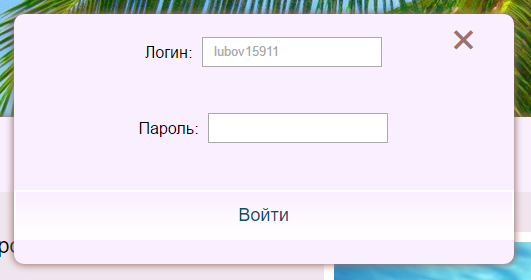


Рисунок 9.9 – Окно авторизации

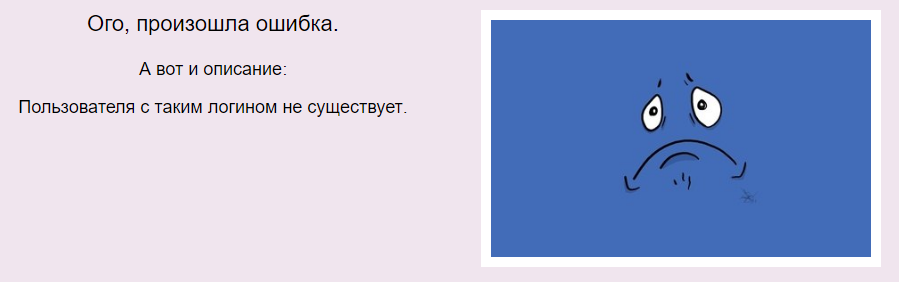


Рисунок 9.10 – Сообщение об ошибке при попытке войти незарегистрированному пользователю

Для операторов, которые по ошибке ввели неверный пароль выводится сообщение о неверном пароле (рисунок 9.11).

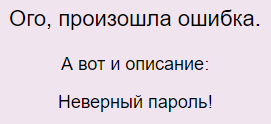


Рисунок 9.11 – Сообщение о неверном пароле

Помимо проверок на неверные данные стоит проверка на попытку отправить пустые поля (рисунок 9.12). Данная проверка стоит на все обязательные поля, которые встречаются в системе (обязательные поля отмечены звездочкой).

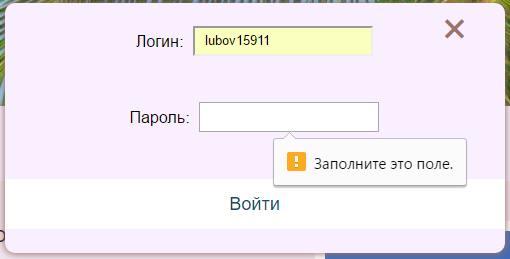


Рисунок 9.12 – Уведомление об обязательном заполнении всех полей

После успешной авторизации сотрудника будет выведена домашняя страница оператора (рисунок 9.13). Нижнее меню поменялось с одного пункта меню на два: Регистрация и Выход. При регистрации нового сотрудника будет выведено окно регистрации (рисунок 9.14).



Рисунок 9.13 – Домашняя страница сотрудника

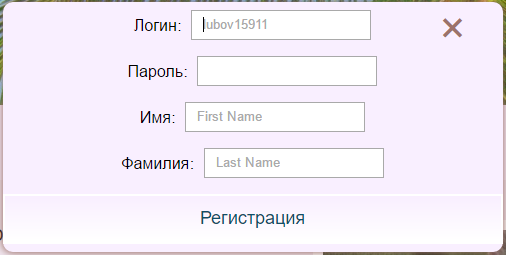


Рисунок 9.14 – Окно регистрации сотрудника

Как видно из рисунка, в окне регистрации добавлены поля с именем и фамилией. При попытке зарегистрировать пользователя с уже существующим логином появится сообщение о невозможности данного действия (рисунок 9.15).

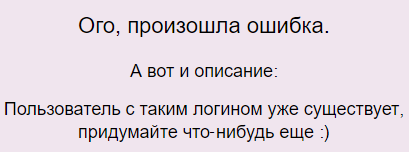


Рисунок 9.15 – Сообщение о невозможности создания нового сотрудника с уже существующим логином

Вместе с нижним меню при авторизации сотрудника поменялось и верхнее. Теперь пункты меню следующие:

* оформить заявление на бронь;
* оформить документы;
* составить анализ продаж.

Также на главной странице администратора есть ссылка на поиск туров, если клиент пришел в турагентство лично с просьбой подобрать ему тур. Алгоритм подбора тура аналогичен подбору тура без авторизации сотрудника.

Одной из главных операций оператора является оформление заявлений на бронь. При выборе данного пункта меню оператор перейден на страницу со списком отправленных заявлений клиентов на бронирование тура (рисунок 9.16). С левой стороны каждого заявления можно поставить галочку, тем самом выбрав несколько заявлений за раз. После чего нажав на клавишу «Оформить выбранные» произойдет изменение данных в базе дынных, а именно: отмеченные заявления получают статус оформленной брони.

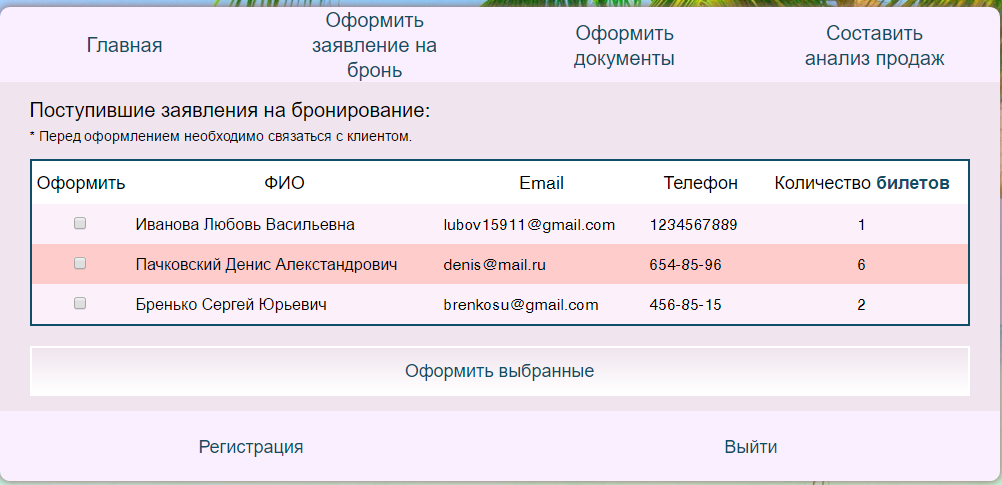


Рисунок 9.16 – Окно со списком отправленных клиентами заявлений

После того, как бронь оформлена, оператор должен оформить документы на путевку, чтобы передать их клиенту. Для этого в верхнем меню нужно нажать на пункт «Оформить документы» (рисунок 9.17).

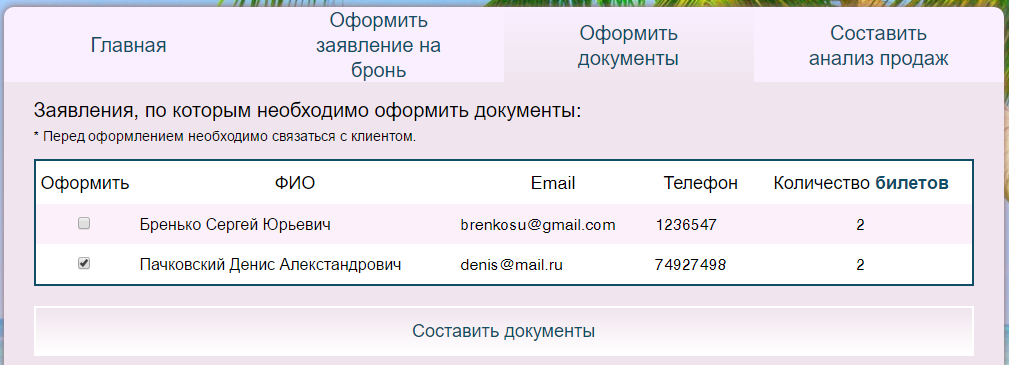


Рисунок 9.17 – Окно оформления документов

Так же как и с оформлением заявлений на бронирование, из списка оформленных броней посредством меток на левой части записи, выбираются заявления, по которым стоит оформить документы. Оформление документов происходит после нажатия на кнопку «Составить документы».

Из списка заявлений выбранные пункты пропадут и удалятся из базы данных. Параллельно с директории «public/files/» будут сохранятся составленные документы (рисунок 9.18).

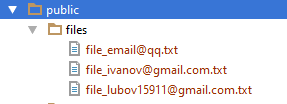


Рисунок 9.18 – Папка с составленными документами

Имена файлов генерируются слиянием шаблонного «file\_» с постой клиента, документы которого оформляются. Так как электронная почта гарантирует уникальность файловых имен. Сам документ включает в себя ФИО клиента, количество билетов, и окончательную стоимость тура в пересчете на количество билетов в трех валютах (рисунок 9.19).

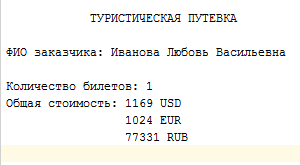


Рисунок 9.19 – Оформленный документ путевки

**ВЫВОДЫ И ЗАКЛЮЧЕНИЯ**

Автоматизация туристической деятельности естественным образом приводит к оптимизации бизнес-процессов. Например, минимизируется время и затраты при прохождении заказа по цепочке от регистрации бронирования в информационной системе до учета факта продажи услуги в бухгалтерской программе. При этом создается база данных, позволяющая делать любые виды отчетности и в любой момент времени получать аналитическую информацию о работе фирмы за любой период. Заметно возрастает качество и скорость обслуживания клиентов.

В ходе выполнения работы были выполнены поставленные задачи посредством детального рассмотрения и изучения основных процессов при предоставлении туристических услуг.

На основе полученных знаний разрабатывалась архитектура проекта с помощью средств UML, стандартов IDEF0 и IEDF1X. Рассмотрены новые технологии разработки, такие как Node.js и Express.js, изучена принципиально новая база данных MongoDB.

Приложение обладает приятным и минималистичным интерфейсом, разработанным средствами шаблонизаторов html, что является довольно удобным решением, принятым с заботой о будущих пользователях программного продукта.

В соответствии с поставленными задачами, система обладает различными возможностями, которые максимально автоматизируют процессы предоставления туристических услуг, что является очень важным для сотрудников турфирмы.

Для клиента возможность дистанционного взаимодействия с турфирмой в наше время является наиболее привлекательным решением при разработке.

Подведя итоги, можно заключить, что разработанный программный продукт соответствует всем заявленным критериям и является точным решением для автоматизированной системы управления работы турфирмы.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

[1] Туризм как экономическая система [Электронный ресурс]. — Электронные данные. — Режим доступа: http://www.aup.ru/books/m19/2\_4.htm

[2] Крутик А.Б. Внедрение систем электронного бронирования и резервирования. // Эффективное развитие туристского предприятия. - 2007.

[3] Осипова О. Я. Транспортное обслуживание. Обслуживание туристов. – М., 2004.

[4] Компьютерные технологии в работе туристических агентств [Текст] // Туризм: практика, проблемы, перспективы. - 2005.- № 11.- С.12-15.

[5] Системы автоматизации для турагентств: какую выбрать [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://pro.tonkosti.ru/it-tehnologii_v_turizme/>sistemy\_avtomatizatsii\_dlya\_turagentstv\_kakuyu\_vybrat- 28792761/

[6] Шилдт, Г. "Java. Полное руководство. 8-е издание". – М: Вильямс.

[7] Крейг Хант.TCP/IP. Сетевое администрирование. Пер. с англ. Отдельное издание/ Зислис, М. – М: «Символ-Плюс», 2006 – 816с.

[8] nodejs.org [Электронный ресурс]. – Электронные данные. – Режим доступа <https://nodejs.org/api/>

[9] Express [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://expressjs.com/>

[10] MongoDB Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.mongodb.com/manual

[11] Крэг Ларман. Применение UML и шаблонов проектирования. 2-е издание. Пер. с англ. – М.: Издательский дом «Вильямс», 2004. — 624 с.

[12] wikipedia.org [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/Шаблон\_проектирования

[13] Гамма, Э. Приемы объектно-ориентированного программирования. Паттерны проектирования. Справочное пособие/ Р. Хелм, Р.Джонсон. – М.: Питер, 2010. — С. 368.

[14] Learning JavaScript Design Patterns [Электронный ресурс]. – Электронные данные. – Режим доступа: https://addyosmani.com/resources/essentialjsdesignpatterns/book/

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Функциональная модель**

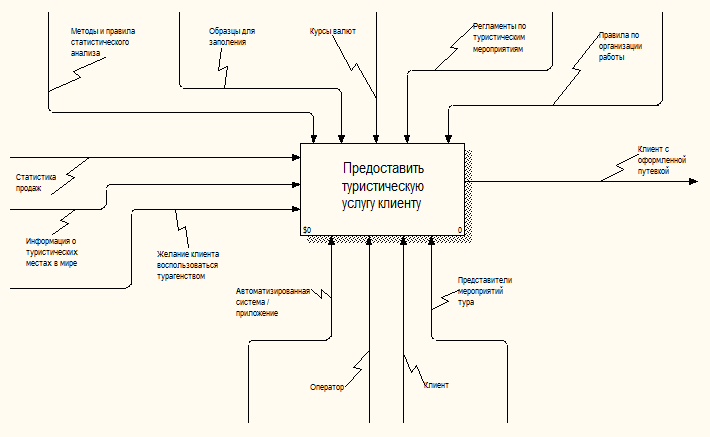


Рисунок А.1 – Контекстная диаграмма

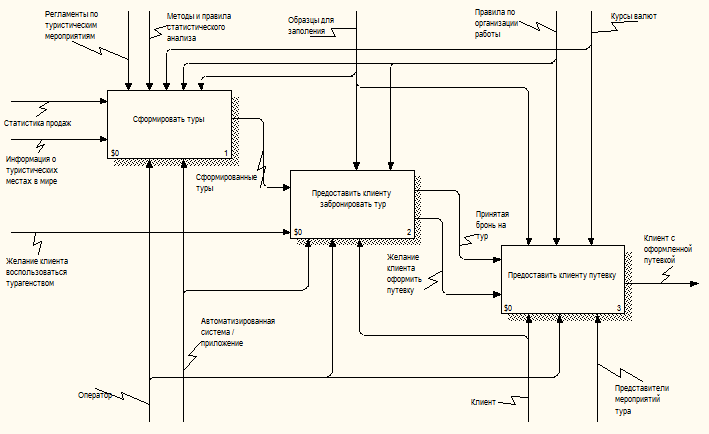


Рисунок А.2 – Декомпозиция первого уровня

Продолжение приложения А

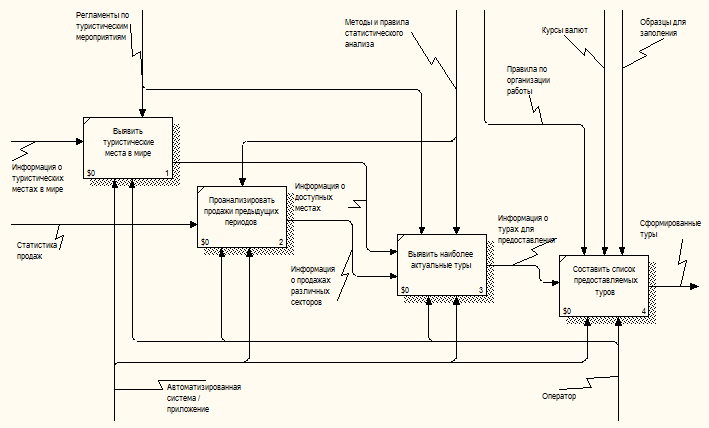


Рисунок А.3 – Декомпозиция второго уровня блока «Сформировать туры»

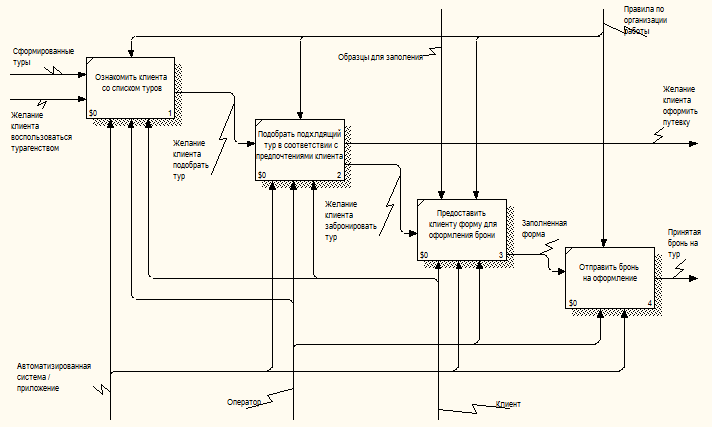


Рисунок А.4 – Декомпозиция второго уровня блока «Предоставить клиенту забронировать тур»

Продолжение приложения А

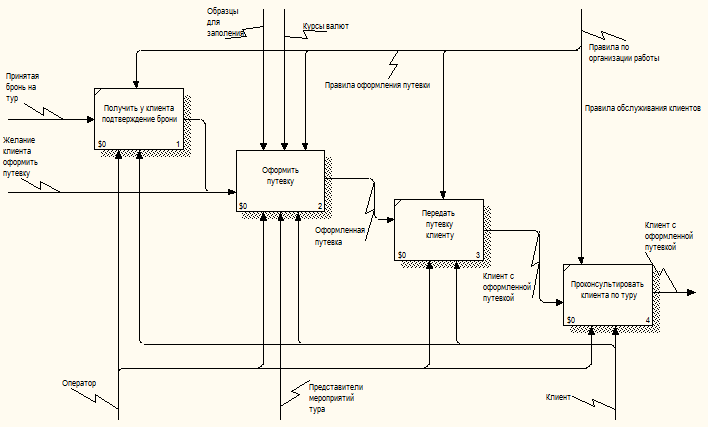


Рисунок А.5 – Декомпозиция второго уровня блока «Предоставить клиенту путевку»

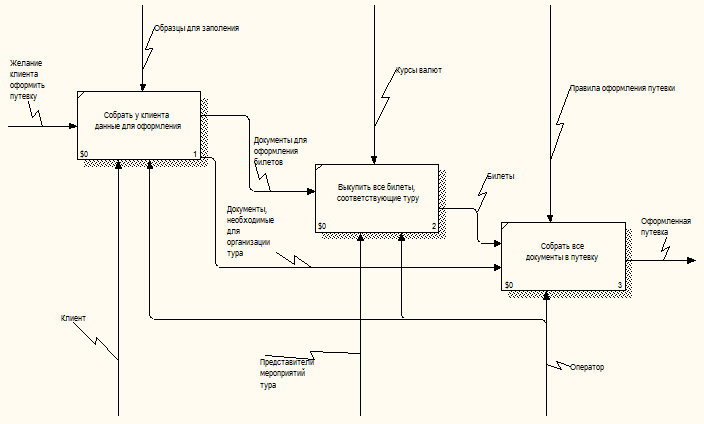


Рисунок А.6 – Декомпозиция третьего уровня блока «Оформить путевку»

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинг основных элементов программы**

**var *express* = require('express');  
var *path* = require('path');  
var *favicon* = require('serve-favicon');  
var *logger* = require('morgan');  
var *cookieParser* = require('cookie-parser');  
var *bodyParser* = require('body-parser');  
  
var *dbConfig* = require('./setup/db');  
var *mongoose* = require('mongoose');  
*mongoose*.connect(*dbConfig*.url);  
  
var *app* = *express*();  
  
*// view engine setup  
app*.set('views', *path*.join(\_\_dirname, 'views'));  
*app*.set('view engine', 'jade');  
  
*app*.use(*favicon*(*path*.join(\_\_dirname, 'public', '/images/1462923104\_15.ico')));  
*app*.use(*logger*('dev'));  
*app*.use(*bodyParser*.json());  
*app*.use(*bodyParser*.urlencoded({ extended: false }));  
*app*.use(*cookieParser*());  
*app*.use(*express*.static(*path*.join(\_\_dirname, 'public')));  
  
*// Passport:*var *passport* = require('passport');  
*//var LocalStrategy = require('passport-local').Strategy;*var *session* = require('express-session');  
  
*app*.use(*session*({ secret: 'SECRET' }));  
*app*.use(*passport*.initialize());  
*app*.use(*passport*.session());  
  
var *flash* = require('connect-flash');  
*app*.use(*flash*());  
  
*// Initialize Passport*var *initPassport* = require('./passport/init');  
*initPassport*(*passport*);  
  
var *routes* = require('./routes/index')(*passport*);  
*app*.use('/', *routes*);  
  
*// catch 404 and forward to error handler  
app*.use(function(req, res, next) {  
 var err = new Error('Not Found');  
 err.status = 404;  
 next(err);  
});  
  
*// error handlers***

Продолжение приложения Б

***// development error handler  
// will print stacktrace*if (*app*.get('env') === 'development') {  
 *app*.use(function(err, req, res, next) {  
 res.status(err.status || 500);  
 res.render('error', {  
 message: err.message,  
 error: err  
 });  
 });  
}  
  
*// production error handler  
// no stacktraces leaked to user  
app*.use(function(err, req, res, next) {  
 res.status(err.status || 500);  
 res.render('error', {  
 message: err.stack,  
 error: {}  
 });  
});  
  
module.exports = *app*;**

**var express = require('express');**

**var router = express.Router();**

**var controller = require('../controllers/controller');**

**var isAuthenticated = function (req, res, next) {**

**// if user is authenticated in the session, call the next() to call the next request handler**

**// Passport adds this method to request object. A middleware is allowed to add properties to**

**// request and response objects**

**if (req.isAuthenticated())**

**return next();**

**// if the user is not authenticated then redirect him to the login page**

**res.redirect('/');**

**};**

**module.exports = function(passport){**

**/\* GET login page. \*/**

**router.get('/', function(req, res) {**

**res.render('index', { message: req.flash('message') });**

**});**

**/\* Handle Login POST \*/**

**router.post('/login', passport.authenticate('login', {**

**successRedirect: '/home',**

**failureRedirect: '/error',**

**failureFlash : true**

**}));**

Продолжение приложения Б

**/\* Handle Registration POST \*/**

**router.post('/signup', passport.authenticate('signup', {**

**successRedirect: '/home',**

**failureRedirect: '/error',**

**failureFlash : true**

**}));**

**/\* GET Home Page \*/**

**router.get('/home', isAuthenticated, function(req, res){**

**res.render('home', { user: req.user });**

**});**

**/\* Handle Logout \*/**

**router.get('/signout', function(req, res) {**

**req.logout();**

**res.redirect('/');**

**});**

**router.get('/error', function(req, res) {**

**res.render('error', { message: req.flash('message') });**

**});**

**router.get('/search', function(req, res) {**

**res.render('search', { user: req.user }); //**

**});**

**router.post('/search', controller.search);**

**router.post('/tour', controller.getTourInfo);**

**router.get('/reservation', function(req, res) {**

**res.redirect('search');**

**});**

**router.post('/reservation', function(req, res) {**

**res.render('reservation', { message: req.body.id });**

**});**

**router.post('/sendReservation', controller.sendReservation);**

**router.post('/filter', controller.filter);**

**router.get('/offlinereservation', isAuthenticated, controller.getReservations);**

**router.post('/executeReservation', controller.executeReservation);**

**router.post('/createDocs', controller.createDocs);**

**router.get('/createdocs', isAuthenticated, controller.getDataForDocs);**

**router.post('/deletedocs', controller.deleteDataForDocs);**

**router.get('/statistics', isAuthenticated, function(req, res) {**

**res.render('statistics', { user: req.user });**

**});**

**router.get('/news', function(req, res) {**

**res.render('news');**

**});**

**router.post('/email', controller.email);**

**return router;};**

Продолжение приложения Б

var Tour = require('../models/tour');

module.exports = function(req, res, next) {

var re = req.body;

var filter = {

'hotel.address.country': re.country,

'hotel.stars': re.stars,

'hotel.cold\_table': re.cold\_table,

'price.USD': { $gte: (+re.USD - 200), $lte: (+re.USD + 200) },

'price.EUR': { $gte: (+re.EUR - 200), $lte: (+re.EUR + 200) },

'price.RUB': { $gte: (+re.RUB - 200), $lte: (+re.RUB + 200) }

};

if (!re.stars) delete filter['hotel.stars'];

if (!re.cold\_table) delete filter['hotel.cold\_table'];

if (!re.USD) delete filter['price.USD'];

if (!re.EUR) delete filter['price.EUR'];

if (!re.RUB) delete filter['price.RUB'];

return Tour.find(filter, function (err, tour) {

if (!err) {

console.log(tour);

return res.render('search', { message: tour });

} else {

res.statusCode = 500;

console.error('Internal error(%d): %s', res.statusCode, err.message);

return res.send({error: 'Server error'});

}

})

}

console.log('4\*\*\*\*\*\*\*\*\*\*\*');

console.log(req.user);

console.log('5\*\*\*\*\*\*\*\*\*\*\*');

return Tour.find(filter, function (err, tour) {

if (!err) {

console.log(tour);

return res.render('search', { message: tour });

} else {

res.statusCode = 500;

console.error('Internal error(%d): %s', res.statusCode, err.message);

return res.send({error: 'Server error'});

}

})

}

Продолжение приложения Б

var Reservation = require('../models/reservation');

module.exports = function(req, res, next) {

for (var i in req.body) {

Reservation.findById(i, function (err, reservation) {

if (err) return handleError(err);

if (reservation.deleteStatus == false) {

reservation.print();

reservation.deleteStatus = true;

reservation.save(function (err) {

if (err) {

console.log('3ERROR!!!');

console.log(err.stack);

}

});

var tickets = reservation.tickets;

var id = reservation.tour\_id;

}

});

}

return res.redirect('/createdocs');

}

var login = require('./login');

var signup = require('./signup');

var User = require('../models/user');

module.exports = function(passport){

// Passport needs to be able to serialize and deserialize users to support persistent login sessions

passport.serializeUser(function(user, done) {

console.log('serializing user: ');

console.log(user);

done(null, user.\_id);

});

passport.deserializeUser(function(id, done) {

User.findById(id, function(err, user) {

console.log('deserializing user:',user);

done(err, user);

});

});

// Setting up Passport Strategies for Login and SignUp/Registration

login(passport);

signup(passport);

};

Продолжение приложения Б

var LocalStrategy = require('passport-local').Strategy;

var User = require('../models/user');

var bCrypt = require('bcrypt-nodejs');

module.exports = function(passport){

passport.use('login', new LocalStrategy({

passReqToCallback : true

},

function(req, username, password, done) {

// check in mongo if a user with username exists or not

User.findOne({ 'username' : username },

function(err, user) {

// In case of any error, return using the done method

if (err)

return done(err);

// Username does not exist, log the error and redirect back

if (!user){

console.log('User Not Found with username '+username);

return done(null, false, req.flash('message', 'Пользователя с таким логином не существует.'));

}

// User exists but wrong password, log the error

if (!isValidPassword(user, password)){

console.log('Invalid Password');

return done(null, false, req.flash('message', 'Неверный пароль!')); // redirect back to login page

}

// User and password both match, return user from done method

// which will be treated like success

return done(null, user);

}

);

})

);

var isValidPassword = function(user, password){

return bCrypt.compareSync(password, user.password);

}

}

var mongoose = require('mongoose');

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

module.exports = function(done) {

mongoose.connection.on('open', function() {

console.info('Connected to mongo server!');

return done();

});

mongoose.connection.on('error', function(err) {

console.error('Could not connect to mongo server!');

console.error(err.message);

return done(err);

});

Продолжение приложения Б

try {

mongoose.connect('mongodb://localhost:27017/tourfirm');

var tmp = new user({ username: 'root', password: 'root' });

tmp.save(function(err) {

if (err) {

console.log('123///');

console.log(err);

} else {

console.log('MEOW');

}

});

console.info('Started connection on mongodb://localhost/tourfirm, waiting for it to open...');

} catch (err) {

console.error('Setting up failed to connect to mongodb://localhost/tourfirm');

console.error(err.message);

done(err)

}

};

var passport = require('passport');

var LocalStrategy = require('passport-local').Strategy;

var mongoose = require('mongoose');

var User = mongoose.model('user');

module.exports = function() {

passport.use(new LocalStrategy({

usernameField: 'email',

passwordField: 'password'

}, function (username, password, done) {

User.findOne({username: username}, function (err, user) {

return err

? done(err)

: user

? password === user.password

? done(null, user)

: done(null, false, {message: 'Incorrect password.'})

: done(null, false, {message: 'Incorrect username.'});

});

}));

passport.serializeUser(function (user, done) {

done(null, user.id);

});

passport.deserializeUser(function (id, done) {

User.findById(id, function (err, user) {

err

? done(err)

: done(null, user);

});

});

Продолжение приложения Б

*var search = require('./searching');*

*var tour = require('./tour-info');*

*var sendReservation = require('./sendReservation');*

*var filter = require('./filter');*

*var getReservations = require('./getReservations');*

*var executeReservation = require('./executeReservation');*

*var getDataForDocs = require('./getDataForDocs');*

*var createDocs = require('./createDocs');*

*var deleteDataForDocs = require('./deleteDataForDocs');*

*var email = require('./email');*

*module.exports = {*

*search: search,*

*getTourInfo: tour,*

*sendReservation: sendReservation,*

*filter: filter,*

*getReservations: getReservations,*

*executeReservation: executeReservation,*

*getDataForDocs: getDataForDocs,*

*createDocs: createDocs,*

*deleteDataForDocs: deleteDataForDocs,*

*email: email*

*};*

{

"name": "untitled",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "node ./bin/www"

},

"dependencies": {

"body-parser": "~1.13.2",

"cookie-parser": "~1.3.5",

"debug": "~2.2.0",

"express": "~4.13.1",

"jade": "~1.11.0",

"morgan": "~1.6.1",

"serve-favicon": "~2.3.0",

"passport-local": "~0.1.6",

"passport": "~0.3.2",

"express-session": "~1.13.0",

"mongoose": "~4.4.14",

"connect-flash": "~0.1.1",

"bcrypt-nodejs": "~0.0.3",

"fs": "~0.0.2",

"node-zip": "~1.1.1",

"nodemailer-smtp-transport": "~2.5.0",

"nodemailer": "~2.4.0",

"email-templates": "~2.3.0",

"q": "~1.4.1"

}

}

# **ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Листинг скрипта генерации базы данных**

*var tour = new Tour({*

*\_id: 1,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Ираклион"*

*},*

*stars: 4,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/3265/AGAPI\_BEACH\_318\_Overview\_1\_4363\_middle.jpg" //link to photos*

*},*

*tour\_name: "AGAPI BEACH",*

*transportation: "air",*

*price: {*

*USD: 1309,*

*EUR: 1147,*

*RUB: 86537*

*},*

*number\_of\_tickets: 1000,*

*description: "Отель с зеленой ухоженной территорией. Разнообразное питание на основе программы «все включено», ежедневные вечерние и дневные мероприятия для взрослых и детей. Развитая инфраструктура. Есть русскоязычный персонал. Напротив отеля автобусная остановка, удобное сообщение с центром Ираклиона. Рекомендуем для семейного отдыха.",*

*tags: "Отдых с детьми, Острова, Пляжи"*

*});*

*var tour2 = new Tour({*

*\_id: 2,*

*hotel: {*

*address: {*

*country: "Россия",*

*city: "Кисловодск"*

*},*

*stars: 4,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/50002379/4\_363\_middle.jpg" //link to photos*

*},*

*tour\_name: "SOLNECHNYI KISLOVODSK",*

*transportation: "train",*

*price: {*

*USD: 208,*

*EUR: 182,*

*RUB: 13760*

*},*

*number\_of\_tickets: 200,*

*description: "«Солнечный» — один из лучших и широко известных санаториев Юга России. Европейский уровень комфорта и высшее качество лечебных и оздоровительных услуг — вот почему даже самые взыскательные клиенты нашей страны и зарубежья возвращаются в «Солнечный» снова и снова.",*

*tags: "Отдых с детьми, СПА и термы"*

*});*

*var tour3 = new Tour({*

*\_id: 3,*

*hotel: {*

*address: {*

Продолжение приложения В

*country: "Болгария",*

*city: "Солнечный Берег"*

*},*

*stars: 3,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/7000054/IMG\_9935\_8252\_middle.JPG" //link to photos*

*},*

*tour\_name: "AMFORA BEACH",*

*transportation: "train",*

*price: {*

*USD: 974,*

*EUR: 864,*

*RUB: 64475*

*},*

*number\_of\_tickets: 500,*

*description: "Бывший отель Delta Beach с новым названием и новым менеджментом. Полностью обновлен. Удобно расположен рядом с пляжем и многими дискотеками. Рекомендуем для молодежного отдыха.",*

*tags: "Пляжи"*

*});*

*var tour4 = new Tour({*

*\_id: 4,*

*hotel: {*

*address: {*

*country: "Болгария",*

*city: "Солнечный Берег"*

*},*

*stars: 4,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/7014120/Panorama\_pool\_hotel\_8109\_2759\_middle.jpg" //link to photos*

*},*

*tour\_name: "ALBA",*

*transportation: "air",*

*price: {*

*USD: 1342,*

*EUR: 1176,*

*RUB: 88786*

*},*

*number\_of\_tickets: 500,*

*description: "Отель находится недалеко от пляжа, в самом центре курорта, рядом с развлечениями и дискотеками. Подходит для активного молодежного отдыха. Не рекомендуем для спокойного отдыха.",*

*tags: "Пляжи"*

*});*

*var tour5 = new Tour({*

*\_id: 5,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: " Элунда"*

*},*

*stars: 5,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/2929/1311786865069\_middle.jpg" //link to photos*

*},*

*tour\_name: "AQUILA ELOUNDA VILLAGE",*

*transportation: "air",*

Продолжение приложения В

*price: {*

*USD: 1201,*

*EUR: 1052,*

*RUB: 79448*

*},*

*number\_of\_tickets: 200,*

*description: "Удобство, комфорт и профессиональный сервис – визитная карточка отеля. Панорамный вид на море и каскад бассейнов придают отелю экзотический облик. Рекомендуем для спокойного респектабельного отдыха пар. Из-за наличия спусков и подъемов людям с ограниченными возможностями передвижения отель не рекомендуется.",*

*tags: "Отдых без детей, Острова, Пляжи, СПА и термы, VIP-отдых и услуги"*

*});*

*var tour6 = new Tour({*

*\_id: 6,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Агиос-Николаос"*

*},*

*stars: 4,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/3106/1311787176281\_middle.jpg" //link to photos*

*},*

*tour\_name: "PARK VILLAGE",*

*transportation: "air",*

*price: {*

*USD: 1169,*

*EUR: 1024,*

*RUB: 77331*

*},*

*number\_of\_tickets: 500,*

*description: "Οтель с зеленой территорией в стиле традиционной критской деревни. Все номера – двухкомнатные апартаменты, просторные и светлые. Прекрасный вид на залив Мирабелло. Небольшой песчаный пляж под навесом деревьев, пологий вход в море. Легкая анимация для взрослых. Активная программа детского мини-клуба. Рекомендуем для семейного отдыха с детьми всех возрастов.",*

*tags: "Отдых с детьми, Острова, Пляжи"*

*});*

*var tour7 = new Tour({*

*\_id: 7,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Элунда"*

*},*

*stars: 5,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/721/1311782930032\_middle.jpg" //link to photos*

*},*

*tour\_name: "DOMES OF ELOUNDA AUTOGRAPH COLLECTION",*

*transportation: "air",*

*price: {*

*USD: 3500,*

*EUR: 3066,*

*RUB: 231530*

*},*

*number\_of\_tickets: 500,*

Продолжение приложения В

*description: "Престижный отель, расположенный в уединенном месте, с видом на остров Спиналонга. Истинная средиземноморская архитектура, мягкие тона интерьеров, изысканная мебель ручной работы – все это гармонирует с окружающей средой. Разнообразный номерной фонд от двухкомнатных сьютов до роскошных резиденций с индивидуальным бассейном. Рекомендуем как для респектабельного семейного отдыха, так и для отдыха романтических пар. Есть возможность проведения свадебных церемоний (символические, официальные церемонии с выездом Мэра, венчание в часовне на территории комплекса). В 3,5 км от отеля находится Элунда, туристический поселок славится своими рыбными тавернами со свежими морепродуктам, а в 1,5 км – историческое местечко Плака.",*

*tags: "Острова, Отдых с детьми, Пляжи, СПА и термы, VIP-отдых и услуги"*

*});*

*var tour8 = new Tour({*

*\_id: 8,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Агиос-Николаос"*

*},*

*stars: 3,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/2774/TERRACE\_VIEW\_7873\_middle.jpg" //link to photos*

*},*

*tour\_name: "ELOUNDA AKTI OLOUS",*

*transportation: "air",*

*price: {*

*USD: 1697,*

*EUR: 1486,*

*RUB: 112259*

*},*

*number\_of\_tickets: 500,*

*description: "Небольшой уютный отель с ограниченной территорией. Находится в курортном местечке Элунда, где множество таверн и кафе. Рекомендуем для экономичного отдыха, а также для молодежи.",*

*tags: "Отдых без детей, Острова, Пляжи"*

*});*

*var tour9 = new Tour({*

*\_id: 9,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Фалираки"*

*},*

*stars: 3,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/50002279/lellis\_achousa\_146\_260\_middle.jpg" //link to photos*

*},*

*tour\_name: "ACHOUSA HOTEL",*

*transportation: "air",*

*price: {*

*USD: 918,*

*EUR: 804,*

*RUB: 60727*

*},*

*number\_of\_tickets: 500,*

*description: "Небольшой отель с ухоженной территорией, утопающей в зелени, и простым номерным фондом. Все номера оборудованы кухонным уголком.*

Продолжение приложения В

*Удачное расположение в местечке Фалираки, с его ресторанами, барами и тавернами. Рекомендуем для бюджетного отдыха любой категории туристов.",*

*tags: "Острова, Пляжи"*

*});*

*var tour10 = new Tour({*

*\_id: 10,*

*hotel: {*

*address: {*

*country: "Греция",*

*city: "Родос"*

*},*

*stars: 5,*

*cold\_table: true,*

*image: "http://s.tez-tour.com/hotel/2211/1311785533585\_middle.jpg" //link to photos*

*},*

*tour\_name: "ALDEMAR AMILIA MARE",*

*transportation: "air",*

*price: {*

*USD: 2686,*

*EUR: 2353,*

*RUB: 177682*

*},*

*number\_of\_tickets: 500,*

*description: "Aldemar Resorts – одна из ведущих гостиничных сетей в Греции. Внимательный русскоговорящий персонал, хорошее питание. Развитая детская инфраструктура. Активная анимация в течение всего дня и вечерняя развлекательная программа. Хороший песчано-галечный пляж отмечен Голубым флагом. Рекомендуем для отдыха туристов любого возраста, семей с детьми и молодежи.",*

*tags: "Отдых с детьми, Острова, Пляжи, VIP-отдых и услуги"*

*});*

*tour.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour2.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour3.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour4.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour5.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

Продолжение приложения В

*}*

*});*

*tour6.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour7.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour8.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour9.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*

*tour10.save(function(err) {*

*if (err) {*

*console.log('3ERROR!!!');*

*console.log(err.stack);*

*}*

*});*