МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА
на тему

ИНФОРМАЦИОННАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА ТУРИСТИЧЕСКОГО АГЕНТСТВА

Студент _____ *31.05.2023* (подпись, дата) Пронин В.А.

Руководитель _____ *31.05.2023* (подпись, дата) Карамышева Н.С.

Нормоконтролёр _____ *31.05.2023* (подпись, дата) А.В. Бычков

Работа допущена к защите (протокол заседания кафедры от 2.06.2023 № 14)

Заведующий кафедрой _____ (подпись) М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от 16.06.23 № 17)

Секретарь ГЭК _____ (подпись) Е.И. Гурин

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники | *Кафедра* «Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

_____ М.А. Митрохин

«26» апреля 2023 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Пронин Валерий Андреевич гр. 19ВВП1 факультета вычислительной техники.

2. Тема работы: «Информационная автоматизированная система туристического агентства»

Тема утверждена приказом ПГУ №458/о от «26» апреля 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать информационную автоматизированную систему туристического агентства на основе интернет – технологий, позволяющее автоматизировать ручной процесс учета оказания туристических услуг в агентстве.

5. Перечень подлежащих разработке вопросов. Требуется провести анализ предметной области: изучить возможности существующих систем аналогичного назначения, выделить их достоинства и недостатки. Выбрать средства разработки для базы данных и для веб – интерфейса. Определить архитектуру приложения.

6. Календарный график выполнения бакалаврской работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	50	26.04.2023 – 29.04.2023	
2	Разработка требований к АС	10	30.04.2023– 2.05.2023	
4	Выбор технологий, применяемых при проектировании и разработке	50	3.05.2023 – 6.05.2023	
5	Проектирование системы	50	7.05.2023 – 10.05.2023	
6	Разработка основной части программы	130	11.05.2023– 22.05.2023	
7	Раздел обеспечения безопасности жизнедеятельности	10	23.05.2023– 26.05.2023	
8	Оформление пояснительной записки	24	27.05.2023– 31.05.2023	

Дата выдачи задания 26 апреля 2023 г.

Руководитель бакалаврской работы 26.04.23 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент Пронин В.А.
(подпись, дата)

Работу к защите допустить Фионова Л.Р.
Декан факультета 09.06.2023
(подпись, дата)

Реферат

Бакалаврская работа содержит 153 л., 70 рис., 19 табл., 10 источн., 3 прил.

ЦЕНТР ТУРИСТИЧЕСКОГО АГЕНТСТВА, ТУРИСТИЧЕСКИЕ УСЛУГИ, *UML*, АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР, *MYSQL*, *JAVA SCRIPT, CSS*

Объектом исследования является автоматизация бизнес-процессов в ООО «Центр туристского сервиса *Insertum*».

Цель работы - разработать автоматизированную информационную систему центра туристического агентства.

В процессе работы проведен анализ предметной области, выполнено проектирование и разработано программное обеспечение информационной системы центра туристического сервиса.

Программная реализация выполнена на языке программирования *PHP 7.4* (фреймворк *Yii2*), стандартизированного языка гипертекстовой разметки документов *HTML, Java Script, CSS*. Среда проектирование базы данных - *MySQL Workbench 8.0.33*, СУБД - *MySQL*.

В результате проделанной работы разработана система, позволяющая решить задачи автоматизации учета клиентов, заказов, оплаты туристских заказов, а также ведение документооборота.

Эффективность разработки заключается в повышении производительности труда сотрудников фирмы, уменьшении расхода времени на формировании отчетности.

Данная система может применяться в других туристических агентствах, использующих для учета вычислительную технику и программное обеспечение.

					ПГУ 1.090301.20.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Пронин В.А.			Информационная автоматизированная система туристического агентства. Пояснительная записка	Лит.	Лист 2 / Листов 153
Провер.		Карамышева НС					
Н. контр.		Бычков А.С.				ФВТ, каф. ВТ гр. 19ВВП1	
Утв.		Митрохин М.А.					

Содержание

Перечень сокращений и обозначений 5
Введение 6
1 Автоматизация деятельности центра туристического сервиса 7
1.1 Анализ предметной области 7
1.2 Анализ рынка готовых программных продуктов 9
1.2.1 Система «Само-ТурАгент» 10
1.2.2 Система «*AKAMSOFT* ТУРАГЕНТСТВО» 11
1.2.3 Система «МоиДокументы-Туризм» 13
1.2.4 Программный комплекс «*ERP.travel*» 15
1.3 Сравнительная характеристика автоматизированных систем 17
1.4 Постановка задачи 17
1.5 Выбор инструментов разработки 18
1.5.1 Выбор системы управления базами данных 18
1.5.2 Выбор движков 18
1.5.3 Безопасность 20
1.5.4 Выбор языков программирования 21
1.6 Разработка требований к автоматизированной системе 22
1.6.1 Назначение разработки 22
1.6.2 Требования к функциональным характеристикам 23
1.6.3 Требования к надёжности 23
1.6.4 Требования к составу и параметрам технических средств 24
1.6.5 Требования к информационной и программной совместимости 24
1.7 Проектирование программного обеспечения 24
1.7.1 Выбор средства проектирования системы 24
1.7.2 Объектно-ориентированное проектирование системы 25
1.8 Проектирование структуры базы данных 38
1.8.1 Логическое проектирование базы данных 38
1.8.2 Разработка структуры связей 42
1.8.3 Нормализация отношений базы данных 43

1.8.4 Физическое проектирование БД 44
1.9 Разработка программного обеспечения 45
1.9.1 Общие сведения 45
1.9.2 Функциональное назначение 45
1.9.3 Регистрация на хостинге 51
1.9.4 Описание программы 55
1.9.5 Описание доступных функций 75
2 Эргономические особенности организации труда на рабочем месте оператора автоматизированной системы 85
2.1 Организации рабочих мест пользователей 85
2.2 Зоны досягаемости 86
2.3 Требования к организации и оборудованию рабочих мест с ПЭВМ пользователей ЭВМ 87
2.4 Системы и виды освещения 88
Заключение 91
Список использованных источников 92
Приложение А – *UML* - диаграммы 94
Приложение Б – Листинг программы 99
Приложение В – Презентация 134

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

АИС – автоматизированная информационная система

БД – база данных

НФ – нормальная форма

ОП – оперативная память

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

ПП – программный продукт

СУБД – система управления базой данных

ЦП – центральный процессор

MVC (Model-View-Controller) – схема разделения данных приложения и управляющей логики на три отдельных компонента

UML (*Unified Modeling Language)* – унифицированный язык моделирования

WPF (*Windows Presentation Foundation)* – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем

Введение

В настоящее время существует тенденция развития сферы услуг как неотъемлемой части жизни человека. Одним из видов таких услуг являются услуги турагентства.

Турагентство - организация, которая занимаются розничной продажей туров от туроператоров.

Туризмом человечество занимается на протяжении многих веков. На современном этапе – это крупный бизнес с многомиллионными финансами, включающий в свой состав не только туроператоров и турагентства, но и крупных перевозчиков воздушным, водным, железнодорожным и авторанспортом. Кроме того, в структуру туристического бизнеса активно интегрированы отели и рестораны в любой точке нашей планеты.

Этот сектор экономики входит в пятерку отраслей с наибольшей долей реализации товаров и услуг через Интернет. В таком виде бизнеса крайне важен правильный выбор поставщиков услуг, основанный на профессиональных знаниях туристского рынка, бизнеса, особенностей и рычагов его развития и управления. Современные компьютерные технологии позволяют организовать работу предприятия с максимальной эффективностью.

Активное внедрение современных компьютерных технологий в деятельность туристических предприятий является необходимым условием их успешной работы, поскольку точность, надежность, оперативность и высокая скорость обработки и передачи информации определяет эффективность управленческих решений в этой сфере. Вопрос комплексной автоматизации процессов бронирования, расчетов с клиентами и гостиницами становится все более актуальным для туристических агентств. Необходимость использования специального программного обеспечения для автоматизации всей внутриофисной работы зависит от объемов, и вида деятельности турфирмы.

Как любое коммерческое предприятие, туристическое агентство направляет свою деятельность на достижение прибыли за счет увеличения числа

клиентов, пользующихся его услугами. В настоящее время в условиях роста числа туристических организаций и нарастающей конкуренции, это становится возможным только за счет оптимизации работы персонала и повышения эффективности обслуживания клиентов. Это достигается благодаря внедрению автоматизированной информационной системы. Основная «обязанность» автоматизированных информационных систем заключается в перекладывании рутинных работ с человека на компьютер, что, в свою очередь, влечет за собой ускорение процессов и повышение эффективности работы персонала.

Из всего вышеизложенного вытекает актуальность бакалаврской работы, которая заключается в повышении эффективности функционирования центра туристического сервиса, а также оптимизации работы его персонала за счет использования автоматизированной информационной системы.

Целью бакалаврской работы является разработка автоматизированной информационной системы центра туристического сервиса, оказывающего услуги по подбору туров, предоставлению путевок и оформлению необходимых документов.

Для достижения целей бакалаврской работы, были определены следующие основные задачи:

– провести анализ предметной области и рынка программного обеспечения в сфере туристических услуг;

– определить основные функции системы и требования к ней;

– выбрать средства проектирования модели системы;

– выполнить проектирование системы с построением схемы базы данных и *UML* – диаграмм;

– разработать программное обеспечение системы, и выполнить ее тестирование;

1 Автоматизация деятельности центра туристического сервиса

1.1 Анализ предметной области

Целью функционирования центра туристического сервиса является создание привлекательных туристских продуктов, соответствующих потребностям туристов, организация продажи туристского продукта в соответствии с современными методами торговли.

В соответствии федеральным законом РФ о туристической деятельности продвижение и реализация туристского продукта осуществляются турагентом на основании договора, заключенного туроператором и турагентом. Турагент осуществляет продвижение и реализацию туристского продукта по поручению туроператора [1].

Роль турагентства в туристской индустрии сводится к следующему функционалу:

– личная консультация туристов по вопросам возможностей отдыха и путешествий по всем имеющимся турам, курортам, туристским центрам и т.д.;

– продвижение (реклама-PR) предложений туроператоров среди туристов;

– продажи туристских услуг/туров с применением современных технологий.

Турагентства подразделяются на:

– многопрофильные (большее число). Ориентируются на облуживание широкого круга туристов, преследующих разные туристские цели;

– специализированные. Осуществляют работу по организации путешествий для однородных по мотивам и целям групп туристов (например, в сфере делового туризма) [2].

Турагент ограничен в ответственности перед туристами. Финансовые гарантии не предусмотрены для турагентов. Основными видами деятельности центра туристского сервиса «Insertum» являются:

– образовательные туры для студентов, магистрантов, аспирантов по профилю обучения;

– образовательные туры для профессионалов разных сфер деятельности;

– промышленные туры и экскурсии;

– изучение иностранного языка в зарубежных языковых школах;

– комбинированные туры (языковые курсы, хобби, спорт, отдых на море);

– изучение иностранного языка и стажировка по специальности;

– программы *Work&Travel, Work&Study*.

На основе анализа функционирования туристической фирмы были выделены следующие бизнес-процессы, которые необходимо автоматизировать:

– учет клиентов;

– учет заказов;

– ведение документооборота;

– учет оплаты туристических заказов;

– информирование сотрудников об авиарейсах и контингенте туристов. Поскольку фирма является небольшой, она не имеет возможности купить дорогостоящий программный продукт.

1.2 Анализ рынка готовых программных продуктов

Для автоматизированной деятельности фирмы в области туристического бизнеса широко используются следующие компьютерные системы:

– «Само-ТурАгент».

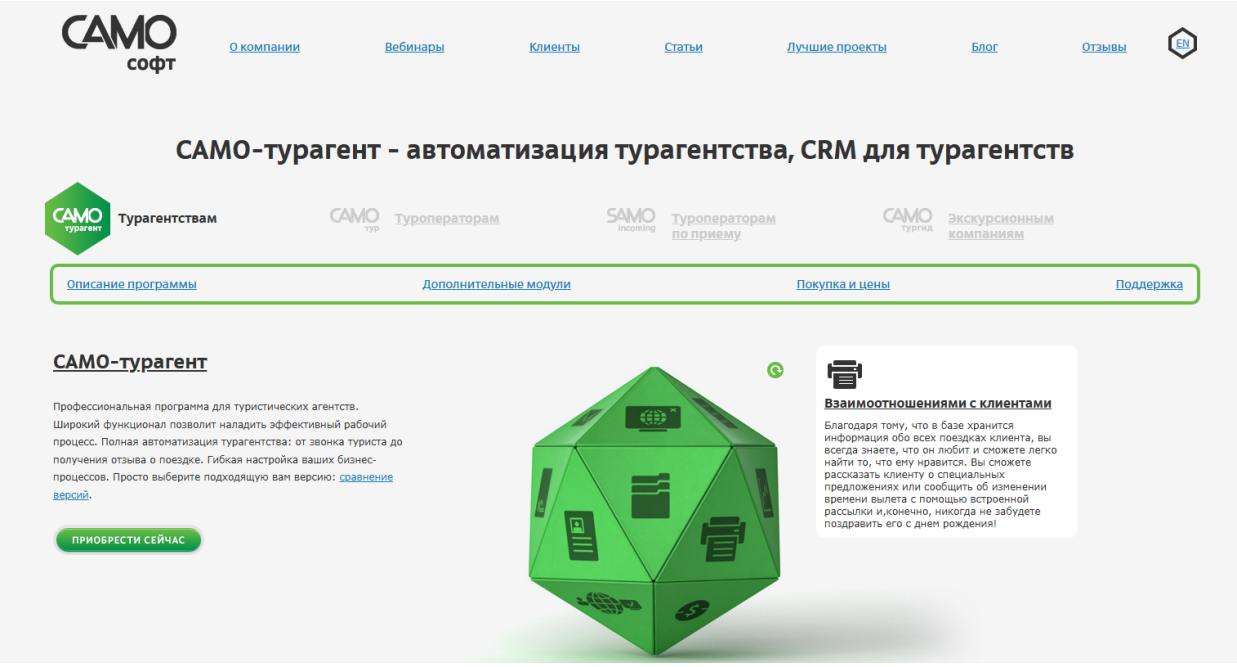
– «*AKAMSOFT* ТУРАГЕНТСТВО »;

– «МоиДокументы-Туризм »;

– «*ERP.travel*».

1.2.1 Система «Само-ТурАгент»

Это программный комплекс для автоматизации турагентства. Программа учитывает все технологические операции деятельности турагентства. Главная



страница системы «Само-ТурАгент» представлена на рисунке 1.1.

Рисунок 1.1 – Главная страница системы «Само-ТурАгент»

Основные возможности программы:

– описание услуг, предоставляемых туристам: гостиниц, транспорта, страхования, визовой поддержки, дополнительных услуг;

– квотирование услуг: номеров гостиниц, мест транспорта и мест тура;

– создание турпакетов из различного набора услуг, расчет прайс-листа и подготовку их к печати;

– оформление заявок туристов;

– подготовка документов туриста: ваучер, билет, страховой полис, анкета для оформления визы, путевка ТУР-1 и др.;

– подготовка заявок партнерам для бронирования и подтверждений покупателям (турагентствам);

– учет платежей покупателей и расчетов с партнерами за предоставленные услуги;

– формирование итоговых отчетов и списков для гостиниц, для авиакомпаний, для посольств, для страховых компаний;

– статистический анализ.

Дополнительные возможности программы:

– импорт цен гостиниц и билетов из таблиц формата MS Excel;

– гибкий механизм ценообразования (комиссия туроператора в процентном или денежном выражении, и их сочетание);

– гибкий механизм назначения комиссионного вознаграждения покупателям;

– визуальный контроль продажи квотируемых услуг;

– контроль взаиморасчетов по каждой заявке;

– сопряжение с почтовыми программами;

– сопряжение с бухгалтерскими программами: "1С:Бухгалтерия 8", «Парус»;

– гибкая система администрирования: назначение прав доступа, как отдельным пользователям, так и группам пользователей, назначение правил контроля менеджеров по продаже, индивидуальные настройки интерфейса каждого пользователя и др.;

– встроенный редактор печатных форм, настраиваемые документы, списки, отчеты в редакторах *MS Word, MS Excel*;

– дружественный, понятный интерфейс пользователя.

1.2.2 Система «*AKAMSOFT* ТУРАГЕНТСТВО»

AKAMSOFT ТУРАГЕНТСТВО — программа-калькулятор туров для автоматизации работы туристического агентства. Главная страница системы «*AKAMSOFT* ТУРАГЕНТСТВО» представлена на рисунке 1.2.

Достоинством является интеграция с офисным пакетом программ *MS Office*, эффективность использования персональных компьютеров в локальной сети повышается в несколько раз.

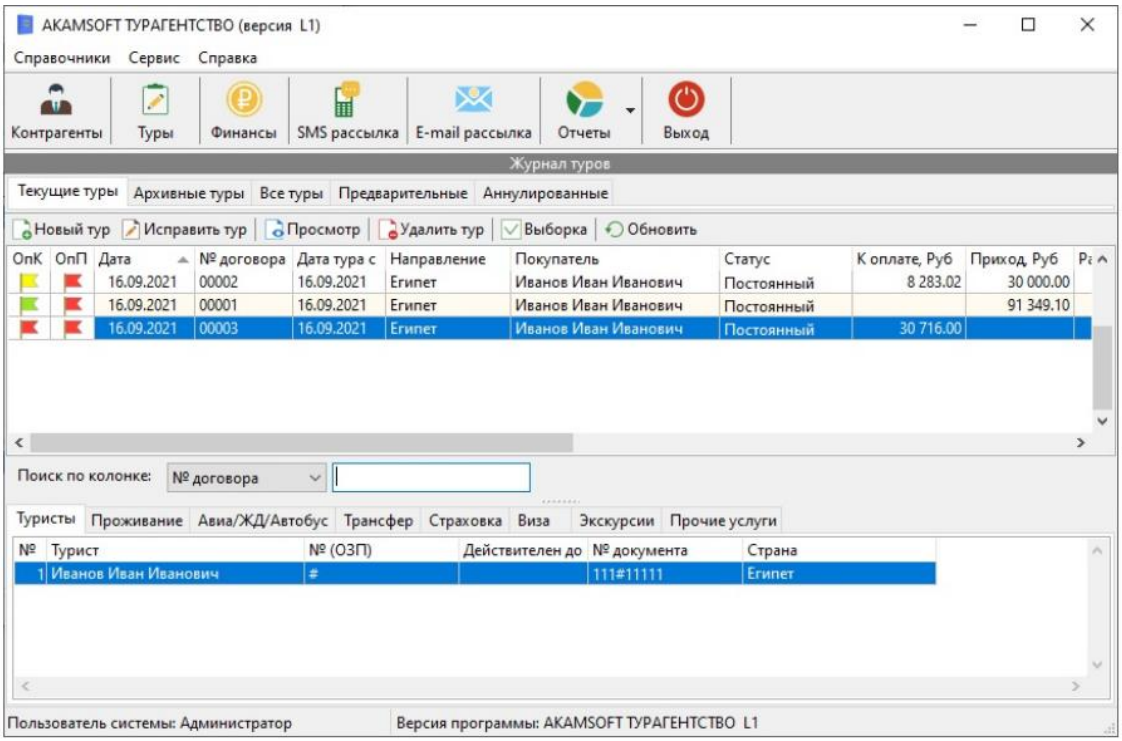


Рисунок 1.2 – Главная страница программы «*AKAMSOFT* ТУРАГЕНТСТВО»

Основные возможности программы:

– легкая бронь;

– формирование и печать договоров, ваучеров и иной сопроводительной документации.

– работа с несколькими валютами;

– удобный поиск сведений в базе;

– возможноть отмечать оплату;

– рассчитать наиболее популярные направления, определить самых доходных работников и провести иную аналитику;

– ведение заявок одновременно несколькими пользователями без потерь производительности;

– анализ результатов деятельности за определенный период и мониторинг действий специалистов в приложении;

– конвертации данных в удобный формат;

– сохранение истории взаимодействия с контрагентами и многое другое.

Возможность выбрать из двух вариантов работы:

– локальная модификация L подойдет для работы одного специалиста на конкретном ПК;

– сетевая N одновременного использования несколькими сотрудниками с разных компьютеров в локальной сети.

1.2.3 Система «МоиДокументы-Туризм»

МоиДокументы-Туризм умная программа для автоматизации турфирм и туристических агентств.

Основные функции программы:

– единая база туристов и централизованный контроль для сети офисов;

– работа с программой из офиса, дома, в отпуске и командировке;

– модуль помощи для ускоренного бронирования на сайтах туроператоров;

– полный комплект документов для туриста, индивидуально для каждой компании любые бланки документов;

– актуальная информация и фингарантии по всем туроператорам России;

– предварительные заявки и заявки, отложенные по сроку вылета;

– детальная информация по каждому туру;

– контроль взаиморасчетов с туристами и туроператорами;

– единый календарь с задачами и напоминаниями на каждый день;

– фиксирование всех этапов взаимодействия с туристами (Интеграция с IP-телефонией, рекламные источники и комментарии);

– *MS* и *E-mail* рассылка туристам (поздравление в день рождения, напоминание о вылете или окончании действия загранпаспорта);

– личный кабинет туриста (информация по туру, контактные данные менеджера, онлайн отплата тура картами *Visa, Mastercard*);

– дисконтные и бонусные карты клиентов;

– статистика и отчеты, с формированием любого отчета по заданным параметрам;

– контроль качества работы менеджера, начисление заработной платы;

– мини-бухгалтерия для отслеживания финансовых операций;

– гибкая система настройки прав доступа для сотрудников компании;

– интеграция с онлайн кассами фирмы АТОЛ и Эвотор;

– интеграция с Вконтакте;

– мобильное приложение для *iOS* и *Android*.

Она интегрирует все необходимые бизнес-процессы и средства коммуникации, обеспечивающие информационный обмен между этими основными участниками туристического рынка:

– механизм квотирования мест в самолетах, номеров в гостиницах;

– гибкий инструмент создания тура как комплексного продукта с системой скидок, комиссий, вариантов перевозки и размещения, условий бронирования и оплаты;

– удобный дисплей наличия мест по ресурсам (для туроператора) и по заявке (для турагента);

– сочетание "он-лайн" бронирования и бронирования по запросу в одном интерфейсе, формирование прайс-листов, экспорт отчетов в *WORD, EXCEL*.

Рабочая область «МоиДокументы-Туризм» представлена на рисунке 1.3.

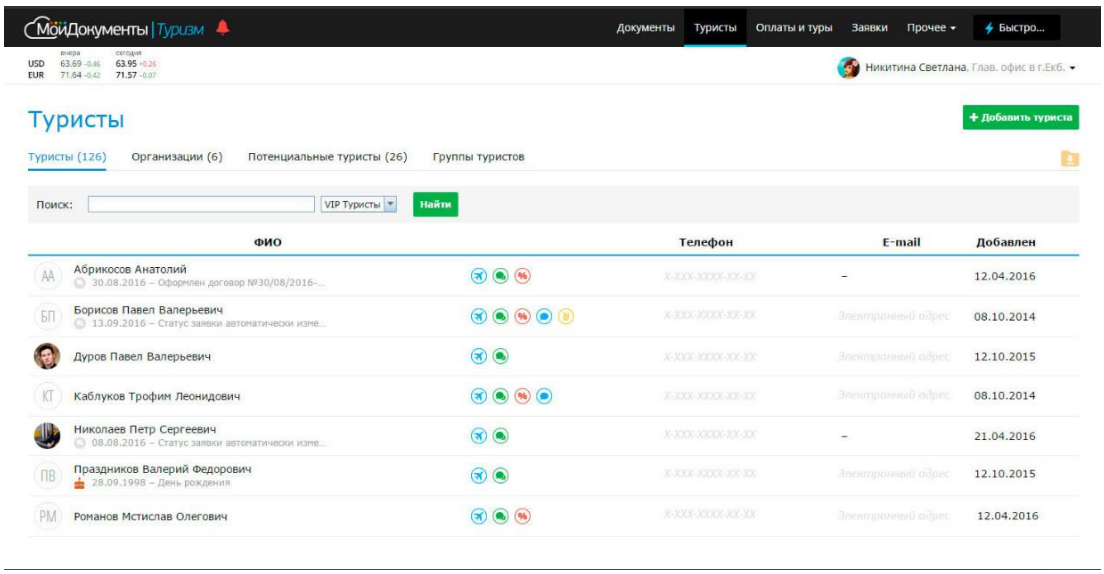


Рисунок 1.3 –Страница программы «МоиДокументы-Туризм»

Основные возможности программы:

– масштабирование, технология балансировки нагрузки и распараллеливания бизнес-процессов;

– современный стандарт защищенных транзакций SSL при передаче коммерческой информации;

– универсальный формат обмена данными *XML*;

– технология работы в *WIN32* клиентском приложении (функции туроператора и турагента) и под браузером (функции турагента);

– трехуровневая архитектура системы для обработки поступающих запросов;

– технологии взаиморасчетов через Интернет (в том числе по кредитным карточкам);

– технология автоматического комплексного бронирования заявки - 24 часа доступа к турам.

1.2.4 Программный комплекс «*ERP.travel*»

Предназначен для автоматизации деятельности туристического агентства. Главная страница системы «ERP.travel» представлена на рисунке 1.4.

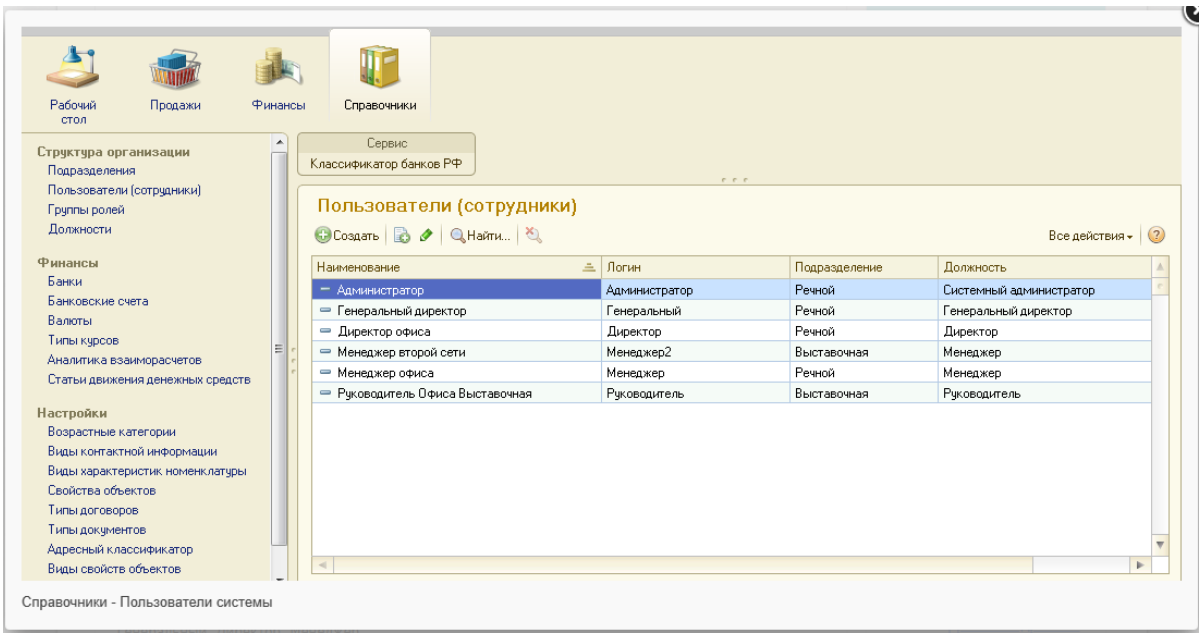


Рисунок 1.4 – Главная страница программы «*ERP.travel*»

ERP.travel - это профессиональная онлайн система для турагентств. Система максимально автоматизирует процесс работы с заказом тура, включая печать всех документов. В системе самое быстрое время оформления заказа тура - всего 4,5 минуты. При этом в системе и самый большой функционал: путевки и заявки туроператорам; оформление индивидуальных туров; приходы и расходы по кассе и банку; косвенные расходы; управленческий учет; отчетность и интеграция с сайтами.

Наибольшие преимущества в системе для сетей турагентств, поскольку только в ней можно организовать работу всех офисов в одной базе. С системой можно работать с ноутбуков и планшетов, из офиса, дома и кафе. В системе автоматизации турагентств *ERP.travel* реализована гибкая система прав доступа турагентам, позволяющая в одной базе вести учет даже не одной, а нескольких сетей турагентств.

В системе *ERP.travel* полностью проработан модуль *CRM* для турагентства и включает в себя: запросы на тур, конверсия в заказы, воронка продаж, массовые рассылки по *e-mail* и *sms*, получение почты прямо в программе, отправка *sms* и *e-mail* из любого места в программе.

Система предлагается агентствам для автоматизации своего учета в современном формате облачного сервиса (*SaaS*).

1.3 Сравнительная характеристика автоматизированных систем

Исследование функциональных возможностей готовых программных продуктов показало, что каждый представленный программный продукт обладает массой функциональных возможностей, каждый из них имеет специфику своей работы, хороший уровень используемых технологий, достаточно высокая масштабируемость.

Но поскольку эти программы рассчитаны на крупные фирмы, имеющие множество территориально распределенных филиалов, функций в системах реализовано гораздо больше, чем необходимо для ООО «Центр туристского сервиса *Insertum*». Высокая стоимость готовых решений, затраты денег и времени на обучение персонала, дорогая эксплуатация является основанием для принятия решения о целесообразности разработатки своего программного продукта.

1.4 Постановка задачи

На сегодняшний день процесс регистрации заявок на туры от клиентов (юридических или физических лиц) частично автоматизирован с помощью пакета офисных программ *Microsoft Office*. Большая часть документов оформляется на бумажных носителях, что уменьшает эффективность работы. Вследствие чего возникает необходимость разработки автоматизированной системы туристического агентства. Таким образом, для решения данной проблемы, исходя из анализа предметной области, была поставлена задача автоматизации организации путем разработки программы, функционирующей как *web*-приложение.

Автоматизация приведет к более эффективному учету регистрации заявок клиентов, а также к более удобному и быстрому подбору необходимого тура и составлению сопроводительных документов. Благодаря внедрению

данной системы в организации штат сотрудников уменьшится, что приведет к снижению экономических издержек, также будут уменьшены затраты на бумажные носители, так как будет налажен электронный документооборот. Вся информация о клиентах, турагентах, турах, странах и отелях, договорах будет храниться и обновляться в единой базе, таким образом, информация сохранит свою целостность и актуальность. Для работы автоматизированной системы необходимо выбрать СУБД, а также средства для ее проектирования.

1.5 Выбор инструментов разработки

1.5.1 Выбор системы управления базами данных

MySQL — это популярная система управления базами данных. Первая версия этой СУБД была выпущена в 1994 году компанией *MySQL AB*.

MySQL обладает рядом преимуществ, что позволяет использовать ее для реализации данного веб – приложения:

– производительность. Для веб-приложений важна скорость работы. *MySQL* – это современная высокопроизводительная СУБД;

– кроссплатформенность. *MySQL* предоставляет кроссплатформенную базу данных: она работает на *Linux*, *FreeBSD* и конечно на *Windows*. Данное свойство СУБД является неоспоримым преимуществом при разработке интернет приложений любого назначения.

MySQL входит в состав популярных стеков для разработки веб-приложений — *LAMP (Linux, Apache, MySQL, PHP)* и *WAMP (Windows, Apache, MySQL, PHP)*.

1.5.2 Выбор движков

Движок базы данных (*database engine*) — это программный модуль, с помощью которого СУБД создает, читает и обновляет данные в базе. В *MySQL* есть два типа движков: транзакционные и нетранзакционные. Транзакционные движки хранения данных подразумевают, что операции в этих БД могут быть отменены, если они не завершены. Всего *MySQL* поддерживает 8 движков:

InnoDB — это распространенный движок, который используется в *MySQL* по умолчанию. Он поддерживает *ACID*, блокировку на уровне строк, восстановление после сбоев и многопоточность. Также это единственный движок, который обеспечивает ограничение ссылочной целостности внешнего ключа. *MyISAM* — этот движок использовался по умолчанию до появления *InnoDB*. И в отличие от него, *MyISAM* не поддерживает *ACID*. В этом движке реализована блокировка только на уровне таблиц, поэтому он небезопасен для транзакций. В тоже время *MyISAM* оптимизирован для чтения. Сфера применения *MyISAM* — это системы хранения с минимальными транзакционными данными. *Memory* — этот движок реализуется хранение данных в памяти в виде кучи. Такая система обеспечивает очень быстрый доступ к данным. *Memory* поддерживает блокировку на уровне таблиц и хранит данные в оперативной памяти — при перезагрузке сервера они теряются. Этот движок подходит для временных таблиц.

CSV — этот движок хранит данные в файлах формата *.csv*. Не поддерживает индексирование и разделение. *CSV* используется для обмена данными между различными программами или приложениями.

Merge — движок работает с таблицами *MyISAM*. Он логически группирует ряд идентичных таблиц *MyISAM* и ссылается на них, как на один объект. *Merge* упрощает управление большими объемами данных и используется в системах хранения данных.

Archive — этот движок оптимизирован для высокоскоростной вставки. При вставке он сжимает данные. *Archive* не поддерживает транзакции, операции *DELELE* и *UPDATE*, но зато поддерживает блокировку на уровне строк. Отлично подходит для хранения больших объемов архивных данных, на которые редко ссылаются. *Federated* используется создания одной БД на нескольких физических серверах.

Подходит для распределенных систем. Движок *Blackhole* принимает данные, но не сохраняет их. На первый взгляд этот движок бесполезен, но на самом деле он находит применение при репликации данных и тестировании. *Blackhole* можно использовать в качестве фильтра между серверами.

1.5.3 Безопасность.

MySQL предлагает пользователям высокий уровень безопасности. В *MySQL* есть встроенные инструменты безопасности, которые поддерживают управление пользователями и их привилегиями. При недостатке стандартных инструментов пользователь всегда может установить дополнительные плагины.

В качестве недостатков для *MySQL* можно указать следующее:

– при работе с малыми и средними приложениями *MySQL* работает быстро, но при увеличении объемов информации у *MySQL* возникают проблемы. Речь идет о базах данных с миллионами строк. Такие БД тяжело масштабировать, а сложные запросы к ним начинают зависать и иногда не успевают выполняться в допустимое время ожидания;

– *MySQL* не соответствует стандарту *SQL*: СУБД не поддерживает некоторые функции и имеет расширения, не относящиеся к стандартному *SQL*. Для большинства проектов это не станет серьезной проблемой, но при миграции на другие СУБД этот момент может всплыть. *MySQL* достаточно универсальная система управления базами данных. При выборе *MySQL* в качестве СУБД нужно отталкиваться от требований проекта.

В целом, *MySQL* находит применение в:

– малых и средних веб-приложениях;

– проектах с требованиями к безопасности;

– в случаях, когда встроенных систем управления базами данных (например *SQLite*) не хватает, а продукты уровня *Oracle Database* или *SQL Server* избыточны;

– проектах, подразумевающих производительность [3].

1.5.4 Выбор языков программирования

Для разработки автоматизированной информационной системы центра туристического агентства используются следующие языки программирования: *JavaScript,* язык разметки *HTML, CSS, PHP* (фреймворк *Yii2*).

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации *ECMAScript* (стандарт *ECMA-262*).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

HTML (от английского *HyperText Markup Language*) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

После перехода на требуемую страницу сайта браузер подгружает *HTML*-файл с информацией о структуре и контенте веб-страницы. *HTML* как бы выстраивает визуальный фундамент сайта, но не «запускает» сайт самостоятельно. Он всего лишь указывает, где располагаются элементы, какой у них будет базовый дизайн, откуда брать стили для элементов и скрипты (обычно их пишут на *JavaScript*).

CSS — формальный язык декорирования и описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего *HTML* или *XHTML*). Также может применяться к любым *XML*-документам, например, к *SVG* или *XUL*. *CSS* используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки *CSS* является ограждение и отделение описания

логической структуры веб-страницы (которое производится с помощью *HTML* или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка *CSS*). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом [4].

PHP создали специально для разработки веб-страниц. На нём можно делать ресурсы любой сложности: от простого лендинга до социальной сети. Код на *PHP* легко встраивается в классический *HTML* — нужно всего лишь добавить соответствующий тег. Язык поддерживают все популярные операционные системы: *Windows, macOS, Linux, UNIX* и не только.

PHP позволяет работать с такими веб-серверами, как *IIS* в *Windows* и *Apach*e в *macOS* и *Linux*. Благодаря такой широкой совместимости у разработчиков практически нет ограничений в выборе веб-сервера и операционной системы.

Yii2 — это компонентный объектно-ориентированный фреймворк для веб-разработки на *PHP*, реализующий парадигму *MVC (Model-View-Controller).* Его название расшифровывается как *Yes it is. Yii2* — вторая версия фреймворка *Yii* [5].

Таким образом, все описанные выше преимущества позволили выбрать язык программирования *PHP 7.4* (фреймворк *Yii2*), стандартизированный язык гипертекстовой разметки документов *HTML, Java Script, CSS,* СУБД - *MySQL*.

1.6 Разработка требований к автоматизированной системе

1.6.1 Назначение разработки

Программа предназначена для автоматизации деятельности сотрудников турагентства.

Целью разработки является решение следующих задач: автоматизация процесса учета клиентов, формирование и учет заказов, ведение отчетности и

учет платежей. Это позволит повысить производительность труда сотрудников и уменьшить расход времени на формирование отчетности.

1.6.2 Требования к функциональным характеристикам

Автоматизированная информационная система центра туристического агентства должна обеспечивать возможность выполнения следующих функций:

– поддержка авторизации пользователей системы;

– учет сведений о клиентах;

– учет сведений о заявках;

– учет оплаты заказов;

– учет сведений о турах (видах, направлениях туров, типах туров, категориях туров);

– учет сведений об услугах по туру;

– учет сведениий о спецпредложениях;

– учет сведениий о туроператорах туров;

– ведение журнала договоров (контрактов);

– учет оплаты за тура (оплата, тип оплаты, назначение оплаты).

Требования к интерфейсу. Интерфейс приложения должен быть интуитивно понятным. Он должен обеспечивать возможность эффективной работы пользователя, владеющего лишь самыми минимальными навыками работы с компьютером.

1.6.3 Требования к надёжности

Основным требованием к разрабатываемой системе является корректное выполнение возложенных на нее функций.

Программа должна корректно реагировать на удаление, изменение данных, т.е. обеспечивать целостность данных. В случаях неправильных действий со стороны пользователя программа должна выдавать соответствующие предупреждения.

Программа должна выдавать сообщения о возникающих ошибках, при неверном задании исходных данных.

Система должна обеспечивать одновременную работу не менее 5 сотрудников.

1.6.4 Требования к составу и параметрам технических средств

Минимальные технические средства:

– тактовая частота процессора – не менее 2,5 МГц;

– оперативная память – не менее 4096 Мбайт;

– объем памяти жестокого диска должен быть не менее 256 Гбайт;

– видеоадаптер с минимальным разрешением 720*p*.

К рабочим станциям должны быть подключены стандартные мышь, клавиатура и принтер.

1.6.5 Требования к информационной и программной совместимости

Программа должна быть спроектирована с использованием *CASE*-средства *StarUML*.

Программа должна быть реализована на языке программирования *PHP* 7.4 (фреймворк *Yii2*), стандартизированного языка гипертекстовой разметки документов *HTML, Java Script, CSS*.

Программа должна работать в операционных системах *MS Windows 10* и выше с фреймворком *Microsoft .NET Framework 4* и поддерживать кроссплатформенную работу с браузерами *IE* в *Microsoft Edge, Google Chrome*.

На сервере должны быть установлены и запущены компоненты системы управления базой данных (СУБД) *MySQL 8.0.32*.

1.7 Проектирование программного обеспечения

1.7.1 Выбор средства проектирования системы

Проектирование программного обеспечения можно осуществить с помощью диаграмм, тем самым визуализировать систему с различных точек зрения. Сложную систему можно представить в виде набора независимых диаграмм, причем ни одна из них не является достаточной для описания системы и

получения полного представления о ней, поскольку каждая из них фокусируется на каком-то определенном аспекте функционирования системы и выражает разный уровень абстракции. Другими словами, каждая диаграмма соответствует некоторой определенной, частной точке зрения на проектируемую систему.

Проектирование функциональной модели конфигурации производится при помощи *CASE*-средства *StarUML*.

StarUML - программный инструмент моделирования, который поддерживает *UML* (Унифицированный язык моделирования). *StarUML* поддерживает одиннадцать различных типов диаграмм. Среда разработки *StarUML* превосходно настраивается в соответствии с требованиями пользователя и имеет высокую степень расширяемости, особенно в области своих функциональных возможностей. Использование *StarUML*, одного из ведущих программных инструментов моделирования, гарантирует достижение максимальной производительности и качества программных проектов.

StarUML предоставляет максимальную степень адаптации среды разработки пользователя, предлагая настройку параметров, которые могут влиять на методологию разработки программного обеспечения, проектную платформу и язык.

В отличие от многих существующих программ, которые используют собственные неэффективные форматы файла модели, *StarUML* оперирует файлами в стандартном формате *XML*. Коды, написанные в легких для чтения структурах и форматах, могут быть легко изменены с помощью синтаксического анализатора *XML* [6].

1.7.2 Объектно-ориентированное проектирование системы

Для создания модели будущей систем, руководствуясь стандартом *UML*, необходимо рассмотреть три наиболее распространенных представления:

– представление условий использования, сконцентрированное на описании функциональности и поведения системы;

– логическое представление, описывающее структуру системы;

– компонентное представление, описывающее, как логические элементы «вписаны» в конечный продукт [7].

В следующих подразделах будут рассмотрены каждое из этих представлений, также будет рассмотрена организация базы данных:

– Диаграмма вариантов использования.

Диаграмма вариантов использования, на которой изображаются актеры и варианты использования, между которыми существуют отношения. Здесь можно показывать и другие элементы *UML*.

Для реализации проекта необходимо разработать 2 диаграммы вариантов использования:

– для актера-администратора, которому доступна вся функциональность системы;

– для актера-сотрудника туристического агентства (турагент).

Актер-администратор выполняет следующие функции:

– управление информацией о сотрудниках;

– управление информацией о заказах, включающее добавление нового заказа, внесение изменений в заказах, поиск заказов, печать документов;

– управление информацией о клиентах, включающее добавление нового клиента, внесение изменений о клиентах, поиск клиентов;

– указание спецпредложений;

– учет туров (направление, категория тура, тип тура и пр.);

– учет оплаты;

– оформление договоров;

– ведение отчетности.

Диаграммы вариантов использования для администратора и сотрудника приведены на рисунке 1.5.

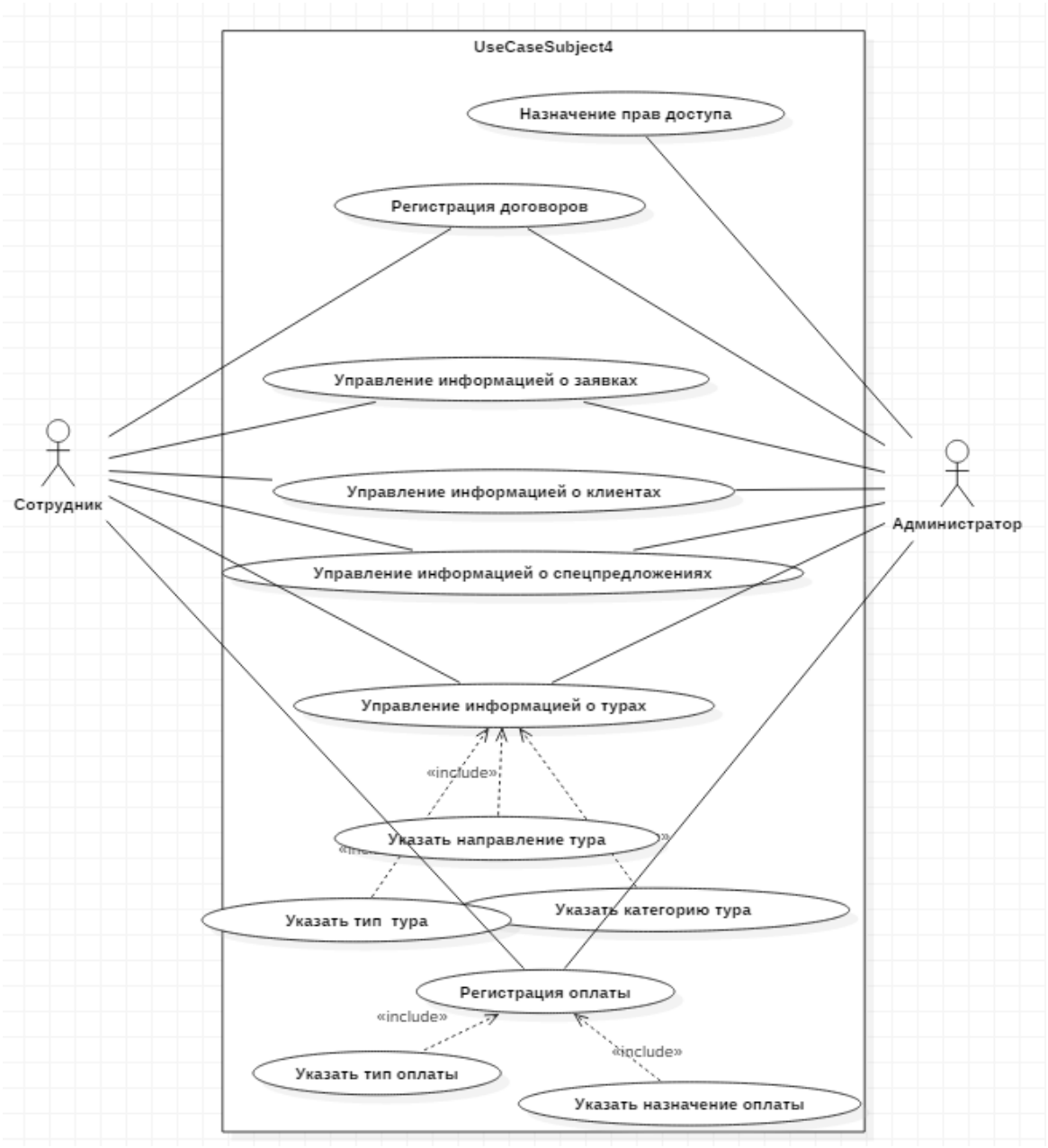


Рисунок 1.5 - Диаграмма вариантов использования

Актер-сотрудник выполняет следующие функции:

– управление информацией о заказах, включающее добавление нового заказа, внесение изменений в заказах, поиск заказов, печать документов;

– управление информацией о клиентах, включающее добавление нового клиента, внесение изменений о клиентах, поиск клиентов;

– указание спецпредложений;

– учет оплаты;

– оформление договоров;

– учет туров (направление, категория тура, тип тура и пр.);

– ведение отчетности.

– Диаграмма деятельности.

Диаграмма деятельности - это диаграмма, описывающая последовательность подобных деятельностей, позволяя при этом одновременно изображать как условное, так и параллельное поведение. Соединение имеет несколько входящих переходов и единственный выходящий переход. Соединение означает окончание условного поведения, которое было начато соответствующим ветвлением [7].

Диаграмма деятельности редактирования заказа представлена на рисунке 1.6.

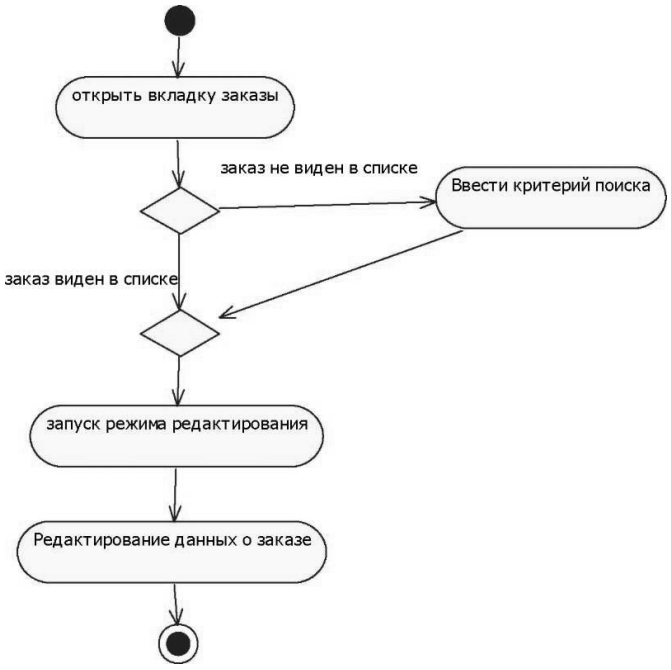


Рисунок 1.6 - Диаграмма деятельности: редактирование заказа

Условное поведение изображается с помощью ветвлений и соединений.

Ветвление имеет единственный входящий переход и несколько выходящих переходов со сторожевыми условиями.

Диаграмма деятельности редактирования заказа. Диаграмма деятельности создания заказов представлена на рисунке 1.7.

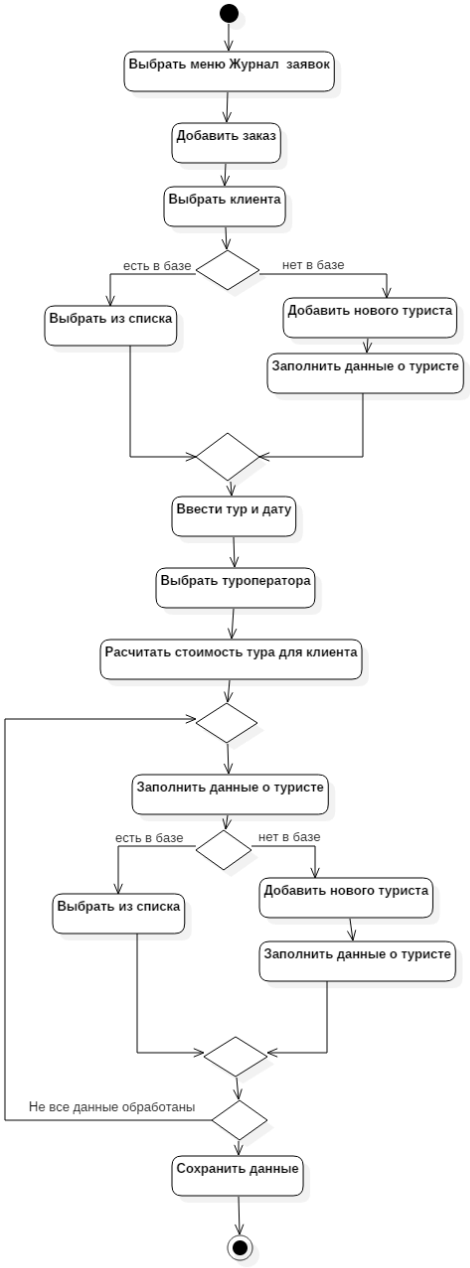


Рисунок 1.7 - Диаграмма деятельности: добавление заказа

Процесс начинается с открытия вкладки заказы, где затем мы находим необходимый заказ и открываем окно редактирования с помощью двойного нажатия.

Процесс создания заказов начинается с открытия вкладки заказы, где затем мы делаем добавление заказа: выбираем заказчика, маршрут, дата тура, туроператора, стоимость тура для клиента и туристов.

Диаграмма деятельности по выводу на печать заказа начинается с процесса открытия вкладки заказы, где затем мы выбираем заказ в списке и выводим на печать.

Диаграмма деятельности по выводу на печать заказа представлена на рисунке 1.8.

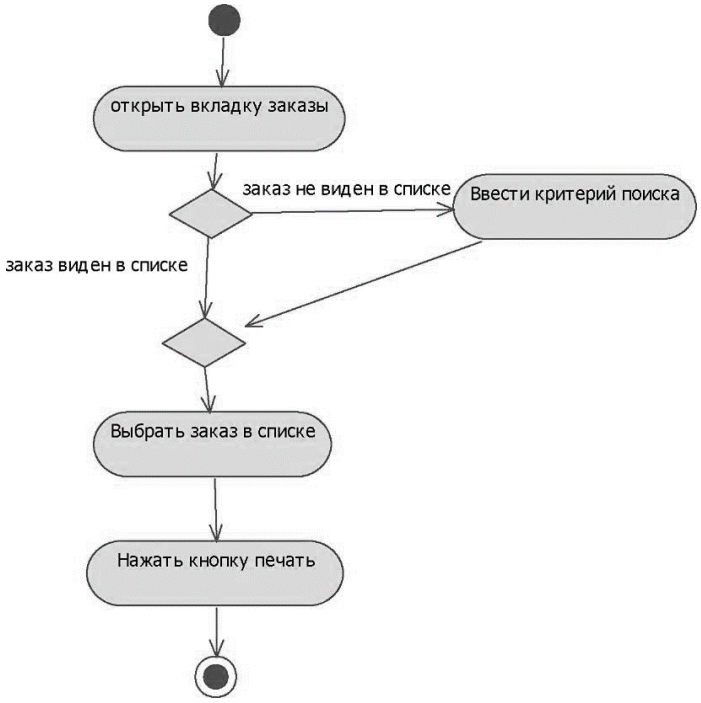


Рисунок 1.8 - Диаграмма деятельности: вывод документа на печать

– Диаграмма последовательности.

Диаграмма последовательности — это диаграмма, на которой отображается множество сообщений между объектами, упорядоченные по временной оси. На диаграмме последовательности объекты изображаются прямоугольниками на вершине вертикальной пунктирной линии. Вертикальная линия называется линией жизни объекта. Она представляет собой жизненный цикл объекта в процессе взаимодействия [7].

В бакалаврской работе представлена диаграмма последовательности работы сотрудника. Объектами являются: сотрудник, интерфейс заказа, база данных.

Последовательность действий состоит в следующем:

– сотрудник регистрируется в системе;

– регистрационные данные сохраняются в базе данных;

– данные сохраняются в интерфейсе заказа;

– пользователь зарегистрирован;

– сотрудник делает запрос на добавление заказа к интерфейсу заказа;

– форма создана и показывается сотруднику;

– сотрудник заполняет форму добавления заказа;

– интерфейс заказа обращается к базе данных для добавления информации о заказах;

– база данных подтверждает добавление информации в интерфейсе заказа;

– сотрудник видит добавленный заказ в интерфейсе заказа;

– сотрудник делает запрос на поиск заказа;

– интерфейс заказа обращается к базе данных с поиском заказа;

– база данных выдает данные о заказе;

– сотрудник видит найденный заказ;

– сотрудник подготавливает документы на печать в интерфейсе заказа;

– интерфейс заказа делает запрос на получение информации к базе данных;

– интерфейс заказа получает информацию от базы данных;

– сотрудник получает подготовленный документ на печать;

– сотрудник выходит из программы;

– сеанс работы с программой завершен.

Диаграмма последовательности работы сотрудника представлена на рисунке 1.9.

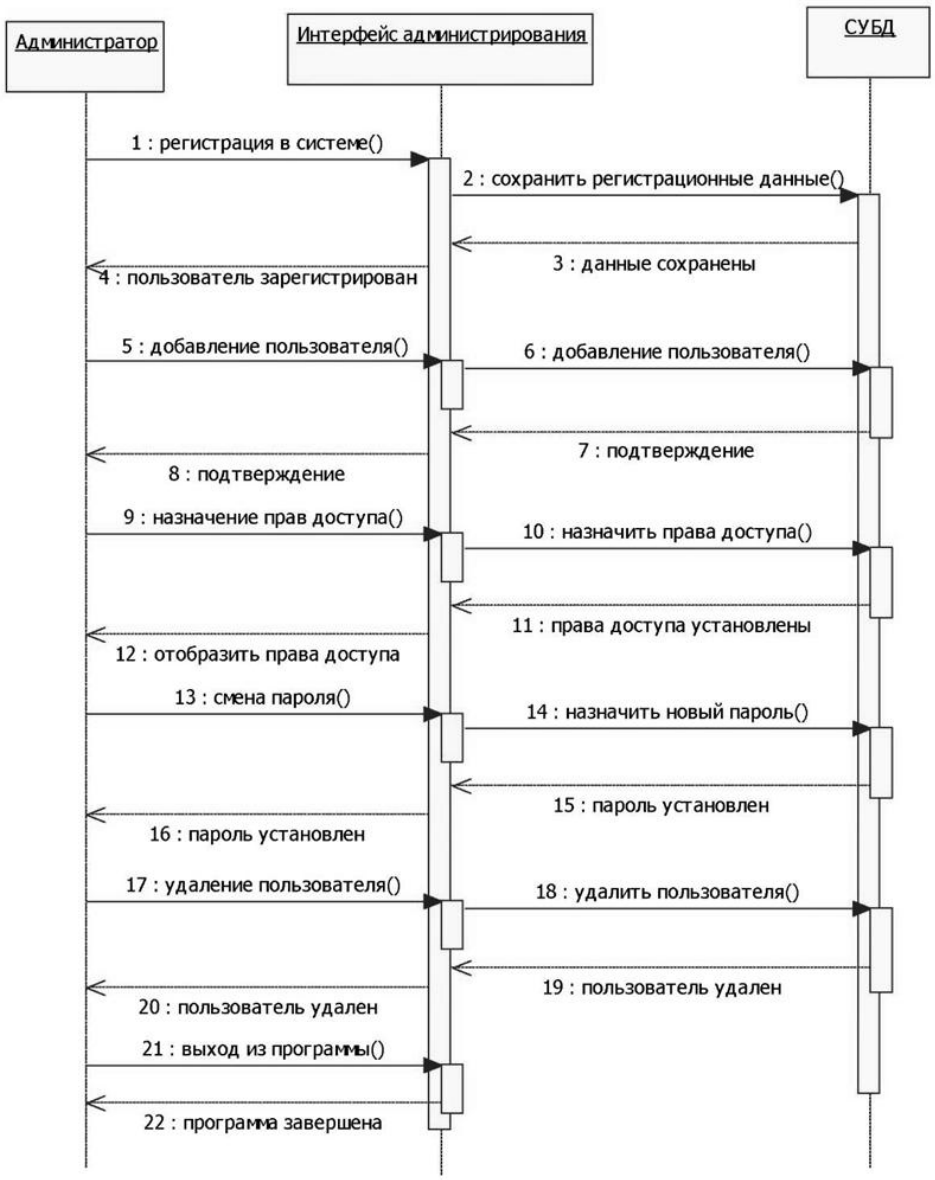


Рисунок 1.9 - Диаграмма последовательности: работа администратора

Следующая диаграмма последовательности - диаграмма работы администратора. Объектами являются: администрирование, интерфейс администрирования, база данных. Последовательность действий следующая:

– администратор регистрируется в системе;

– регистрационные данные сохраняются в базе данных;

– данные сохраняются в интерфейсе заказа;

– пользователь зарегистрирован;

– администратор нажимает на добавление пользователя интерфейсе администрирования;

– добавление пользователя происходит в базе данных;

– база данных подтверждает добавление нового пользователя;

– администратор видит добавленного пользователя;

– администратор нажимает на кнопку назначение прав доступа сотрудникам;

– назначение прав доступа идет в базу данных;

– права доступа установлены в интерфейсе администрирования;

– администратор видит подтвержденные права доступа;

– администратор нажимает на смену пароля в интерфейсе администрирования;

– интерфейс администрирования назначает новый пароль в базу данных;

– база данных подтверждает смену пароля;

– администратор проинформирован, что пароль установлен;

– администратор нажимает на удаление пользователя в интерфейсе администрирования;

– происходит удаление пользователя из базы данных;

– база данных подтверждает удаление пользователя;

– администратор видит удаление пользователя;

– администратор выходит из программы;

– сеанс работы с программой завершен.

Диаграмма последовательности работы администратора представлена на рисунке 1.10.

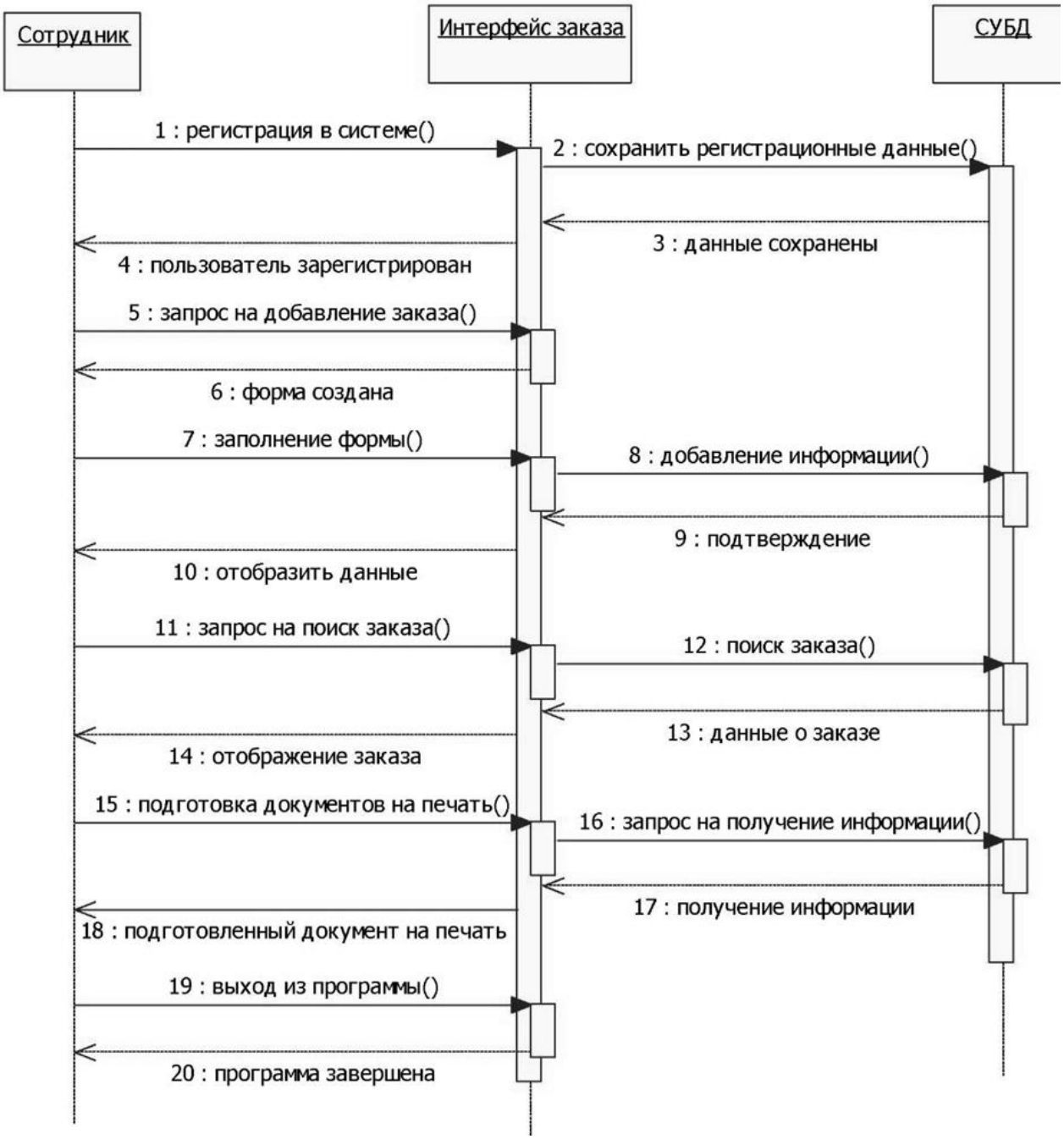


Рисунок 1.10 - Диаграмма последовательности: работа сотрудника

– Диаграмма классов.

Диаграммы классов используются при моделировании проектируемой системы наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов, изображенных на

ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Класс — это основной строительный блок проектируемой системы. Это понятие присутствует и в объектно-ориентированных языках программирования, то есть между классами *UML* и программными классами есть соответствие, являющееся основой для автоматической генерации программных кодов или для выполнения реинжиниринга. Каждый класс имеет название, атрибуты и операции. Класс на диаграмме показывается в виде прямоугольника, разделенного на 3 области. В верхней содержится название класса, в средней - описание атрибутов (свойств), в нижней - названия операций - услуг, предоставляемых объектами этого класса [7].

Все классы находятся в отношении ассоциации. Центральным компонентом программы является окно *MainWindow*, которое создается при инициализации приложений *App*. На окне *MainWindow* располагаются вкладки *Tour*, *Payment*, *Client*, *Administration*. *Administration* позволяет выполнять функции администрирования: добавление пользователя, удаление пользователя, смена пароля, изменение прав доступа. *Client* позволяет добавлять клиентов и находить существующих клиентов по имени. *Payment* позволяет добавлять новые оплаты, искать существующие оплаты и подготавливать документы для печати. *Home* - это домашняя закладка, на которой собраны сведения, необходимые пользователю при начале сеанса работы с программой. *ClientWindow* - окно для редактирования сведений о клиенте, которое вызывается с закладки *Clients*, который обращается к *MainWindows* при поиске всех заказов, связанных с выбранным пользователем.

OrderWindow - окно для редактирования сведений о заказе, которое вызывается с закладки *Orders*.

Для выбора клиентов и операторов был реализован элемент управления *SelectControl* и его наследники *ClientSelectControl* и *OperatorSelectControl*. Они позволяют отображать и выбирать клиентов и операторов из доступных.

TouragencyDbTools - вспомогательный класс, осуществляющий соединение с базой данных, отправку *SQL* запросов и получение результатов, а также авторизацию пользователя при первом обращении к базе данных.

Диаграмма классов представлена на рисунке 1.11.

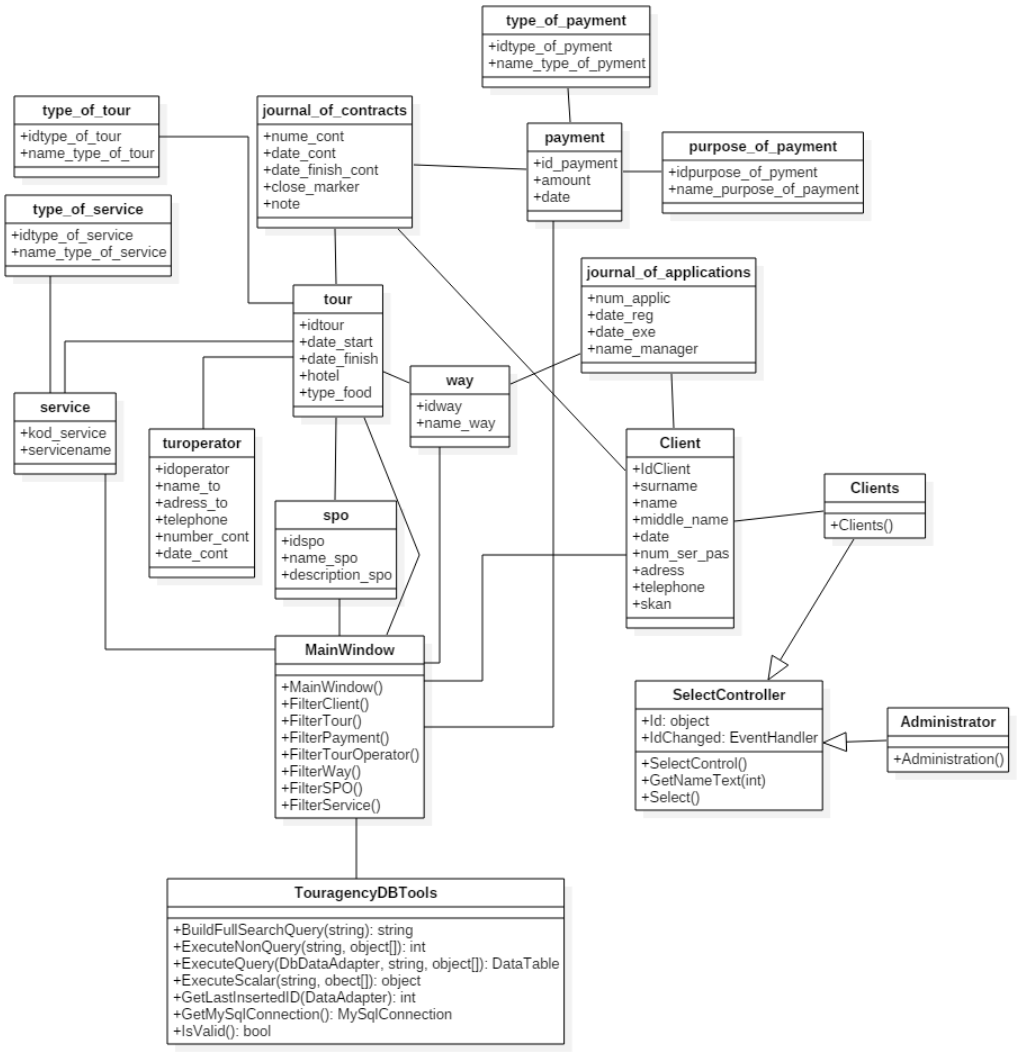


Рисунок 1.11 - Диаграмма классов

– Диаграмма развертывания.

В контексте *UML*, диаграмма развертывания входит в семейство структурных диаграмм, так как описывает один из аспектов самой системы, а именно — физическое развертывание информации, выработанной программой

на аппаратных компонентах. В отношении информации, которую генерирует программа, применяется термин «артефакт».

В состав диаграммы развертывания входит несколько типов фигур *UML*. Трехмерные блоки, или узлы, символизируют базовые программные или аппаратные элементы системы.

Линии, идущие от одного узла к другому, применяются для обозначения связей, а фигуры поменьше, расположенные внутри блоков, — для программных артефактов, которые развертываются на узлах [7].

Диаграммы развертывания используются в разных целях. Например, с их помощью можно: наглядно показать, какие программные элементы развертываются на тех или иных аппаратных компонентах проиллюстрировать обработку процессов исполнения аппаратными компонентами провести обзор топологии аппаратного комплекса. Компьютеры имеют доступ по локальной сети к серверу *MySQL*, а также к принтеру.

С помощью программы *Touragency*, сотрудники работают с заказами и обслуживают клиентов.

Диаграмма развертывания представлена на рисунке 1.12

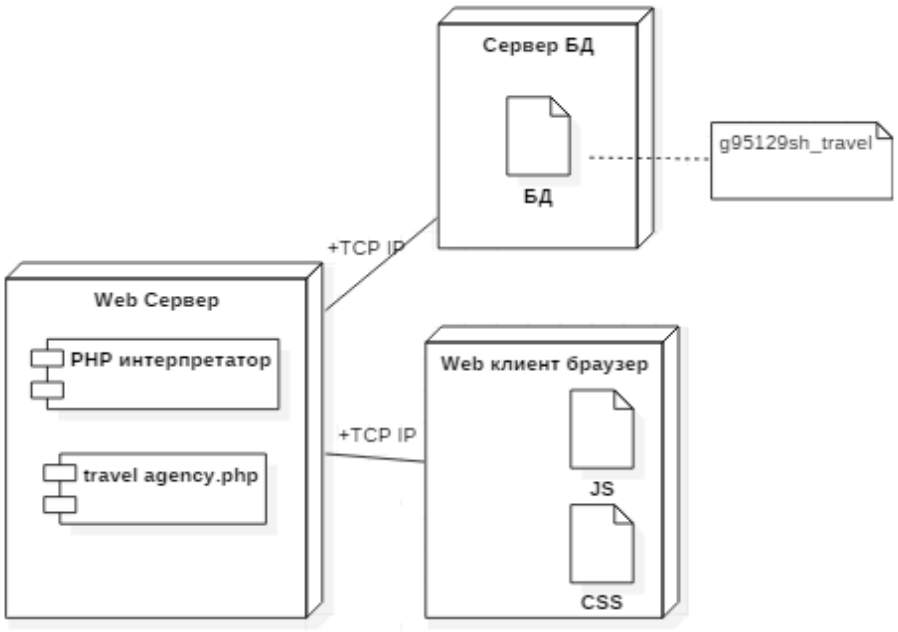


Рисунок 1.12 - Диаграмма развертывания

1.8 Проектирование структуры базы данных

1.8.1 Логическое проектирование базы данных

В результате анализа предметной области было выделено 16 структур данных, представленных в таблице 1.1 - 1.16.

Таблица 1.1 - Структура данных «Клиент»

Наименование поля	Описание применения
ID_ клиента	Номер клиента
Фамилия	Фамилия
Имя	Имя клиента
Отчество	Отчество клиента
Дата рождения	Дата рождения клиента
Паспорт	Реквизиты паспортных данных
Местопроживание	Местопроживание клиента
Телефон	Номер телефона
Электронная копия паспорта	Электронная копия паспорта

.

2 - Структура данных «Вид услуги»

Наименование поля	Описание применения
ID_вида_услуги	Номер вида услуги
Наименование вида услуги	Наименование вида услуги
Примечание	Примечание (описание)

Таблица 1.3 - Структура данных «Журнал заявок»

Наименование поля	Описание применения
ID_записи	Номер записи
ID_направления	Идентификатор направления путешествия
ID_ клиента	Номер клиента

Наименование поля	Описание применения
Дата заявки	Дата зарегистрированной заявки на тур
Дата окончания	Дата окончания действия заявки
Фамилия менеджера	Фамилия менеджера турагентства
Служебное сообщение	Описание

Таблица 1.4 - Структура данных «Категория тура»

Наименование поля	Описание применения
ID_категории	Номер категории
Наименование	Категория турпредложения

Таблица 1.5 - Структура данных «Тур»

Наименование поля	Описание применения
ID_тура	Номер тура
Начало	Начало тура
Окончание	Окончание тура
Проживание	Проживание по туру
Питание	Тип писания по туру
ID_категории	Номер категории
ID_направления	Номер направления
ID_туропер	Номер туроператора
ID_типа	Номер типа тура

Таблица 1.6 - Структура данных «Тип тура»

Наименование поля	Описание применения
ID_типа	Номер типа
Название	Название типа тура

Таблица 1.7 - Структура данных «Услуги по туру»

Наименование поля	Описание применения
ID_услуги по туру	Номер услуги по конкретному туру
ID_услуги	Номер услуги
ID_тура	Номер тура

Таблица 1.8 - Структура данных «Услуга»

Наименование поля	Описание применения
ID_услуги	Номер услуги
Название услуги	Название услуги
ID_вида	Номер вида

Таблица 1.9 - Структура данных «Туроператор»

Наименование поля	Описание применения
ID_туропер	Номер туроператора
Наименование	Наименование оператора турагентства
Адрес	Адрес нахождения туроператора
Номер телефона	Номер телефона
ID_ договора	Номер договора
Дата заключения	Дата договора с туроператором

Таблица 1.10 - Структура данных «Журнал контрактов»

Наименование поля	Описание применения
Номер контракта	Номер контракта
ID_тура	Номер тура
ID_ клиента	Номер клиента
Дата начала договора	Дата начала договора
Дата окончания договора	Дата окончания договора
Закрыто	Статус договора
Служебное сообщение	Описание

Таблица 1.11 - Структура данных «Специальное предложение»

Наименование поля	Описание применения
ID_ СПО	Номер спецпредложения
Специальное предложение	Название СПО
Описание	Подробное описание СПО

Таблица 1.12 - Структура данных «Платеж»

Наименование поля	Описание применения
ID_платежа	Номер платежа за тур
ID_записи	Номер договора
Общая сумма за тур	Общая сумма за тур
Дата платежа	Дата платежа
ID_типа_оплаты	Номер типа оплаты
ID_ назначения оплаты	Идентификатор назначения оплаты

Таблица 1.13 - Структура данных «Тип оплаты»

Наименование поля	Описание применения
ID_типа_оплаты	Номер вида оплаты
Название	Название вида оплаты

Таблица 1.14 - Структура данных «Назначение оплаты»

Наименование поля	Описание применения
ID_ назначения	Идентификатор назначения оплаты
Наименования назначения оплаты	Наименование назначения оплаты

Таблица 1.15 - Структура данных «Направление тура»

Наименование поля	Описание применения
Код направления тура	Идентификатор направления

Наименование поля	Описание применения
Наименование направления тура	Название направления туристической путевки

Таблица 1.16 - Структура данных «Спецредложение_по_туру»

Наименование поля	Описание применения
ID_спецпредложения_ по туру	Номер СПО
ID_ Спецредложения	Номер спецпредложения
ID_тура	Номер тура

1.8.2 Разработка структуры связей

Для обеспечения целостности базы данных между таблицами устанавливаются связи, реализуемые в виде ссылочных полей. Различают идентифицирующую и неидентифицирующую связи. Если внешний ключ передаётся в дочернюю таблицу и, в результате создаётся зависимая таблица, то такая связь называется идентифицирующей связью между родительской и дочерней таблицами. При установлении неидентифицирующей связи дочерняя таблица остаётся независимой. Описание связей между таблицами приведено в таблице 1.17.

Таблица 1.17 – Структура связей между таблицами

Родительская таблица	Дочерняя таблица	Тип связи	Мощность связи	Связующий атрибут
Категория тура	Тур	неидентифицирующая	*1:N*	Идентификатор категории тура
Направление тура	Тур	неидентифицирующая	*1:N*	Идентификатор направления
Туроператор	Тур	неидентифицирующая	*1:N*	Идентификатор туроператора
Тип тура	Тур	неидентифицирующая	*1:N*	Идентификатор вида тура
Тур	Услуги по туру	идентифицирующая	*1:N*	Идентификатор тура, соответствующей туру

Родительская таблица	Дочерняя таблица	Тип связи	Мощность связи	Связующий атрибут
Тур	СПО по туру	идентифицирующая	*1:N*	Идентификатор Номер тура
Услуга	Услуги по туру	идентифицирующая	*1:N*	Идентификатор услуги
Вид услуги	Услуга	неидентифицирующая	*1:N*	Идентификатор вида услуги туристического пакета
Тур	Журнал договоров	неидентифицирующая	*1:N*	Идентификатор тура
Клиент	Журнал договоров	идентифицирующая	*1:N*	Идентификатор клиента
Договор	Оплата	неидентифицирующая	*1:N*	Идентификатор договора
Тип оплаты	Оплата	неидентифицирующая	*1:N*	Идентификатор вида оплаты
Назначение оплаты	Оплата	неидентифицирующая	*1:N*	Идентификатор назначения оплаты
СПО	СПО по туру	идентифицирующая	*1:N*	Идентификатор СПО

1.8.3 Нормализация отношений базы данных

При проектировании БД упор в первую очередь делается на достоверность и непротиворечивость хранимых данных, причем эти свойства не должны утрачиваться в процессе работы с данными, т.е. после многочисленных изменений, удалений и дополнений данных по отношению к первоначальному состоянию БД. Под нормализацией отношения подразумевается процесс приведения отношения к одной из нормальных форм (НФ) [8].

БД центра туристского сервиса находится в 1 НФ, так как каждое значение каждого из атрибутов всех отношений является атомарным, то есть хранит одно единственное значение и не является ни списком, ни множеством значений. БД находится во 2 НФ, так как она находится в 1 НФ, и при этом все не ключевые атрибуты зависят только от ключа целиком, а не от какой-то его части. БД находится в 3 НФ, так как она находится во 2 НФ и все не ключевые

атрибуты взаимно независимы, то есть факты, хранимые в таблицы, зависят только от ключа.

1.8.4 Физическое проектирование БД

На этапе логического проектирования на основе семантического анализа предметной области осуществляется разбиение БД на таблицы для СУБД, а также организация индексов и все связанные с этим ограничения целостности и средства защиты, выбирается целевая СУБД, определяется физическая организация данных. Соответствие таблиц сущностям логической модели системы приведено в таблице 1.18.

Таблица 1. 18 - Соответствие таблиц логической и физической моделей БД

Логическая модель	Имя таблицы
Клиент	*client*
Категория тура	*category _tour*
Тур	*tour*
Тип тура	*type_of_tour*
Услуги по туру	*services_tour*
Услуга	*service*
Вид услуги	*type_of_service*
Туроператор	*touroperator*
Журнал договоров	*journal_of_contracts*
Платеж	*payment*
Тип оплаты оплаты	*type_of_payment*
Назначение оплаты	*purpose_of_payment*
Направление тура	*way*
Журнал заявок	*journal_of_applications*
Спецпредложения (СПО)	*spo*
СПО по туру	*spo_on_the_tour*

Физическая модель базы данных была разработана в среде *MySQL Workbench 8.0.33* (рисунок 1.13).

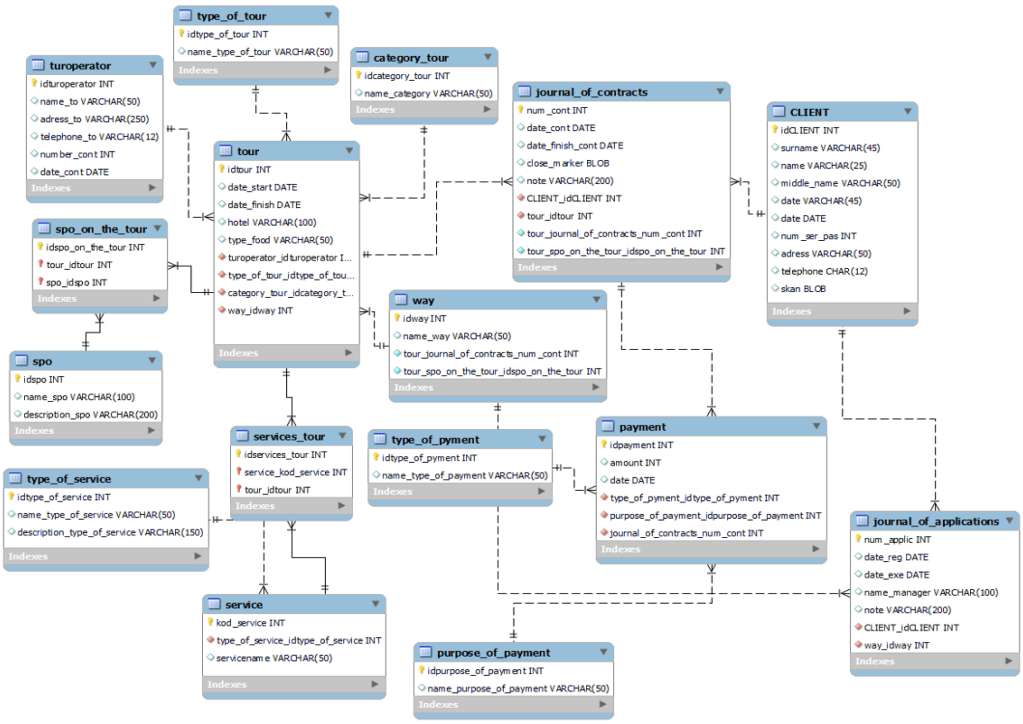


Рисунок 1.13 – Физическая модель базы данных

1.9 Разработка программного обеспечения

1.9.1 Общие сведения

Программная реализация выполнена на языке программирования *PHP 7.4* (фреймворк *Yii2*), стандартизированного языка гипертекстовой разметки документов *HTML, Java Script, CSS*. Система разработана для использования под управлением ОС *Windows* 7 и выше.

1.9.2 Функциональное назначение

Разрабатываемая информационная система центра туристического агентства предназначена для автоматизации работ, связанных с учетом клиентов; учетом сведений о заказах, а именно добавление заказа на основе выбора тура клиентом, где учитывается маршрут, дата тура, стоимость, туроператор. Система предоставляет возможность ввода, редактирования, а также просмотра о текущих и предстоящих событиях; формировать печатные формы документов, включающий договор реализации туристского продукта, сведения о туроператорах, письменное согласие туриста, заявку на бронирование услуг; просмотр отчета о деятельности туристической фирмы с

интервалом времени, включающий сведения о заказах ио оплатах, полученных фирмой, а также просмотр сведений о туроператоре. Таким образом, система должна существенным образом сократить затраты рабочего времени и повысить эффективность работы за счет того, что система легка в обращении, позволяет хранить большое количество сведений в одной базе данных, автоматизирует создание различных отчетностей и др.

Для локальной разработки проекта использовался *Open Server Panel*. *Open Server Panel* — это локальный веб-сервер для *Windows* и программная среда, созданная специально для веб-разработчиков, поэтому скачиваем и устанавливаем.

После успешной установки необходимо запустить консоль *OSPanel* перейти в директорию domains (команда *"cd domains"*) и выполнить команду *«composer create-project --prefer-dist yiisoft/yii2-app-basic travel-agency»* для предварительной установки проекта на фреймворке *Yii2*. После успешной установки проекта переходим в папку проекта (команда «*cd travel-agency*») устанавливаем необходимые библиотеки запустив команду *«composer install»* и запускаем инициализацию проекта выполнив команду *«php yii init»*.

Открываем проект в *IDE PhpStorm*. Далее согласно разработанной схеме БД формируем миграции в папке migrations. Миграцию можно создать вручную, либо через консоль выполнив команду *«php yii migrate/create migration_name»*, где *«migration_name»* – это название миграции, обычно описывающее предназначение этой миграции.

После того как все миграции созданы, необходимо создать БД и прописать данные доступы к ней, для этого откроем *PhpMyAdmin* и выполним необходимые действия. После этого можно выполнить миграции к созданной БД запустив в консоле команду *«php yii migrate»*. В результате успешного выполнения миграций в базе данных будут созданы все необходимые таблицы,

а также таблица *migration* в которой будут храниться название и время всех выполненных миграций.

Следующим шагом будет создание моделей, согласно тем таблицам, что были созданы в результате выполнения миграций. Модели будут отвечать за манипулирование данными в БД, поэтому они будут наследованы от класса *ActiveRecord*. Создавать модели можно вручную, либо использовать модуль *Gii* для автоматической генерации кода. После того как все модели созданы, создадим контроллеры и представления. Они также могут быть созданы вручную или при помощи модуля *Gii*.

Для демонстрации готовой программы был выбран бесплатный хостинг beget.com, который предоставляет:

– размещение одного сайта;

– память - 1Гб;

– размещение одной базы данных;

– *FTP* доступ к файловой системе сайта.

Используя данные *FTP* загружаем все файлы проекта на выделенный бесплатный сервер от beget.com и прописываем доступы от БД.

Делаем экспорт структуры БД с локального компьютера используя *PhpMyAdmin*, в результате чего получаем файл с расширением «*.sql*». Импортируем файл в базу данных на *beget.ru*.

Для проектирования *Web*-приложения существует множество различных архитектурных паттернов [9]. Перечислим несколько основных:

– модель-представление-контроллер (*Model View Controller*);

– контроллер страниц (*Page Controller*);

– преобразователь (*Transform View*);

– представление по шаблону (*Template View*);

– контроллер запросов (*Front Controller*) [9].

Изучив каждый паттерн проектирования, можно сделать сводную таблицу с кратким описанием (таблица 1.19).

Таблица 1.19 – *Web*-паттерны представления данных

Название	Описание
Модель-представление-контроллер (*Model View Controller*)	Представление отображает содержимое модели средствами графического интерфейса. Контроллер получает входные данные от пользователя, выполняет операции над моделью и указывает представлению на необходимость соответствующего обновления.
Контроллер страниц *(Page Controller*)	В основе контроллера страниц лежит идея создания компонентов, которые будут выполнять роль контроллеров для каждой страницы Веб-сайта. На практике количество контроллеров не всегда в точности соответствует количеству страниц, поскольку иногда при щелчке на ссылке открываются страницы с разным динамическим содержимым.
Контроллер запросов (*Front Controller*)	Контроллер запросов обрабатывает все запросы, поступающие к Веб-сайту, состоит из двух частей: Веб-обработчика и иерархии команд. Веб-обработчик – это объект, выполняющий фактическое получение *POST*- или *GET*-запросов, поступивших на Веб-сервер. Веб-обработчик обычно реализуется в виде класса, поскольку он не генерирует никаких откликов. Команды также являются классами; более того, им не нужно знать о наличии Веб-окружения, несмотря на то, что им часто передается информация из *HTTP*-запросов.
Представление по шаблону (*Template View*)	Основная идея, лежащая в основе типового решения представление по шаблону, – вставка маркеров в текст готовой статической *HTML*-страницы. При вызове страницы для обслуживания запроса эти маркеры будут заменены результатами некоторых вычислений (например, результатами выполнения запросов к базе данных). Подобная схема

	позволяет создавать статическую часть страницы с помощью обычных средств, например, текстовых редакторов, работающих по принципу *WYSIWYG*, и не требует знания языков программирования. Для получения динамической информации маркеры обращаются к отдельным программам.
Преобразователь (*Transform View)*	Преобразует записи в *HTML* по одной. Когда выполняются запросы к БД, вы получаете данные, но этого недостаточно, чтобы отобразить нормальную *web*-страницу. Использование *Transform View* подразумевает преобразование, когда на входе есть модель, а на выходе *HTML*.

В рамках реализации проекта для выпускной квалификационной работы был выбран паттерн *MVC* (рисунок 1.14).

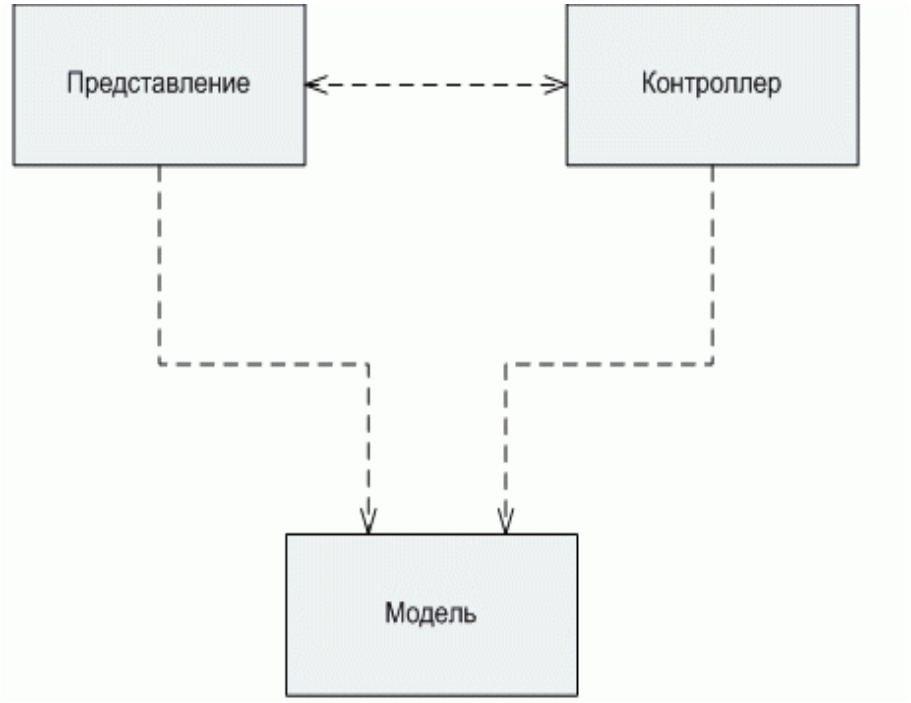


Рисунок 1.14 – *MVC*-паттерн

MVC(Model-View-Controller) – паттерн, разделяющий логику приложения на три части:

– *model* – модуль отвечающий за работу с базой данных, бизнес-логику и различные вычисления;

– *view* – модуль в виде набора пользовательских интерфейсов, отображающих полученные в результате вычислений или выполнения бизнес-логики данные от модели;

– *controller* – модуль обрабатывающий запросы пользователя, параметры запроса передает в модель для дальнейшей обработки и получения данных.

Использую данный шаблон можно писать независимые друг от друга блоки кода, то есть при изменении различных блоков не будет нарушена работа остальной части приложения.

При таком разделении компонентов приложения, соблюдается концепция разделения ответственности. При соблюдении такой концепции намного проще проводить работу над отдельными компонентами.

Выбор этого паттерна объясняется тем, что паттерн интуитивно понятен, прост в реализации, создан для создания гибких и масштабируемых приложений, которые можно будет в дальнейшем легко сопровождать и дополнять функционал приложения.

Для того чтобы реализовать проект в виде *web*-приложения придерживаться выбранного архитектурного решения, был выбран один из самых распространенных фреймворков *Yii*.

Yii – это компонентный и высокопроизводительный фреймворк на языке PHP, предназначенный для быстрой разработки современных *web*-приложений. *Yii* является *full-stack* фреймворком, включающий в себя проверенные и безопасные возможности, такие как *ActiveRecord* для реляционных и *NoSQL* баз данных, поддержку *REST API* и другие.

Согласно правилам фреймворка для реализации архитектуры *MVC* организация директорий проекта, следующая:

– в папке *models* хранятся все модели, представляющие собой классы, наследованные от базового класса *yii\base\Model*;

– в папке *controllers* хранятся контроллеры в виде классов, наследованных от базового контроллера *yii\base\Controller*, с определенным названием,

состоящим из *ID* контроллера и окончанием «*Controller*» (например, контроллер *SiteController* будет иметь *ID – site*). Каждый контроллер состоит из действий (*actions*) – это общедоступные функции с определенным названием, начинающимся с предопределенного слова «*action*» и *ID* действия (например, действие *actionIndex* имеет *ID – index*);

– в папке *views* хранятся представления в виде *PHP*-файлов в которых находится *html*-разметка для отображения страниц интерфейса конечному пользователю.

Название директорий в этой папке указывается согласно *ID* контроллера, которым используется.

Для доступа к действиям контроллеров, конечный пользователь использует маршруты. Маршруты – это строки, составленные по определенному правилу «*ID* контроллера/*ID* действия». Например, маршрут «*auth/login*» будет обращаться к контроллеру *AuthController* и действию в нем *actionLogin*. При необходимости, правила маршрутов можно изменить в конфигурационном файле приложения.

1.9.3 Регистрация на хостинге

Перед тем, как начать разрабатывать автоматизированную информационную систему центра туристического агентства, необходимо зарегистрироваться на хостинге. В качестве хостигга для разработке был выбран *beget.ru.* Выбор был определен возможностями, которые предоставляет данный хостинг:

– функциональный виртуальный хостинг;

– быстрые *SSD*-диски на всех тарифах;

– доступна установка *CMS*, форумов и блогов, а также других приложение в один клик;

– качественная работа службы поддержки – связь по телефону, через электронную почту, систему тикетов, *Skype, Telegram, Facebook*;

– домены в подарок на отдельных тарифных планах;

– отменные показатели производительности;

– доступные расценки на регистрацию и продление доменов;

– тестовый период – 30 дней;

– удобная панель управления собственной сборки;

– бесплатные *SSL*-сертификаты - можно выбрать подходящий тип;

– доступность тарифов для высоконагруженных сайтов с большой посещаемостью;

– хорошая партнерская программа.

Стартовая страница сайта *beget.ru* представлена на рисунке 1.15.

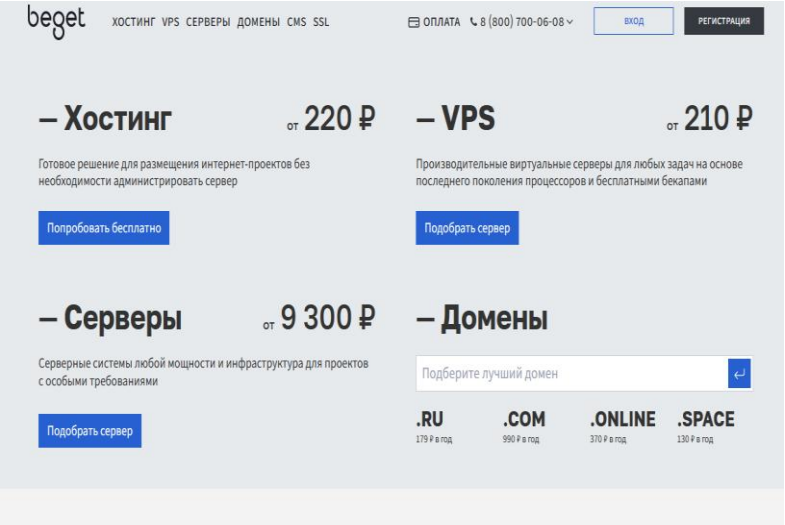


Рисунок 1.15 – Стартовая страница *Beget*

После успешной регистрации хостинг предоставляет инструменты для разработки сайта турагентства.

Страница с инструментами для разработки сайта *beget.ru* представлена на рисунке 1.16.

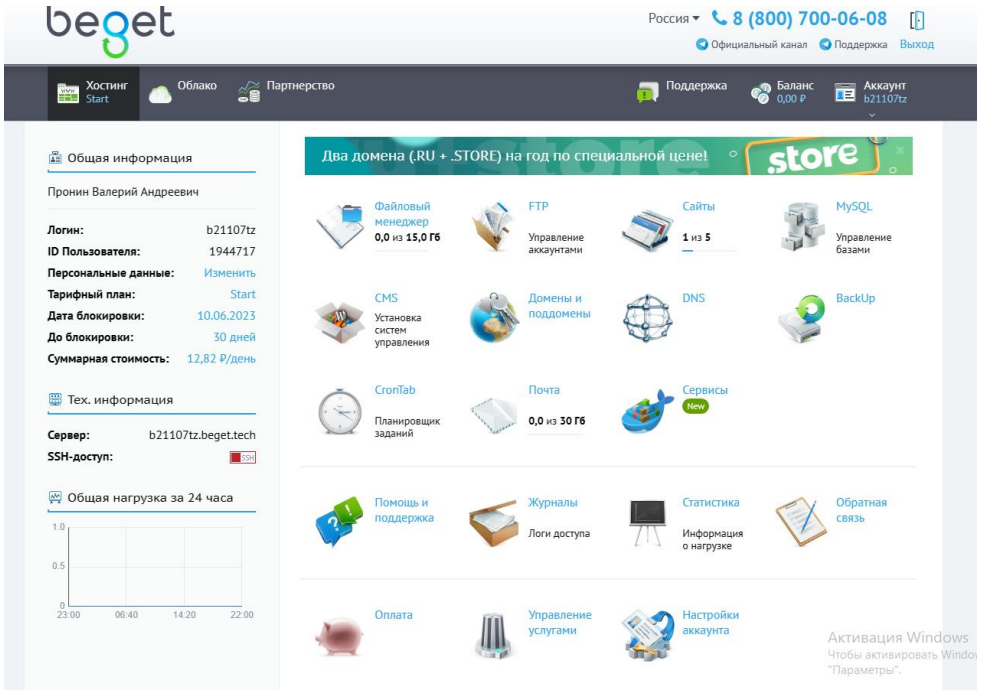


Рисунок 1.16 – Страница с инструментами для разработки

Была создана база данных с использованием СУБД – *MySQL*.

Страница с инструментами для создания новой базы данных на *beget.ru* представлена на рисунке 1.17.

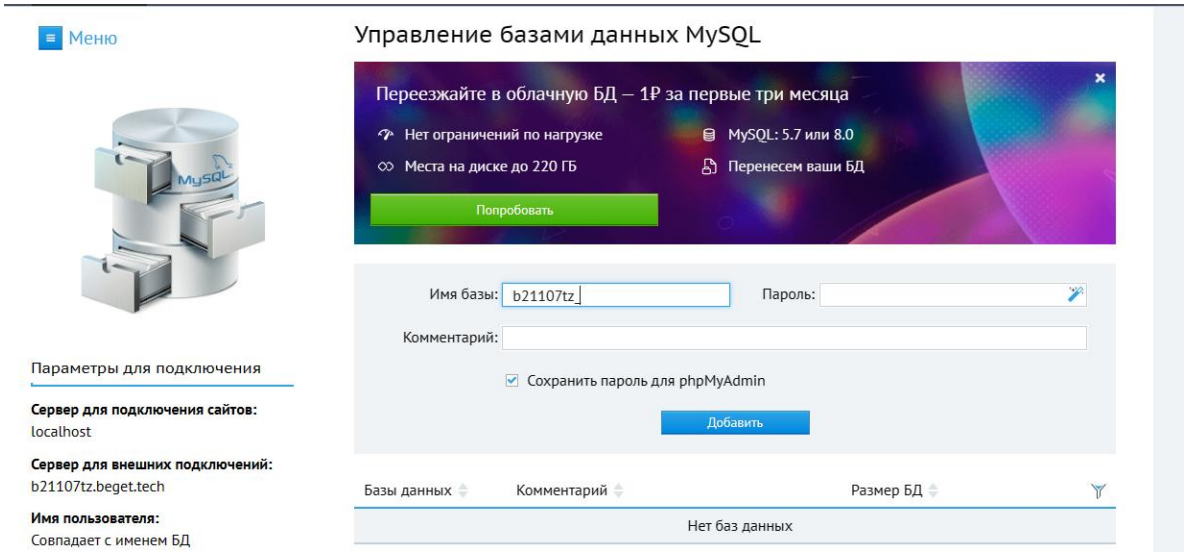


Рисунок 1.17 – Страница для создания новой базы данных

После создания базы данных с помощью утилиты *phpMyAdmin* можно перейти в окно для создания структуры базы данных туристического агентства.

Страница утилиты *phpMyAdmin* представлена на рисунке 1.18.

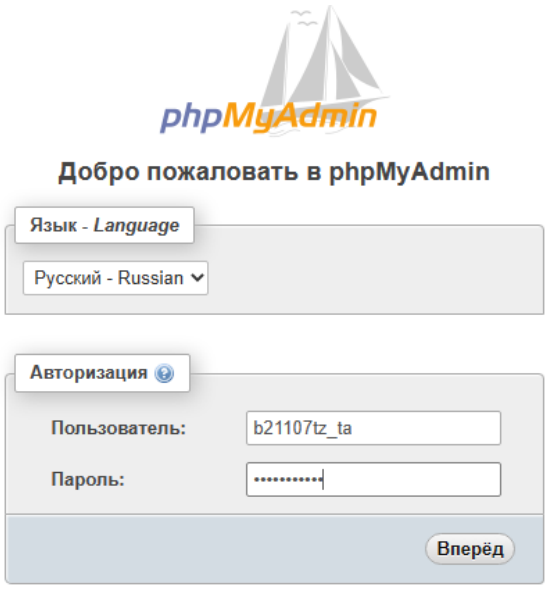


Рисунок 1.18 – Страница утилиты *phpMyAdmin*

Кроме того, необходимо зарегистрировать доменное имя на хостинге. Для этого, на страницы с инструментами *beget.ru* был выбран раздел «Сайты». Сайт успешно был создан и зарегистрирован. Страница регистрации нового сайта представлена на рисунке 1.19.

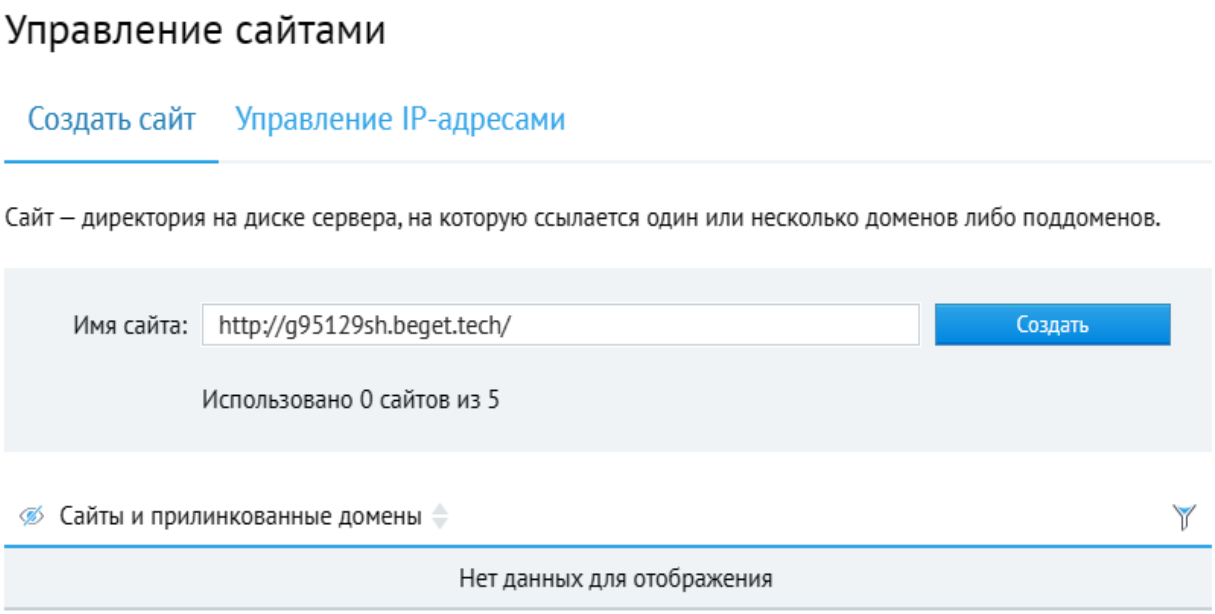


Рисунок 1.19 – Страница регистрации нового сайта

1.9.4 Описание программы

Файл *assets/AppAsset.php* – файл, в котором описаны подключаемые ресурсы (*JS* и *CSS* файлы), используемые в шаблоне *web*-приложения. Описанные в этом файлы ресурсы будут автоматически включены в *html*-разметку *web*-приложения (рисунок 1.20).

```php
class AppAsset extends AssetBundle
{
    public $basePath = '@webroot';
    public $baseUrl = '@web';
    public $css = [
        'css/site.css',
    ];
    public $js = [
    ];
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap4\BootstrapAsset'
    ];
}
```

Рисунок 1.20 - Класс *AppAsset* с определяющий подключаемые ресурсы

Файлы *config/db.php* – конфигурационный файл, определяющий параметры, необходимые для создания и инициализации экземпляра *yii\db\Connection*, через который будут осуществляться *SQL* запросы к описанной базе данных. Также может хранить конфигурацию кеширования запросов к БД (рисунок 1.21).

```php
<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=g95129sh_travel',
    'username' => 'g95129sh_travel',
    'password' => 'bQ4e%5ne',
    'charset' => 'utf8',

    // Schema cache options (for production environment)
    //'enableSchemaCache' => true,
    //'schemaCacheDuration' => 60,
    //'schemaCache' => 'cache',
];
```

Рисунок 1.21 - Конфигурация подключения к базе данных

Файл *config/params.php* – файл, хранящий глобальные параметры, которые могут использоваться в любом месте *web*-приложения (рисунок 1.22).

```php
<?php

return [
    'adminEmail' => 'admin@example.com',
    'senderEmail' => 'noreply@example.com',
    'senderName' => 'Турагенство',
];
```

Рисунок 1.22 - Определение глобальных параметров приложения

Файл *config/web.php* – конфигурационный файл *web*-приложения, в котором определяются название приложения, язык, подключаемые компоненты и глобальные параметры (рисунок 1.23).

```php
$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';

$config = [
    'id' => 'basic',
    'name' => 'Турагенство',
    'basePath' => dirname( path: __DIR__),
    'bootstrap' => ['log'],
    'language' => 'ru-RU',
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'   => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [...],
        'cache' => [...],
        'user' => [...],
        'errorHandler' => [...],
        'mailer' => [...],
        'log' => [...],
        'db' => $db,
        'urlManager' => [...],
    ],
    'params' => $params,
];
```

Рисунок 1.23 - Конфигурация *web*-приложения

Файл *controllers/ClientController.php* – файл-контроллер, хранящий действия для работы с моделью *Client.php*; доступные действия: *index* – отвечает за просмотр всех клиентов, сохраненных в БД, *create* – добавление нового клиента, *update* – редактирование данных существующего клиента, *view* – просмотр всех данных о клиенте, *delete* – удаление записи из БД о клиенте (рисунок 1.24).

```php
/**
 * Lists all Client models.
 *
 * @return string
 */
public function actionIndex()
{
    $searchModel = new ClientSearch();
    $dataProvider = $searchModel->search($this->request->queryParams);

    return $this->render( view: 'index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}
```

Рисунок 1.24 - Функция *actionIndex ()* отвечающяя за вывод, сортировку и поиск клиентов

Файл *controllers/JornalOfApplicationsController.php* – файл-контроллер, хранящий действия для работы с моделью *JornalOfApplications.php;* доступные действия: *index* – отвечает за просмотр всех заявок клиентов, *create* – добавление новой заявки, *update* – редактирование данных ранее созданной заявки, *view* – просмотр всех данных в заявке, *delete* – удаление заявки из БД (рисунок 1.25).

```php
/**
 * Displays a single JournalOfApplications model.
 * @param int $id ID
 * @return string
 * @throws NotFoundHttpException if the model cannot be found
 */
public function actionView($id)
{
    return $this->render( view: 'view', [
        'model' => $this->findModel($id),
    ]);
}
```

Рисунок 1.25 - Функция *actionView()* отвечающая за просмотр данных одной модели

Файл *controllers/JornalOfContracts.php* – файл-контроллер, хранящий действия для работы с моделью *JornalOfContracts.php*; доступные действия: *index* – отвечает за просмотр всех договоров, *create* – добавление нового договора, *update* – редактирование данных договора, *view* – просмотр всех данных о договоре, *delete* – удаление договора (рисунок 1.26).

```php
/**
 * Creates a new JournalOfContracts model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return string|\yii\web\Response
 */
public function actionCreate()
{
    $model = new JournalOfContracts();

    if ($this->request->isPost) {
        if ($model->load($this->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }
    } else {
        $model->loadDefaultValues();
    }

    return $this->render( view: 'create', [
        'model' => $model,
    ]);
}
```

Рисунок 1.26 - Функция *actionCreate*() отвечающая за показ формы при *GET*-запросе и сохранение данных в БД при *POST* запросе

Файл *controllers/PaymentController.php* – файл-контроллер, хранящий действия для работы с моделью *Payment.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск оплат, *create* – добавление новой оплаты, *update* – редактирование данных по оплате, *view* – просмотр всех данных об оплете, *delete* – удаление оплаты из БД. Функция *findModel()* принимает на вход ключ *ID* модели *Payment* и пытается найти запись в БД. При отсутствии данных будет выброшено исключение с соответствующим сообщением (рисунок 1.27).

```php
/**
 * Finds the Payment model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param int $id ID
 * @return Payment the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    if (($model = Payment::findOne(['id' => $id])) !== null) {
        return $model;
    }

    throw new NotFoundHttpException( message: 'The requested page does not exist.');
}
```

Рисунок 1.27 - Функция *findModel()* принимает на вход ключ *ID* модели *Payment*

Файл *controllers/PurposeOfPaymentController.php* – файл-контроллер, хранящий действия для работы с моделью *PurposeOfPayment.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск назначений оплаты, *create* – добавление нового назначения оплаты, *update* – редактирование данных назначения, *delete* – удаление записи из БД. Функция *actionUpdate()* при *GET*-запросе отображает форму с предзаполненными полями найденной модели, при *POST*-запросе данные в БД будут обновлены (рисунок 1.28).

```php
/**
 * Updates an existing PurposeOfPayment model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param int $id ID
 * @return string|\yii\web\Response
 * @throws NotFoundHttpException if the model cannot be found
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($this->request->isPost && $model->load($this->request->post()) && $model->save()) {
        \Yii::$app->session->setFlash( key: 'success', value: 'Данные успешно сохранены');
        return $this->redirect(['index']);
    }

    return $this->render( view: 'update', [
        'model' => $model,
    ]);
}
```

Рисунок 1.28 - Функция *actionUpdate()* при *GET*-запросе

Файл *controllers/ServiceController.php* – файл-контроллер, хранящий действия для работы с моделью *Service.php*; доступные действия: *index* –

отвечает за просмотр, сортировку и поиск услуг, *create* – добавление новой услуги, *update* – редактирование данных об услуги, *delete* – удаление услуги из БД (рисунок 1.29).

```php
/**
 * Deletes an existing Service model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param int $id ID
 * @return \yii\web\Response
 * @throws NotFoundHttpException if the model cannot be found
 */
public function actionDelete($id)
{
    $model = $this->findModel($id);
    if ($model->delete())
        \Yii::$app->session->setFlash( key: 'success',  value: 'Данные успешно удалены');

    return $this->redirect(['index']);
}
```

Рисунок 1.29 – Функция *actionDelete()* находит и удаляет модель с ключом *ID*, при успешном удалении выдает соответствующее сообщение

Таке используются следуюшие виды контроллеров:

– файл *controllers/SiteController.php* – файл-контроллер, хранящий действия инициализации приложения доступные действия: *index* – отвечает за показ стартового окна;

– файл *controllers/SpoController.php* – файл-контроллер, хранящий действия для работы с моделью *Spo.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск СПО; *create* – добавление нового СПО; *update* – редактирование данных СПО, *delete* – удаление СПО из БД.

– файл *controllers/TourCategoryController.php* – файл-контроллер, хранящий действия для работы с моделью TourCategory.php; доступные действия: *index* – отвечает за просмотр, сортировку и поиск категорий тура, *create* – добавление новой категории тура, *update* – редактирование данных категории, *delete* – удаление категории из БД.

– файл *controllers/TourController.php* – файл-контроллер, хранящий действия для работы с моделью *Tour.php*; доступные действия: *index* – отвечает

за просмотр, сортировку и поиск туров, *create* – добавление нового тура, *update* – редактирование данных тура, *view* – просмотр всех данных о туре, *delete* – удаление тура из БД.

– файл *controllers/TouroperatorController.php* – файл-контроллер, хранящий действия для работы с моделью *Touroperator.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск туроператоров, *create* – добавление нового туроператора, *update* – редактирование данных туроператора, *view* – просмотр всех данных о туроператоре, *delete* – удаление туроператора из БД.

– файл *controllers/TypeOfServiceController.php* – файл-контроллер, хранящий действия для работы с моделью *TypeOfService.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск типов услуг, *create* – добавление нового типа услуг, *update* – редактирование данных типа услуг, *delete* – удаление типа услуг из БД.

– файл *controllers/TypeOfTourController.php* – файл-контроллер, хранящий действия для работы с моделью *TypeOfTour.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск типов тура, *create* – добавление нового типа тура, *update* – редактирование данных типа тура, *delete* – удаление типа тура из БД.

– файл *controllers/WayController.php* – файл-контроллер, хранящий действия для работы с моделью *Way.php*; доступные действия: *index* – отвечает за просмотр, сортировку и поиск направлений, *create* – добавление нового направления, *update* – редактирование данных направления, *delete* – удаление направления из БД.

Файл *migrations/m230501_193549_create_client_table.php* – файл-миграции, создающий таблицу *«client»* (рисунок 1.30).

```php
public function safeUp()
{
    $this->createTable( table: '{{%client}}', [
        'id' => $this->primaryKey()->comment( comment: 'Личный номер клиента'),
        'surname' => $this->string()->notNull()->comment( comment: 'Фамилия'),
        'name' => $this->string()->notNull()->comment( comment: 'Имя'),
        'middle_name' => $this->string()->comment( comment: 'Отчество'),
        'date' => $this->date()->notNull()->comment( comment: 'Дата рождения'),
        'num_ser_pas' => $this->integer()->notNull()->comment( comment: 'Серия и номер паспорта'),
        'address' => $this->string()->notNull()->comment( comment: 'Адрес'),
        'telephone' => $this->string()->notNull()->comment( comment: 'Телефон'),
        'skan' => $this->binary( length: 16777215)->comment( comment: 'Скан паспорта')
    ]);
}
```

Рисунок 1.30 – Функция *safeUp()* выполняющая создание таблицы *client*

Файл *migrations/ m230501_193706_create_tour_category_table.php* – файл миграции, создающий таблицу *«category_tour»* (рисунок 1.31).

```php
public function safeUp()
{
    $this->createTable( table: '{{%tour_category}}', [
        'id' => $this->primaryKey()->comment( comment: 'Код категории тура'),
        'name_category' => $this->string()->notNull()->comment( comment: 'Наименование категории тура')
    ]);
}
```

Рисунок 1.31 – Функция *safeUp()* выполняющая создание таблицы *category_tour*

Файл *migrations/ m230501_193717_create_tour_table.php* – файл миграции, создающий таблицу *«tour»* (рисунок 1.32).

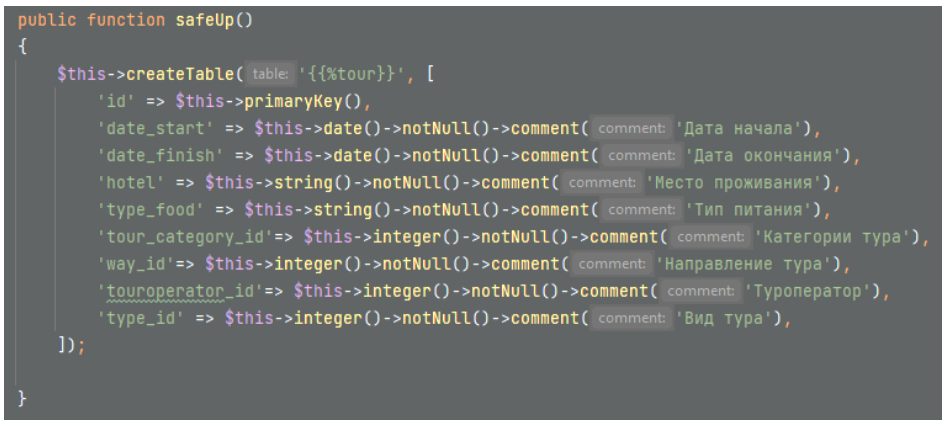


Рисунок 1.32 – Функция *safeUp()* выполняющая создание таблицы tour

Файл *migrations/ m230501_194428_create_type_of_tour_table.php* – файл миграции, создающий таблицу *«tour»* (рисунок 1.33).

Рисунок 1.33 – Функция *safeUp()* выполняющая создание таблицы *type_of_tour*

Файл *migrations/m230501_194618_create_service_table.php* – файл миграции, создающий таблицу *«service»* (рисунок 1.34).

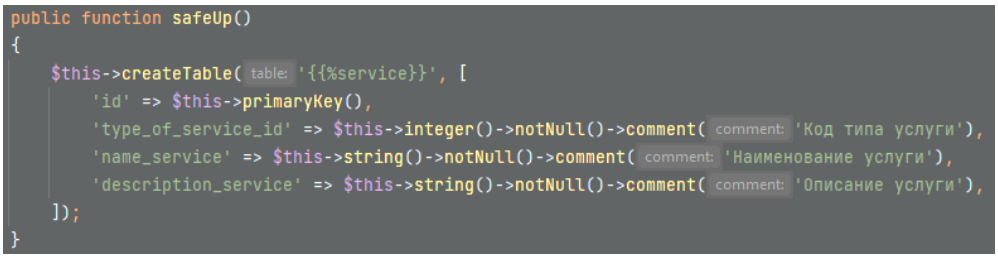


Рисунок 1.34 – Функция *safeUp()* выполняющая создание таблицы *service*

Файл *migrations/ m230501_194656_create_type_of_service_table.php* – файл миграции, создающий таблицу *«type_of_service»* (рисунок 1.35).

```php
public function safeUp()
{
    $this->createTable( table: '{{%type_of_service}}', [
        'id' => $this->primaryKey(),
        'name_type_of_service' => $this->string()->notNull()->comment( comment: 'Наименование вида услуги')
    ]);
}
```

Рисунок 1.35 – функция *safeUp()* выполняющая создание таблицы *type_of_service*

Файл *migrations/ m230501_195809_create_services_tour_table.php* – файл миграции, создающий таблицу *«services_tour»* (рисунок 1.36).

```php
public function safeUp()
{
    $this->createTable( table: '{{%services_tour}}', [
        'id' => $this->primaryKey(),
        'tour_id' => $this->integer()->notNull()->comment( comment: 'Код тура'),
        'service_id' => $this->integer()->notNull()->comment( comment: 'Код услуги')
    ]);
}
```

Рисунок 1.36 – Функция *safeUp()* выполняющая создание таблицы *services_tour*

Файл *migrations/ m230501_195836_create_touroperator_table.php* – файл миграции, создающий таблицу *«touroperator»* (рисунок 1.37).

```php
public function safeUp()
{
    $this->createTable( table: '{{%touroperator}}', [
        'id' => $this->primaryKey(),
        'name' => $this->string()->notNull()->comment( comment: 'Наименование туроператора'),
        'address' => $this->string()->notNull()->comment( comment: 'Адрес'),
        'telephone' => $this->string()->notNull()->comment( comment: 'Телефон'),
        'num_contract' => $this->integer()->notNull()->comment( comment: 'Номер договора'),
        'date_contract' => $this->date()->notNull()->comment( comment: 'Дата догвора'),
    ]);
}
```

Рисунок 1.37 – Функция *safeUp()* выполняющая создание таблицы *touroperator*

Файл *migrations/ m230501_195905_create_journal_of_contracts _table.php* – файл миграции, создающий таблицу *«journal_of_contracts»* (рисунок 1.38).

```php
public function safeUp()
{
    $this->createTable( table: '{{%journal_of_contracts}}', [
        'id' => $this->primaryKey(),
        'tour_id' => $this->integer()->notNull()->comment( comment: 'Код тура'),
        'client_id' => $this->integer()->notNull()->comment( comment: 'Код клиента'),
        'date_contract' => $this->date()->notNull()->comment( comment: 'Дата заключения договора'),
        'date_finish_contract' => $this->date()->notNull()->comment( comment: 'Дата закрытия договора'),
        'close_marker' => $this->binary()->comment( comment: 'Отметка о закрытии'),
        'note' => $this->string()->comment( comment: 'Примечание')
    ]);
}
```

Рисунок 1.38 – Функция *safeUp()* выполняющая создание таблицы *journal_of_contracts*

Файл *migrations/ m230501_195931_create_payment_table.php* – файл миграции, создающий таблицу *«payment»* (рисунок 1.39).

```php
public function safeUp()
{
    $this->createTable( table: '{{%payment}}', [
        'id' => $this->primaryKey(),
        'contract_id' => $this->integer()->notNull()->comment( comment: 'Код договора'),
        'amount' => $this->integer()->notNull()->unsigned()->comment( comment: 'Сумма оплаты'),
        'date' => $this->date()->notNull()->comment( comment: 'Дата оплаты'),
        'type_of_payment_id' => $this->integer()->notNull()->comment( comment: 'Тип оплаты'),
        'purpose_of_payment_id' => $this->integer()->notNull()->comment( comment: 'Назначение оплаты'),
    ]);

}
```

Рисунок 1.39 – Функция *safeUp()* выполняющая создание таблицы payment

Файл *migrations/ m230501_195955_create_type_of_payment_table.php* – файл миграции, создающий таблицу *«type_of_payment»* (рисунок 1.40).

```php
public function safeUp()
{
    $this->createTable( table: '{{%type_of_payment}}', [
        'id' => $this->primaryKey(),
        'name_type_of_payment' => $this->string()->notNull()->comment( comment: 'Наименование типа оплаты')
    ]);
}
```

Рисунок 1.40 – Функция *safeUp()* выполняющая создание таблицы *type_of_payment*

Файл *migrations/ m230501_200023_create_purpose_of_ payment_table.php* – файл миграции, создающий таблицу *«purpose_of_payment»* (рисунок 1.41).

```php
public function safeUp()
{
    $this->createTable( table: '{{%purpose_of_payment}}', [
        'id' => $this->primaryKey(),
        'name_purpose_of_payment' => $this->string()->notNull()->comment( comment: 'Наименование назначения
    ]);
}
```

Рисунок 1.41 – Функция *safeUp()* выполняющая создание таблицы *purpose_of_payment*

Файл *migrations/m230501_200036_create_way_table.php* – файл миграции, создающий таблицу *«way»* (рисунок 1.42).

```php
public function safeUp()
{
    $this->createTable( table: '{{%way}}', [
        'id' => $this->primaryKey(),
        'name_way' => $this->string()->notNull()->comment( comment: 'Наименование направления'),
    ]);
}
```

Рисунок 1.42 – Функция *safeUp()* выполняющая создание таблицы *way*

Файл *migrations/ m230501_200149_create_journal_of_applications _table.php* – файл миграции, создающий таблицу *«journal_of_applications»* (рисунок 1.43).

```php
public function safeUp()
{
    $this->createTable( table: '{{%journal_of_applications}}', [
        'id' => $this->primaryKey(),
        'client_id' => $this->integer()->notNull()->comment( comment: 'Код клиента'),
        'way_id' => $this->integer()->notNull()->comment( comment: 'Код направления'),
        'date_reg' => $this->date()->notNull()->comment( comment: 'Дата регистрации заявки'),
        'date_exe' => $this->date()->notNull()->comment( comment: 'Дата выполнения заявки'),
        'name_manager' => $this->string()->comment( comment: 'Имя менеджера'),
        'note' => $this->string()->comment( comment: 'Примечание')
    ]);
}
```

Рисунок 1.43 – Функция *safeUp()* выполняющая создание таблицы journal_of_application

Файл *migrations/ m230501_200220_create_spo_table.php* – файл миграции, создающий таблицу *«spo»* (рисунок 1.44).

```php
public function safeUp()
{
    $this->createTable( table: '{{%spo}}', [
        'id' => $this->primaryKey(),
        'name_spo' => $this->string()->notNull()->comment( comment: 'Наименование СПО'),
        'description_spo' => $this->string()->notNull()->comment( comment: 'Описание СПО')
    ]);
}
```

Рисунок 1.44 – Функция *safeUp()* выполняющая создание таблицы *spo*

Файл *migrations/ m230501_200258_create_spo_on_the_tour_table.php* – файл миграции, создающий таблицу *«spo_on_the_tour»* (рисунок 1.45).

```php
public function safeUp()
{
    $this->createTable( table: '{{%spo_on_the_tour}}', [
        'id' => $this->primaryKey(),
        'spo_id' => $this->integer()->notNull()->comment( comment: 'Код СПО'),
        'tour_id' => $this->integer()->notNull()->comment( comment: 'Код тура')
    ]);

    $this->addForeignKey( name: '{{%fk-tour-tour_category_id}}', table: '{{%tour}}', columns: 'tour_category_id', refTable: '{{%tour_cat
    $this->addForeignKey( name: '{{%fk-tour-way_id}}', table: '{{%tour}}', columns: 'way_id', refTable: '{{%way}}', refColumns: 'id', dele
    $this->addForeignKey( name: '{{%fk-tour-touroperator_id}}', table: '{{%tour}}', columns: 'touroperator_id', refTable: '{{%touroperat
    $this->addForeignKey( name: '{{%fk-tour-type_id}}', table: '{{%tour}}', columns: 'type_id', refTable: '{{%type_of_tour}}', refColumns
    $this->addForeignKey( name: '{{%fk-service-type_of_service_id}}', table: '{{%service}}', columns: 'type_of_service_id', refTable: '{
    $this->addForeignKey( name: '{{%fk-services_tour-tour_id}}', table: '{{%services_tour}}', columns: 'tour_id', refTable: '{{%tour}}',
    $this->addForeignKey( name: '{{%fk-services_tour-service_id}}', table: '{{%services_tour}}', columns: 'service_id', refTable: '{{%se
    $this->addForeignKey( name: '{{%fk-journal_of_contracts-tour_id}}', table: '{{%journal_of_contracts}}', columns: 'tour_id', refTab
    $this->addForeignKey( name: '{{%fk-journal_of_contracts-client_id}}', table: '{{%journal_of_contracts}}', columns: 'client_id',
    $this->addForeignKey( name: '{{%fk-payment-contract_id}}', table: '{{%payment}}', columns: 'contract_id', refTable: '{{%journal_of_c
    $this->addForeignKey( name: '{{%fk-payment-type_of_payment_id}}', table: '{{%payment}}', columns: 'type_of_payment_id', refTable: '{
    $this->addForeignKey( name: '{{%fk-payment-purpose_of_payment_id}}', table: '{{%payment}}', columns: 'purpose_of_payment_id', ref
    $this->addForeignKey( name: '{{%fk-journal_of_applications-client_id}}', table: '{{%journal_of_applications}}', columns: 'client_
    $this->addForeignKey( name: '{{%fk-journal_of_applications-way_id}}', table: '{{%journal_of_applications}}', columns: 'way_id',
    $this->addForeignKey( name: '{{%fk-spo_on_the_tour-tour_id}}', table: '{{%spo_on_the_tour}}', columns: 'tour_id', refTable: '{{%tour
    $this->addForeignKey( name: '{{%fk-spo_on_the_tour-spo_id}}', table: '{{%spo_on_the_tour}}', columns: 'spo_id', refTable: '{{%spo}}'

}
```

Рисунок 1.45 – Функция *safeUp()* выполняющая создание таблицы *spo_on_the_tour*

Файл *models/Client.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«client»* (рисунок 1.46). Функция *rules()* возвращает массив правил валидации полей, для проверки значений перед сохранением в таблицу. Функция *tableName()* возвращает название таблицы БД, которую реализует данный класс.

Рисунок 1.46 – Класс *Client*

Файл *models/JournalOfApplications.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«journal_of_applications»* (рисунок 1.47). Функция *getWay()* определяет отношение между моделью *JournalOfApplications* и *Way*.

Рисунок 1.47 – Функция *getClient()* определяет отношение между моделью *JournalOfApplications* и *Client*

Файл *models/JournalOfContracts.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«journal_of_contracts»* (рисунок 1.48).

Рисунок 1.48 – Содержимое класса *JournalOfContracts*

Файл *models/Payment.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«payment»* (рисунок 1.49).

Рисунок 1.49 – Функция *getContract()* определяет отношение модели *Payment* и *JournalOfContract*

Функция *getContract()* определяет отношение модели *Payment* и *PurposeOfPayment*. Функция *getContract()* определяет отношение модели *Payment* и *TypeOfPayment*.

Для доступа и манипулирования данными, хранящимися в базе данных, были разработаны следующие объекты:

– файл *models/PurposeOfPayment.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«purpose_of_payment»*;

– файл *models/Service.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«service»*;

– файл *models/ServicesTour.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«service_tour»*;

– файл *models/ Spo.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«spo»*;

– файл *models/ SpoOnTheTour.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«spo_on_the_tour»*;

– файл *models/ Tour.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«tour»*;

– файл *models/ TourCategory.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«tour_category»*;

– файл *models/ Touroperator.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«touroperator»*;

– файл *models/ TypeOfPayment.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«type_of_payment»*;

– файл *models/ TypeOfService.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«type_of_service»*;

– файл *models/ TypeOfTour.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных *«type_of_tour»*;

– файл *models/ Way.php* – объектно-ориентированный класс для доступа и манипулирования данными, хранящимися в базе данных «way».

Файл *models/search/ ClientSearch.php* – класс наследованный от *Client.php* и используемый для поиска и сортировки данных при просмотре общего списка клиентов (рисунок 1.50).

```php
public function search($params)
{
    $query = Client::find();
    $dataProvider = new ActiveDataProvider([
        'query' => $query,
    ]);
    $this->load($params);
    if (!$this->validate()) {
        return $dataProvider;
    }
    // grid filtering conditions
    $query->andFilterWhere([
        'id' => $this->id,
        'date' => $this->date,
        'num_ser_pas' => $this->num_ser_pas,
    ]);
    $query->andFilterWhere(['like', 'surname', $this->surname])
        ->andFilterWhere(['like', 'name', $this->name])
        ->andFilterWhere(['like', 'middle_name', $this->middle_name])
        ->andFilterWhere(['like', 'address', $this->address])
        ->andFilterWhere(['like', 'telephone', $this->telephone])
        ->andFilterWhere(['like', 'skan', $this->skan]);

    return $dataProvider;
}
```

Рисунок 1.50 – Функция *search()* принимает *GET*-параметры и фильтрует данные согласно заданным параметрам

Для поиска и сортировки данных в базе данных были реализованы следующие объекты:

– файл *models/search/JournalOfApplicationsSearch.php* – класс наследованный от *JornalOfApplications.php* и используемый для поиска и сортировки данных при просмотре общего списка заявок;

– файл *models/search/JournalOf ContractsSearch.php* – класс наследованный от *JournalOfContracts.php* и используемый для поиска и сортировки данных при просмотре общего списка договоров;

– файл *models/search/TourSearch.php* – класс наследованный от *Tours.php* и используемый для поиска и сортировки данных при просмотре общего списка туров (рисунок 1.51);

```php
public function search($params)
{
    $query = Tour::find();
    $dataProvider = new ActiveDataProvider([
        'query' => $query,
    ]);

    $this->load($params);

    if (!$this->validate()) {
        return $dataProvider;
    }
    // grid filtering conditions
    $query->andFilterWhere([
        'date_start' => $this->date_start,
        'date_finish' => $this->date_finish,
        'tour_category_id' => $this->tour_category_id,
        'way_id' => $this->way_id,
        'touroperator_id' => $this->touroperator_id,
        'type_id' => $this->type_id,
    ]);
    $query->andFilterWhere(['like', 'hotel', $this->hotel])
        ->andFilterWhere(['like', 'type_food', $this->type_food]);
    return $dataProvider;
}
```

Рисунок 1.51 – Функция *search()* принимает *GET*-параметры и фильтрует данные согласно заданным параметрам

– файл *models/search/PaymentSearch.php* – класс наследованный от Payment.php и используемый для поиска и сортировки данных при просмотре общего списка оплат;

– файл *models/search/ServiceSearch.php* – класс наследованный от *Service.php* и используемый для поиска и сортировки данных при просмотре общего списка услуг;

– файл *models/search/TouroperatorSearch.php* – класс наследованный от *Touroperator.php* и используемый для поиска и сортировки данных при просмотре общего списка туроператоров.

Файлы в папке *views* разбиты по папкам с названиями, соответствующими моделям, отображение которых они реализуют для конечного пользователя web-приложения.

В папке встречаются следующие файлы: *index.php* – отвечает за отображение общего списка данных соответствующей модели. Соответствует действию *index* класса-контроллера, *create* – отвечает за отображение формы добавления записи соответствующей модели в БД, *update* – отвечает за отображение формы редактирования данных в БД соответствующей модели, *view* – отвечает за отображение формы просмотра данных соответствующей модели, *_form* – отображает форму добавления/редактирования с полями соответствующей модели, *_search* – отображает форму поиска с полями соответствующей модели (рисунок 1.52).

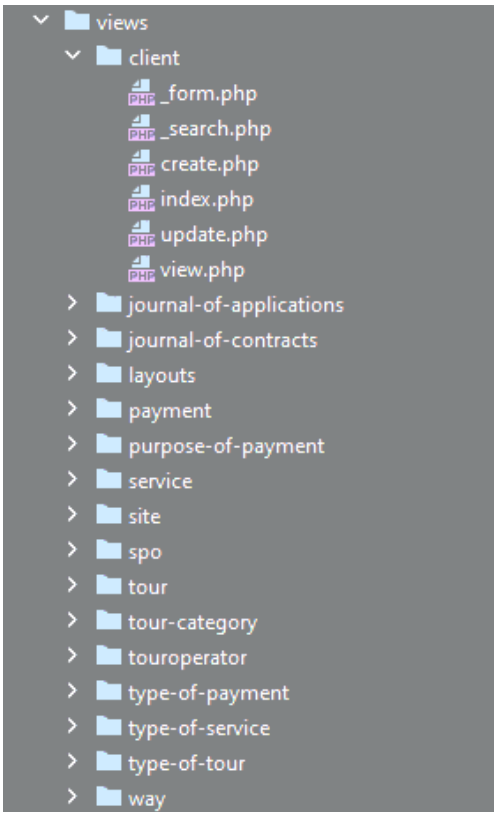


Рисунок 1.52 – Иерархия каталогов представлений с *html*-разметкой

Файл *web/index .php* – файл хранящий код входного скрипта *web*-приложения (рисунок 1.53).

```php
<?php

// comment out the following two lines when deployed to production
defined( constant_name: 'YII_DEBUG') or define('YII_DEBUG', true);
defined( constant_name: 'YII_ENV') or define('YII_ENV', 'dev');

require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/../vendor/yiisoft/yii2/Yii.php';

$config = require __DIR__ . '/../config/web.php';

(new yii\web\Application($config))->run();
```

Рисунок 1.53 – Инициализация приложения

1.9.5 Описание доступных функций

Для того, чтобы запустить программу необходимо прописать в адресной строке браузера следующий адрес: *http://g95129sh.beget.tech/web/*. Загрузится главная страница приложения, в левой стороне, которго находится полоса меню (рисунок 1.54).

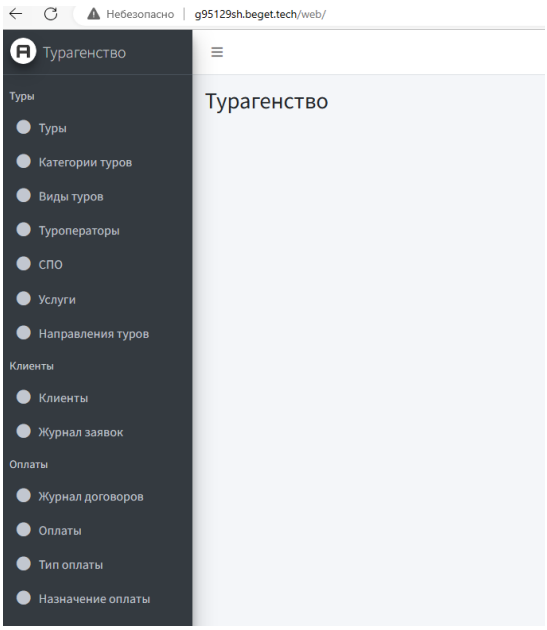


Рисунок 1.54 – Главная страница сайта турагентства

Web – приложение центра туристического агентства должна обеспечивать возможность выполнения следующих функций:

– администрирование системы;

– поддержка авторизации пользователей системы;

– учет сведений о клиентах;

– учет сведений о заказах;

– учет оплаты заказов;

– учет сведений о сотрудниках;

– формирование печатных форм документов;

– учет сведений о турах и категориях туров, направлениях туров;

– ведение истории заказов клиентов;

– учет сведений о спецпредложениях по турам;

– формирование отчетов по отчетным периодам деятельности фирмы.

При выборе пунктов меню осуществляется переход по *web* – страницам сайта турагентства. Для того, чтобы добавить информацию о туре необходимо выбрать пункт меню «Туры». После загрузки страницы будет представлена возможность турагенту просмотреть существующие туры или добавить новый тур (рисунок 1.55).

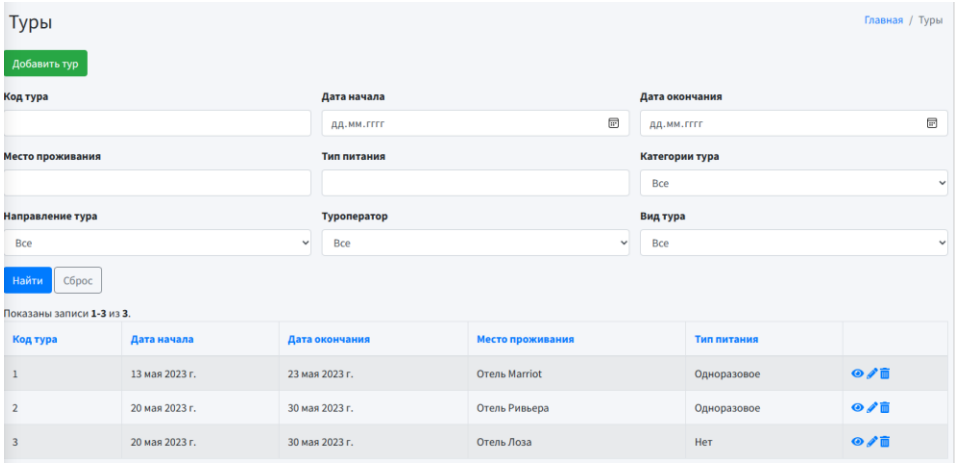


Рисунок 1.55 – Работа с разделами «Туры»

Для добавления информации о новом туре пользователь должен нажать на кнопку «Добавить тур». Откроется форма с пустыми полями. Для заполнения доступны следующие поля: дата начала тура; дата окончания тура; место проживания; тип питания; категория тура; направление тура; туроператор; вид тура; услуги тура; СПО по туру.

После заполнения всех полей необходимо нажать на кнопку «Сохранить» (рисунок 1.56).

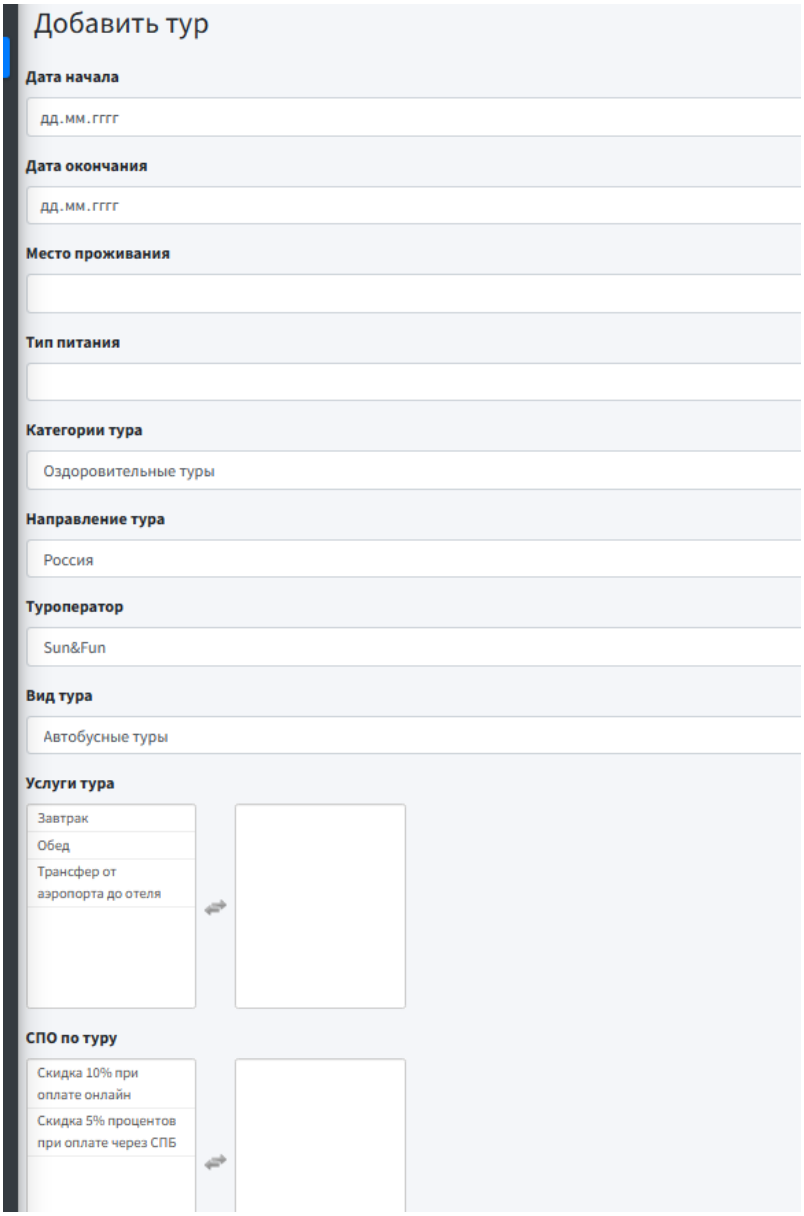


Рисунок 1.56 – Форма добавления тура

Для реализации туров покупателям, в программу необходимо добавить типы предоставляемых услуг, например: питание, трансфер и пр. (рисунок 1.57).

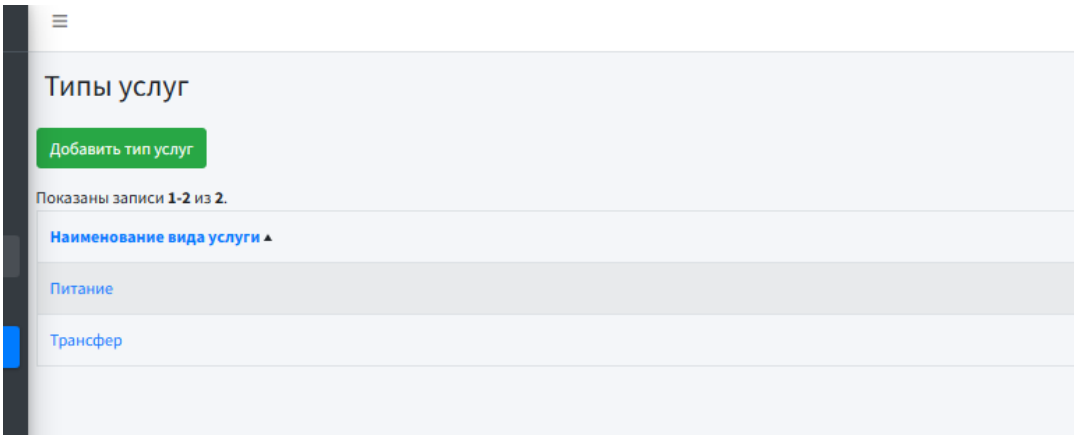


Рисунок 1.57 – Страница «Типы услуг»

Для заполнения категории туров необходимо перейти на страницу «Категории туров». Нажав на кнопку «Добавить категорию тура», администратор может добавить необходимую категорию, например: оздоровительные туры, познавательные туры, спортивные туры и др. (рисунок 1.58).

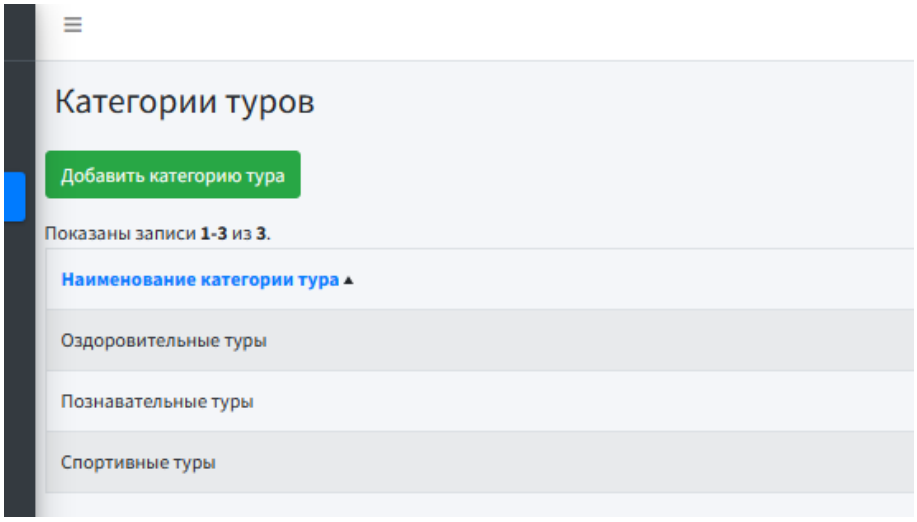


Рисунок 1.58 – Создание категории туров

Кроме того, чтобы вести полонённый бизнес, необходимо добавить в систему информацию о туроператорах (рисунок 1.59).

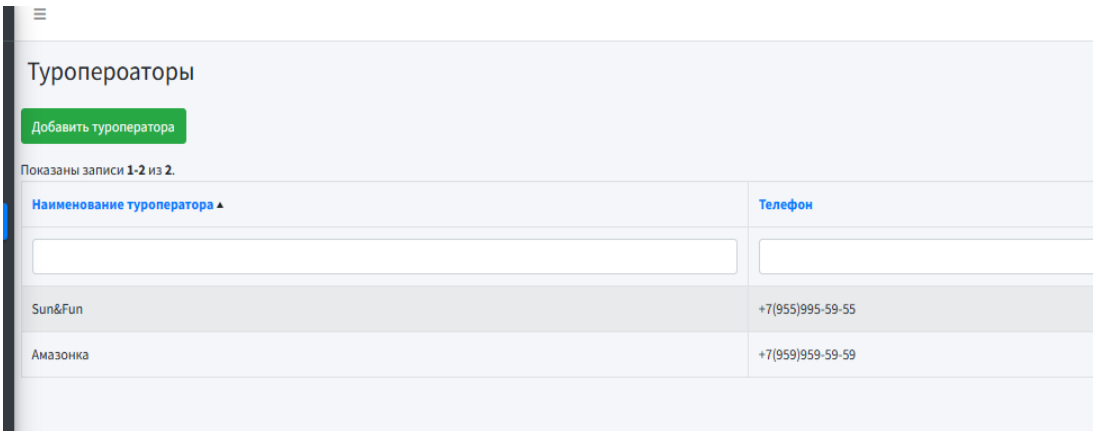


Рисунок 1.59 – Страница «Туроператоры»

Для этого выбирается пункт меню «Туроператоры» и по кнопке «Добавить туроператора» открывается форма для добавления информации о туроператоре (рисунок 1.60).

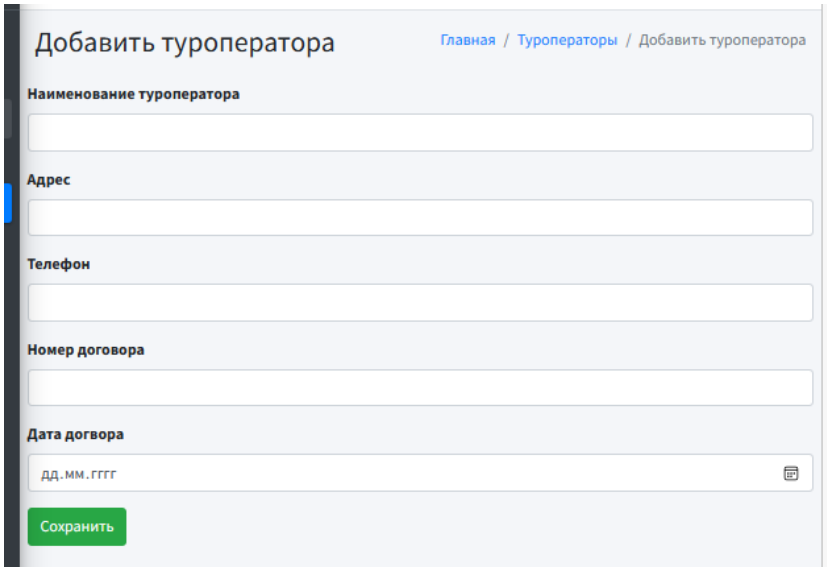


Рисунок 1.60 – Страница «Туроператоры»

В системе можно зарегистрировать список специальных предложений по туру, которые предоставляются клиенту. Для этого выбирается пункт меню СПО и поле нажатия на кнопку «Добавить СПО» можно занести информацию по спецпредложениям, например: информацию о различных скидках (рисунок 1.61).

Рисунок 1.61 – Страница «СПО»

Турагентства могут предоставлять клиентам различные типы услуг, например: трансфер, дополнительное питание и пр. Добавление списка услуг осуществляется через соответствующую форму после нажатия на кнопку «Типы услуг» (рисунок 1.62).

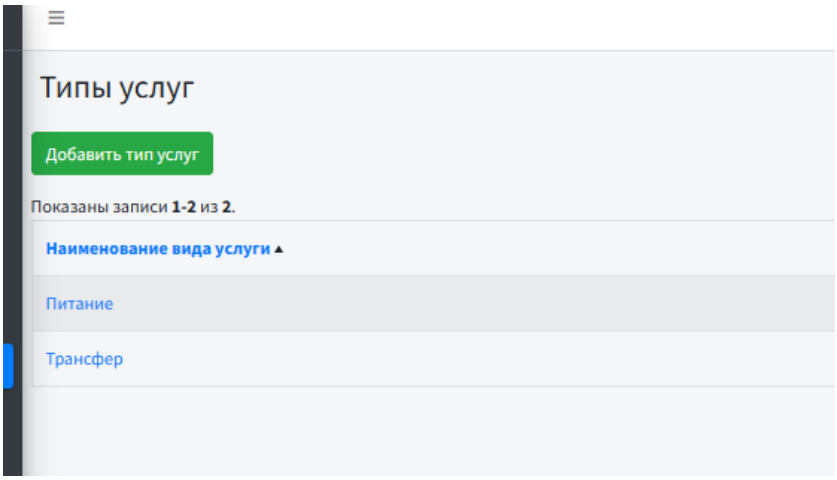


Рисунок 1.62 – Работа кнопки «Услуги»

В системе хранится информация о направлениях, для которых продаются туры. Чтобы зафиксировать направление, необходимо нажать на кнопку «Направления торов» (рисунок 1.63).

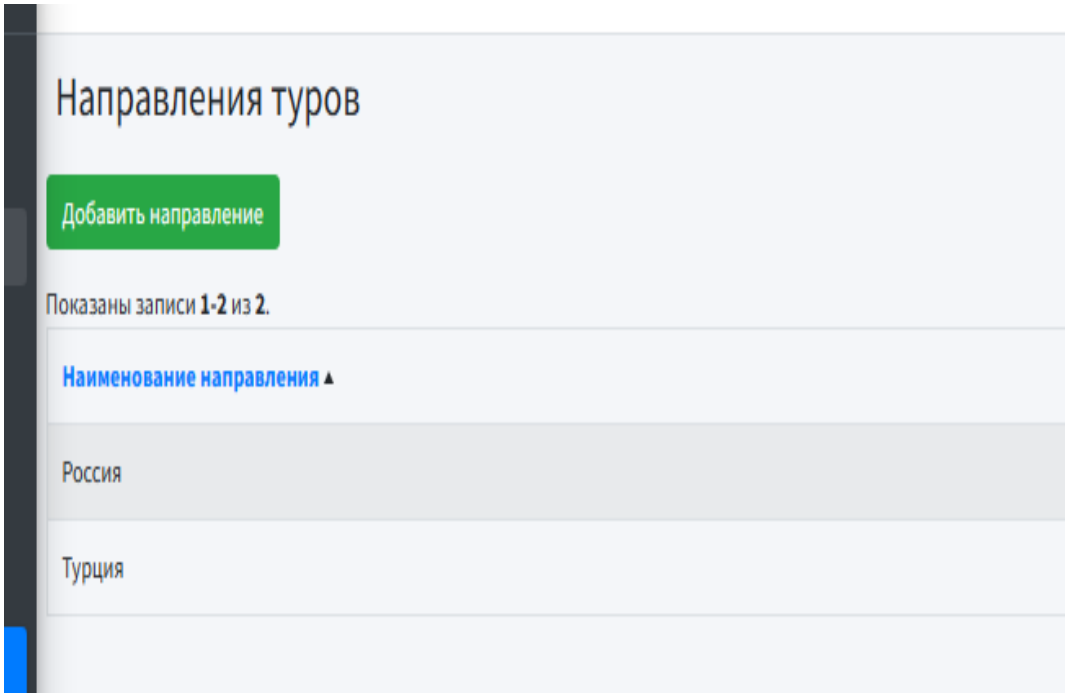


Рисунок 1.63 – Страница сайта с направлениями туров

Аналогичным образом заполняется информация о клиентах турагентства, желающих купить тур. При нажатии на кнопку «Добавить клиента» загружается страница с формой «Добавить клиента», кроме того, предусмотрена возможность, позволяющая загрузить сканкопию паспорта или другого документа, удостоверяющего личность клиента (рисунок 1.64).

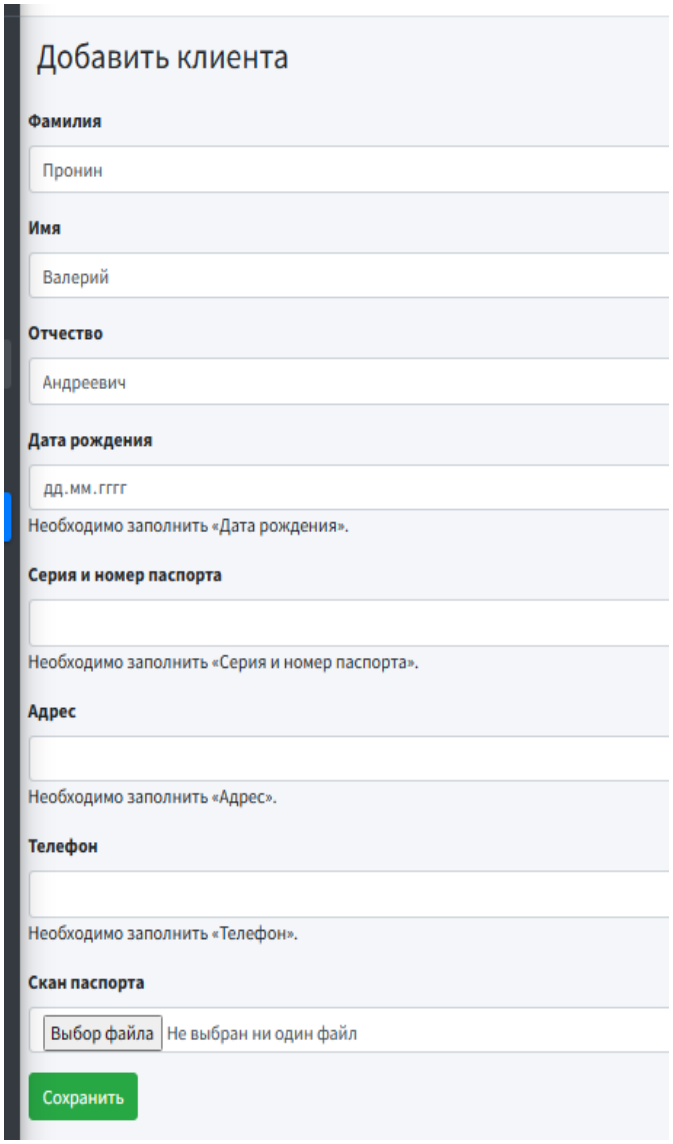


Рисунок 1.64 – Форма для добавления клиента турагентства

Кроме предоставления туров клиентам, система может фиксировать факт оплаты и хранить информацию о заключенных договорах. При добавлении информации система отслеживает правильность и полноту заполнения полей формы. При неверном заполнении или при неполном заполнении полей формы система выдает соответствующие сообщения об ошибках (рисунок 1.65).

Рисунок 1.65 – Форма заполнения информации об оплате тура

Результаты оплаты по туру сохраняются в журнале «Оплаты» (рисунок 1.66).

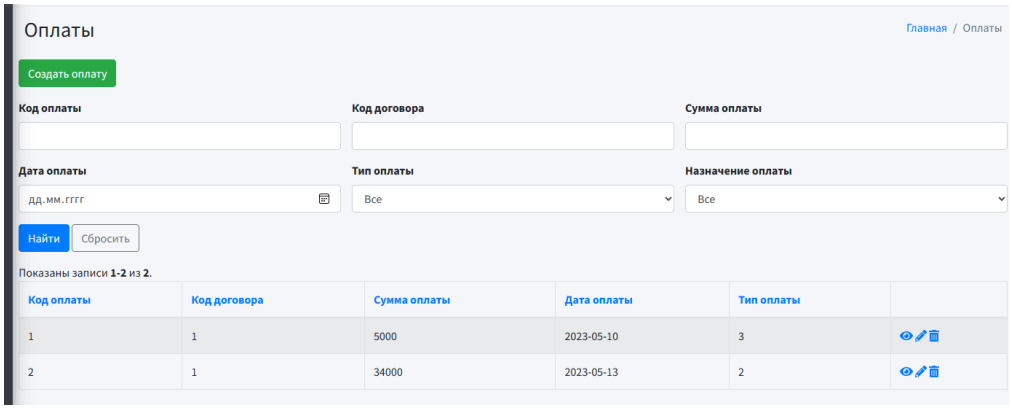


Рисунок 1.66 – Форма заполнения информации об оплате тура

Система позволяет хранить информацию о заключенных договорах на оказание услуг. Для занесения информации о договорах необходимо выбрать пункт главного меню «Журнал договоров» (рисунок 1.67).

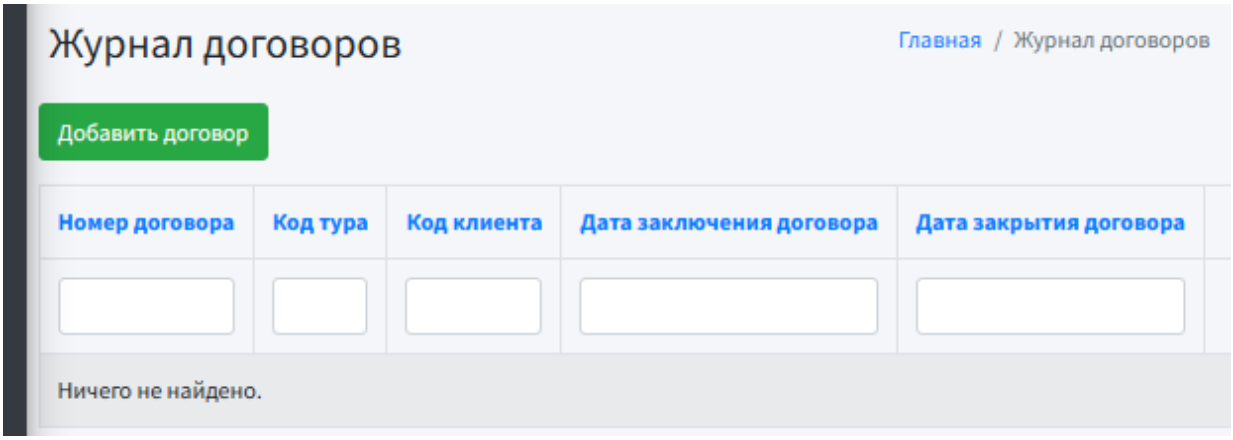


Рисунок 1.67 – Форма «Журнал договоров»

Для того, чтобы можно было работать в системе, необходимо пройти авторизацию, введя логин admin и пароль: 1q2w3e4r5t (рисунок 1.68).

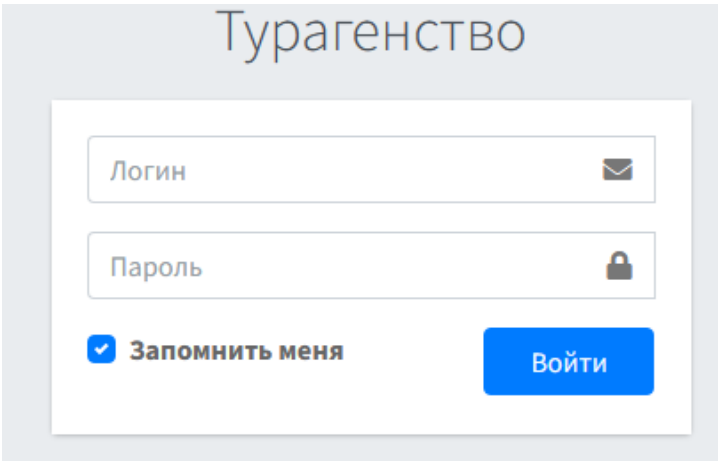


Рисунок 1.68 – Форма авторизации в системе

2 Эргономические особенности организации труда на рабочем месте оператора автоматизированной системы

2.1 Организации рабочих мест пользователей

При организации рабочих мест необходимо учитывать то, что конструкция рабочего места, его размеры и взаимное расположение его элементов должны соответствовать антропометрическим, физиологическим и психофизиологическим данным человека, а также характеру. При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м [10].

Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5 - 2,0 м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла)

следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Пример организации рабочего места и правильная позиция оператора за компьютером представлен на рисунке 2.1.

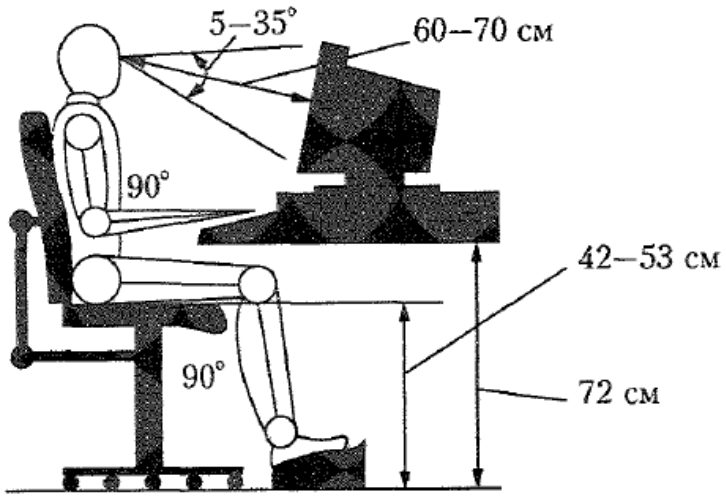


Рисунок 2.1 – Организация рабочего места

2.2 Зоны досягаемости

Важнейшими характеристиками рабочего пространства являются зоны досягаемости. Под досягаемой зоной понимается та часть пространства, которая ограничена крайними точками, достигаемыми руками и ногами оператора, сохраняющего свое положение неизменным. Оптимальная зона – это часть

пространства, находящаяся в пределах досягаемости середины ладони или пяты, при основном исходном положении тела оператора.

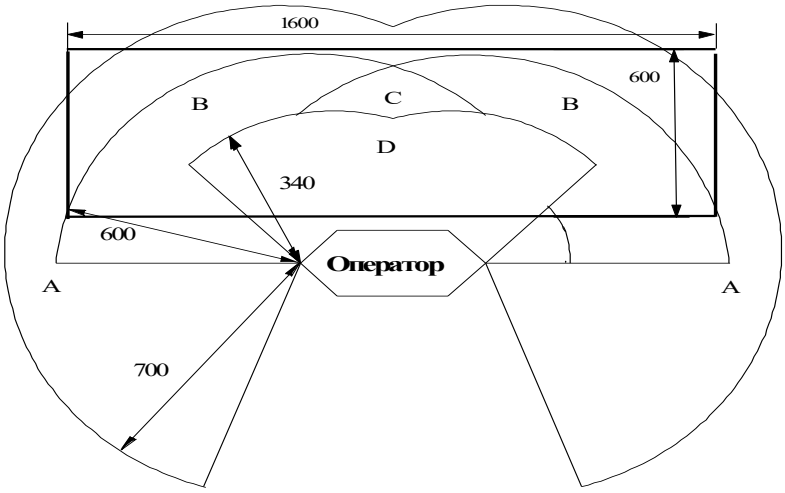


Рисунок 2.2 – Зоны досягаемости

Зоны досягаемости в горизонтальной плоскости представлены на рисунке (рисунок 2.2), где выделены зоны легкой досягаемости (B, C, D) и зона максимальной досягаемости (A). В зонах B и D возможны наиболее быстрые, точные, координированные и наименее утомляющие движения. Сектор досягаемости для каждой руки имеет угол 180 градусов. Оптимальному рабочему пространству соответствует зона D. При работе оператора в положении “сидя” рекомендуются следующие параметры рабочего пространства: ширина – не менее 700 мм; глубина – не менее 400 мм.

2.3 Требования к организации и оборудованию рабочих мест с ПЭВМ пользователей ЭВМ

Помещения для занятий оборудуются одноместными столами, предназначенными для работы с ПЭВМ.

Конструкция одноместного стола для работы с ПЭВМ должна предусматривать:

– две раздельные поверхности: одна горизонтальная для размещения ПЭВМ с плавной регулировкой по высоте в пределах 520 - 760 мм и вторая –

для клавиатуры с плавной регулировкой по высоте и углу наклона от 0 до 15 градусов с надежной фиксацией в оптимальном рабочем положении (12 - 15 градусов);

– ширину поверхностей для ВДТ и клавиатуры не менее 750 мм (ширина обеих поверхностей должна быть одинаковой) и глубину не менее 550 мм;

– опору поверхностей для ПЭВМ или ВДТ и для клавиатуры на стояк, в котором должны находится провода электропитания и кабель локальной сети. Основание стояка следует совмещать с подставкой для ног;

– отсутствие ящиков;

– увеличение ширины поверхностей до 1200 мм при оснащении рабочего места принтером.

Высота края стола, обращенного к работающему с ПЭВМ, и высота пространства для ног должны соответствовать росту обучающихся в обуви.

Линия взора должна быть перпендикулярна центру экрана и оптимальное ее отклонение от перпендикуляра, проходящего через центр экрана в вертикальной плоскости, не должно превышать +-5 градусов, допустимое +-10 градусов.

2.4 Системы и виды освещения

К системам производственного освещения предъявляются следующие требования:

– уровень освещенности на рабочем месте должен соответствовать характеру выполняемой зрительной работы;

– достаточно равномерное распределение яркости на рабочих поверхностях и в окружающем пространстве;

– отсутствие резких теней и бликов;

– постоянство освещения во времени;

– оптимальная направленность излучаемого осветительными приборами светового потока;

– долговечность, экономичность, электро- и пожаробезопасность, эстетичность, удобство и простота эксплуатации. В зависимости от природы источника световой энергии различают естественное, искусственное и совмещенное освещение [10].

Как правило, применяют одностороннее боковое естественное освещение. Причем свет проемы лучше располагать с северной, северо-западной иле северо-восточной стороны здания. В машинных залах рабочие места операторов, работающих с дисплеями, располагают подальше от окон и таким образом, чтобы оконные проемы находились с боку. Если экран дисплея обращен к оконному проему, необходимы специальные экранирующие устройства. Окна рекомендуется снабжать светорассеивающими шторами, регулируемыми жалюзи или солнцезащитной пленкой с детализированным покрытием.

В тех случаях, когда одного естественного освещения не достаточно, устраивают совмещенное освещение. При этом дополнительное искусственное освещение применяют не только в темное, но и в светлое время суток.

По конструктивному исполнению искусственное освещение может быть общим и комбинированным. При общем освещении все рабочие места получают освещение от общей осветительной установки. Комбинированное освещение наряду с общим включает местное освещение, сосредоточивающее световой поток непосредственно на рабочих местах. Применение одного местного освещения недопустимо, так как возникает необходимость частой переадаптации зрения, создаются глубокие и резкие тени и другие неблагоприятные факторы.

Для общего искусственного освещения следует использовать источники света с индексом цветопередачи Ra >= 85%.

Требования к освещению рабочих мест в помещениях, оснащенных персональными компьютерами составляет 500 лк.

Равномерность освещенности должна быть не менее 0,6 в основных помещениях. Коэффициент пульсации освещенности от общего искусственного освещения в помещениях, с рабочими местами, на которых эксплуатируется автоматизированная система не должен превышать нормативных значений, регламентируемых в зависимости от функционального назначения помещения. В помещениях различного функционального назначения, оборудованными ПК, коэффициент пульсации не должен превышать 5% [10].

Для обеспечения оптимальных условий зрительной работы операторов дисплейных устройств необходима определенная цветовая отделка помещений. Так, при восприятии информации на экране зеленого цвета целесообразно окрашивать стену, на которую направлен взгляд оператора, в оливково - зеленый цвет с r=0.4. Окраске поверхности следует придавать матовую фактуру.

Условия труда для каждого человека должны быть оптимальными. У офисных сотрудников должны быть свои условия, у производственных- свои. В целом же оптимальные условия труда включают в себя следующие факторы:

Кондиционирование – это тот параметр, который позволит создать комфортные условия труда в разные времена года (когда особенно жарко или особенно холодно). Очень плохо работать в жарком помещении и летом, и зимой, так как повышается утомляемость, снижается производительность труда, увеличивается вероятность простудных заболеваний. Кроме того, происходит перегрев организма, что ведет к увеличению заболеваемости сотрудников. Оптимальная температура помещения от 18 до 22°С. В зависимости от привычек сотрудников и времени года она может подбираться в указанном выше диапазоне. Отопительная система должна распространять тепло равномерно по помещению офиса, а в холодное время года дополнительное тепло может быть получено за счет кондиционера или электрокамина.

Заключение

В ходе подготовки выпускной квалификационной работы бакалавра была разработана и отлажена автоматизированная информационная система центра туристического агентства.

В работе был выполнен анализ предметной области. Разработаны логическое и физическое представление базы данных системы. Построены *UML* - диаграммы системы.

Программная реализация выполнена на языке программирования *PHP 7.4* (фреймворк *Yii2*), стандартизированного языка гипертекстовой разметки документов *HTML, Java Script, CSS*. Среда проектирование базы данных – *MySQL Workbench 8.0.33*, СУБД – *MySQL*.

Пошаговая отладка и тщательное тестирование показали, что разработанная система функционирует корректно.

Разработанное программное обеспечение позволило в значительной степени упростить и ускорить ведение учета заказов и клиентов, формировать необходимые отчетные данные на предприятии.

В разделе обеспечения безопасности жизнедеятельности были рассмотрены вопросы эргономическихих особенностей организации труда на рабочем месте оператора автоматизированной системы.

Таким образом, задание на бакалаврскую работу выполнено в полном объеме.

Список использованных источников

1. Федеральный закон «Об основах туристской деятельности в Российской Федерации» от 24.11.1996 N 132-ФЗ (ред. от 28.05.2022) // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения: 28.05.2023).

2. Кондрашова, Е. В. Организация деятельности туристского предприятия : учебно-методическое пособие / Е. В. Кондрашова. — Улан-Удэ : ВСГИК, 2020. — 84 с. — Текст : электронный // Лань : электронно-библиотечная система. — *URL*: *https://e.lanbook.com/book/215366* (дата обращения: 03.05.2023). — Режим доступа: для авториз. пользователей.

3. Ключков А. Что такое *MySQL*. *//ITResearch*. Аналитика российского рынка ИТ. - М.: 2001. - №7.

4. Дакетт Джон. *HTML* и *CSS*. Разработка и создание веб-сайтов// Издательство Эксмо, серия Мировой компьютерный бестселлер 2019 г., твердая обложка, 480 страниц.

5. Сафронов, М. Разработка веб-приложений в *Yii 2* / М. Сафронов. - М.: ДМК, 2019. - 392 с.

6. Иванов Д.Ю., Новиков Ф.А. Унифицированный язык моделирования UML. - М.: 2020. - 264с.

7. Буч Г., Рамбо Д., Якобсон И. Язык *UML*. Руководство пользователя. 2-е издание: пер. с англ. Н. Мухин. - М: ДМК Пресс, 2007. - 496с.

8. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. - М.: Вильям, 2003.

9. Колисниченко, Д.Н. *PHP* и *MySQL*. Разработка веб-приложений. Профессиональное программирование / Д.Н. Колисниченко. - СПб.: *BHV*, 2015. - 592 с.

10. СанПин 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды

обитания» // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения: 28.05.2023).

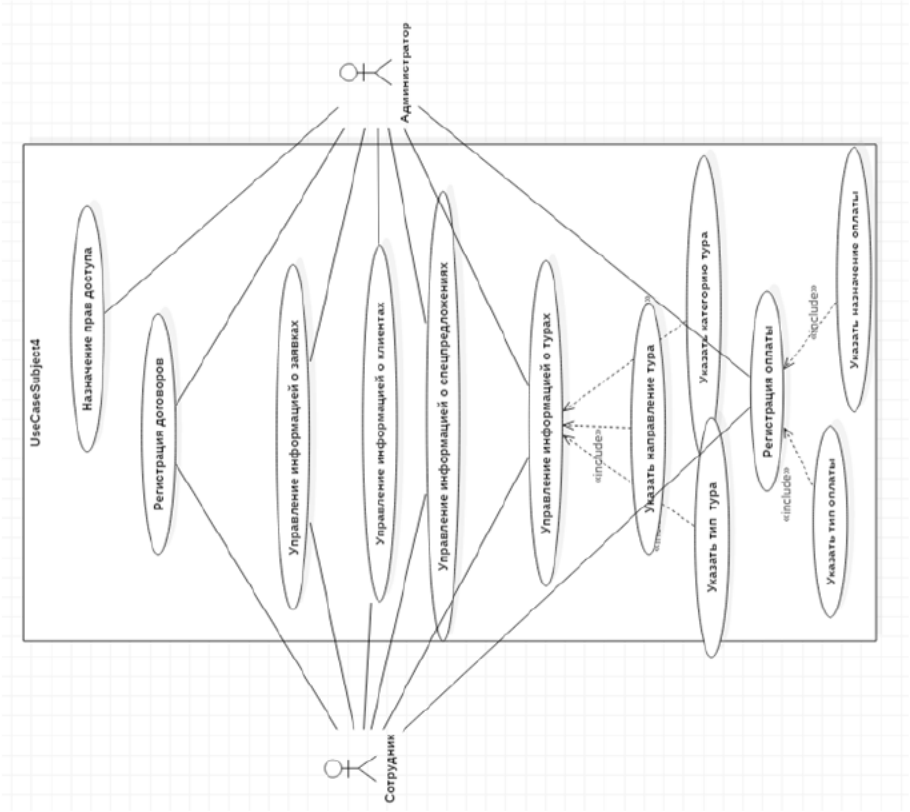
Приложение А

(Обязательное)

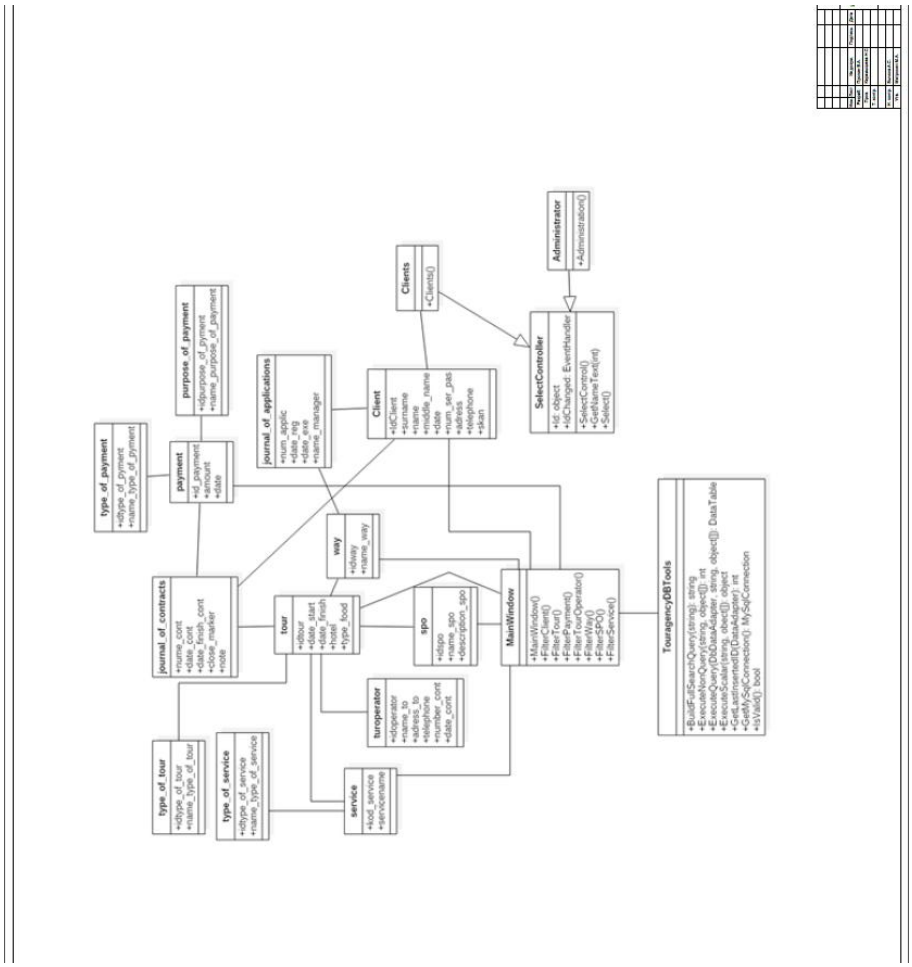
Информационная автоматизированная система туристического агентства

UML - диаграммы

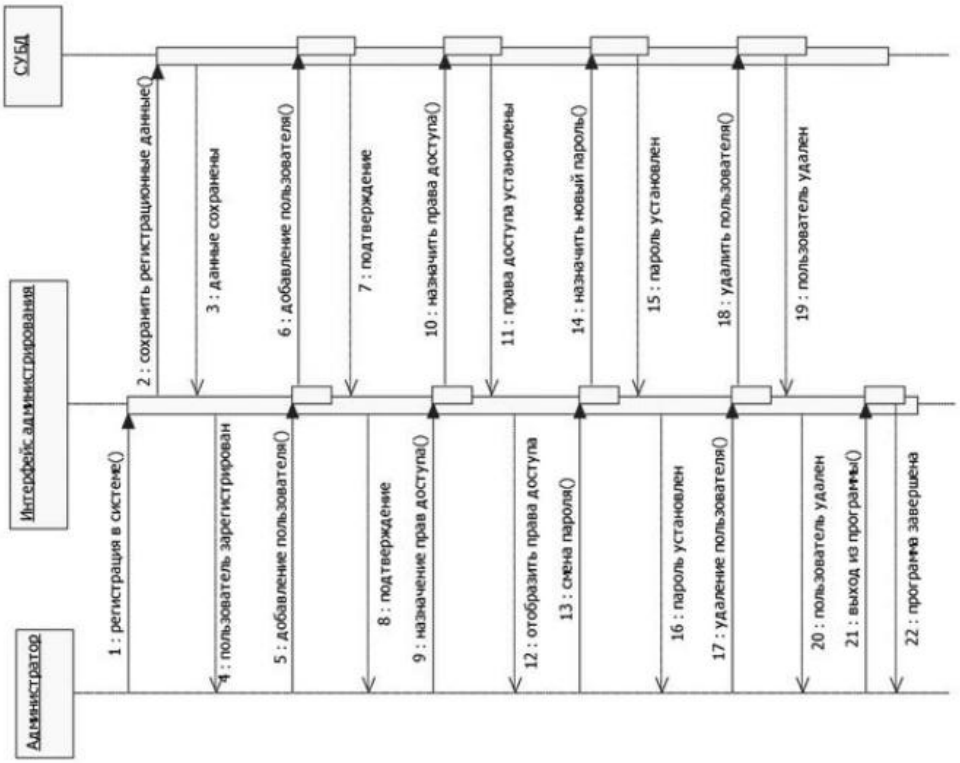
Приложение А.1 – Диаграмма прецедентов



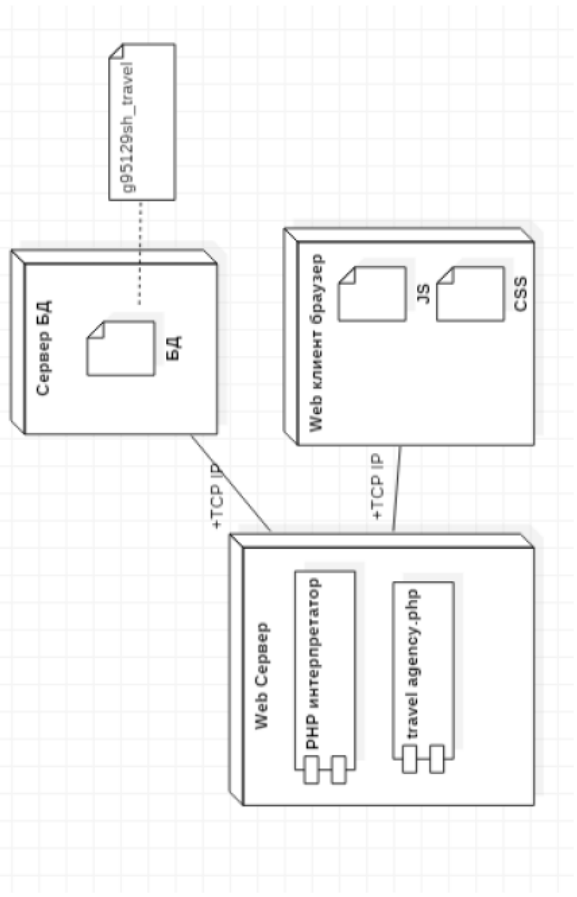
Приложение А.2 – Диаграмма классов



Приложение А.3 – Диаграмма последовательностей



Приложение А.4 – Диаграмма развертывания



Приложение Б

(Обязательное)

Информационная автоматизированная система туристического агентства

Листинг программы

Файл AppAsset.php

```
<?php
namespace app\assets;
use yii\web\AssetBundle;
class AppAsset extends AssetBundle
{
    public $basePath = '@webroot';
    public $baseUrl = '@web';
    public $css = ['css/site.css'];
    public $js = [];
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap4\BootstrapAsset'
    ];
}
```

Файл db.php

```
<?php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=g95129sh_travel',
    'username' => 'g95129sh_travel',
    'password' => 'bQ4e%5ne',
    'charset' => 'utf8',
];
```

Файл params.php

```
<?php
return [
    'adminEmail' => 'admin@example.com',
    'senderEmail' => 'noreply@example.com',
    'senderName' => 'Турагенство',
];
```

Файл *web*.php

```
<?php
$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';
$config = [
    'id' => 'basic',
    'name' => 'Турагенство',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'language' => 'ru-RU',
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'   => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [
            // !!! insert a secret key in the following (if it is empty) - this is required by
cookie validation
            'cookieValidationKey' => 'HmrchXj4kz8GH74QIvno9dqx3ZZcGZi3',
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
        'errorHandler' => [
            'errorAction' => 'site/error',
        ],
        'mailer' => [
            'class' => \yii\symfonymailer\Mailer::class,
            'viewPath' => '@app/mail',
            // send all mails to a file by default.
            'useFileTransport' => true,
        ],
        'log' => [
            'traceLevel' => YII_DEBUG ? 3 : 0,
            'targets' => [
                [
                    'class' => 'yii\log\FileTarget',
                    'levels' => ['error', 'warning'],
                ],
            ],
        ],
```

```
            'db' => $db,
            'urlManager' => [
                'suffix' => '/',
                'enablePrettyUrl' => true,
                'showScriptName' => false,
                'rules' => [
                    '<controller:[-a-z0-9_]+>/' => '<controller>/index ',
                    '<controller:[-a-z0-9_]+>/<id:\d+>' => '<controller>/view',
                    '<controller:[-a-z0-9_]+>/<action:[-a-z0-9_]+>/<id:\d+>'                    =>
'<controller>/<action>',
                    '<controller:[-a-z0-9_]+>/<action:[-a-z0-9_]+>' => '<controller>/<action>',
                ],
            ],
        ],
        'params' => $params,
    ];

    if (YII_ENV_DEV) {
        // configuration adjustments for 'dev' environment
        $config['bootstrap'][] = 'debug';
        $config['modules']['debug'] = [
            'class' => 'yii\debug\Module',
        ];
        $config['bootstrap'][] = 'gii';
        $config['modules']['gii'] = [
            'class' => 'yii\gii\Module',
        ];
    }
    return $config;
```

Файл ClientController.php

```
<?php
namespace app\controllers;
use app\models\Client;
use app\models\search\ClientSearch;
use yii\helpers\VarDumper;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\web\UploadedFile;
class ClientController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::class,
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $searchModel = new ClientSearch();
        $dataProvider = $searchModel->search($this->request->queryParams);

        return $this->render('index ', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }
    public function actionCreate()
    {
        $model = new Client();
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                return $this->redirect(['view', 'id' => $model->id]);
            }
        } else {
```

```
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);
            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                return $this->redirect(['view', 'id' => $model->id]);
            }
            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $this->findModel($id)->delete();
            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = Client::findOne(['id' => $id])) !== null) {
                return $model;
            }
            throw new NotFoundHttpException('The requested page does not exist.');
        }
        public function actionSkan($id){
            $model = $this->findModel($id);
     filename=печать.pdf');    //Skan_'.$model->surname.    '_'    .    $model->name.    '_'    .
$model->middle_name.'.pdf');

            $decoded = base64_decode($model->skan);
            $file = 'skan.pdf';
            file_put_contents($file, $decoded);

            if (file_exists($file)) {
                header('Content-Description: File Transfer');
                header('Content-Type: application/octet-stream');
                header('Content-Disposition: attachment; filename="'.basename($file).'"');
                header('Expires: 0');
                header('Cache-Control: must-revalidate');
                header('Pragma: public');
                header('Content-Length: ' . filesize($file));
                readfile($file);
                exit;
            }

        }
    }
```

Файл JournalOfApplicationsController.php

```
<?php
namespace app\controllers;
use app\models\JournalOfApplications;
use app\models\search\JournalOfApplicationsSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class JournalOfApplicationsController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $searchModel = new JournalOfApplicationsSearch();
```

```
            $dataProvider = $searchModel->search($this->request->queryParams);

            return $this->render('index ', [
                'searchModel' => $searchModel,
                'dataProvider' => $dataProvider,
            ]);
        }
        public function actionView($id)
        {
            return $this->render('view', [
                'model' => $this->findModel($id),
            ]);
        }
        public function actionCreate()
        {
            $model = new JournalOfApplications();

            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    return $this->redirect(['view', 'id' => $model->id]);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);

            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                return $this->redirect(['view', 'id' => $model->id]);
            }

            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $this->findModel($id)->delete();
            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = JournalOfApplications::findOne(['id' => $id])) !== null) {
                return $model;
            }

            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл JournalOfContractsController.php

```
    <?php
    namespace app\controllers;
    use app\models\JournalOfContracts;
    use app\models\search\JournalOfContractsSearch;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    class JournalOfContractsController extends Controller
    {
        public function behaviors()
        {
            return array_merge(
                parent::behaviors(),
                [
                    'verbs' => [
                        'class' => VerbFilter::className(),
                        'actions' => [
                            'delete' => ['POST'],
                        ],
                    ],
                ]
            );
        }
```

```
        public function actionIndex ()
        {
            $searchModel = new JournalOfContractsSearch();
            $dataProvider = $searchModel->search($this->request->queryParams);
            return $this->render('index ', [
                'searchModel' => $searchModel,
                'dataProvider' => $dataProvider,
            ]);
        }
        public function actionView($id)
        {
            return $this->render('view', [
                'model' => $this->findModel($id),
            ]);
        }
        public function actionCreate()
        {
            $model = new JournalOfContracts();
            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    return $this->redirect(['view', 'id' => $model->id]);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);
            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                return $this->redirect(['view', 'id' => $model->id]);
            }
            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $this->findModel($id)->delete();
            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = JournalOfContracts::findOne(['id' => $id])) !== null) {
                return $model;
            }

            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл PaymentController.php

```
    <?php
    namespace app\controllers;
    use app\models\Payment;
    use app\models\search\PaymentSearch;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    class PaymentController extends Controller
    {
        public function behaviors()
        {
            return array_merge(
                parent::behaviors(),
                [
                    'verbs' => [
                        'class' => VerbFilter::className(),
                        'actions' => [
                            'delete' => ['POST'],
                        ],
                    ],
                ]
            );
        }
        public function actionIndex ()
```

```
        {
            $searchModel = new PaymentSearch();
            $dataProvider = $searchModel->search($this->request->queryParams);
            return $this->render('index ', [
                'searchModel' => $searchModel,
                'dataProvider' => $dataProvider,
            ]);
        }
        public function actionView($id)
        {
            return $this->render('view', [
                'model' => $this->findModel($id),
            ]);
        }
        public function actionCreate()
        {
            $model = new Payment();

            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    return $this->redirect(['view', 'id' => $model->id]);
                }
            } else {
                $model->loadDefaultValues();
            }

            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);

            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                return $this->redirect(['view', 'id' => $model->id]);
            }

            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $this->findModel($id)->delete();

            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = Payment::findOne(['id' => $id])) !== null) {
                return $model;
            }
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл PurposeOfPaymentController.php

```
<?php
namespace app\controllers;
use app\models\PurposeOfPayment;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class PurposeOfPaymentController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
```

```
            );
        }
        public function action*Index* ()
        {
            $dataProvider = new ActiveDataProvider([
                'query' => PurposeOfPayment::find(),
                'sort' => [
                    'defaultOrder' => [
                        'name_purpose_of_payment' => SORT_ASC,
                    ]
                ],
            ]);
            return $this->render('*index* ', [
                'dataProvider' => $dataProvider,
            ]);
        }
        public function action*Create*()
        {
            $model = new PurposeOfPayment();

            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');

                    return $this->redirect(['*index* ']);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('*create*', [
                'model' => $model,
            ]);
        }
        public function action*Update*($id)
        {
            $model = $this->findModel($id);
            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                return $this->redirect(['*index* ']);
            }
            return $this->render('*update*', [
                'model' => $model,
            ]);
        }
        public function action*Delete*($id)
        {
            $model = $this->findModel($id);
            if ($model->*delete*())
                \Yii::$app->session->setFlash('success', 'Данные успешно удалены');

            return $this->redirect(['*index* ']);
        }
        protected function findModel($id)
        {
            if (($model = PurposeOfPayment::findOne(['id' => $id])) !== null) {
                return $model;
            }

            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл ServiceController.php

```
<?php
namespace app\controllers;
use app\models\search\ServiceSearch;
use app\models\Service;
use yii\*web*\Controller;
use yii\*web*\ForbiddenHttpException;
use yii\*web*\NotFoundHttpException;
use yii\filters\VerbFilter;
class ServiceController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
```

```
                            'class' => VerbFilter::class,
                            'actions' => [
                                'delete' => ['POST'],
                            ],
                        ],
                    ]
                );
            }
         string
             */
            public function actionIndex ()
            {
                $searchModel = new ServiceSearch();
                $dataProvider = $searchModel->search($this->request->queryParams);

                if (!$searchModel->type_of_service_id)
                    throw new ForbiddenHttpException('Не указан тип услуги');

                return $this->render('index ', [
                    'searchModel' => $searchModel,
                    'dataProvider' => $dataProvider,
                ]);
            }
            public function actionCreate()
            {
                $model = new Service();
                $model->type_of_service_id                                                    =
\Yii::$app->request->getQueryParam('type_of_service_id');
                if (!$model->type_of_service_id)
                    throw new ForbiddenHttpException('Не указан тип услуги');
                if ($this->request->isPost) {
                    if ($model->load($this->request->post()) && $model->save()) {
                        \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                        return  $this->redirect(['index  ',  'ServiceSearch[type_of_service_id]'  =>
$model->type_of_service_id]);
                    }
                } else {
                    $model->loadDefaultValues();
                }
                return $this->render('create', [
                    'model' => $model,
                ]);
            }
            public function actionUpdate($id)
            {
                $model = $this->findModel($id);
                if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                    \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');
                    return   $this->redirect(['index   ',   'ServiceSearch[type_of_service_id]'   =>
$model->type_of_service_id]);
                }
                return $this->render('update', [
                    'model' => $model,
                ]);
            }
            public function actionDelete($id)
            {
                $model = $this->findModel($id);
                if ($model->delete())
                    \Yii::$app->session->setFlash('success', 'Данные успешно удалены');
                return $this->redirect(['index ']);
            }
            protected function findModel($id)
            {
                if (($model = Service::findOne(['id' => $id])) !== null) {
                    return $model;
                }
                throw new NotFoundHttpException('The requested page does not exist.');
            }
        }
```

Файл SiteController.php

```
<?php
namespace app\controllers;
use Yii;
class SiteController extends Controller
{
    public function actions()
    {
```

```
            return [
                'error' => [
                    'class' => 'yii\web\ErrorAction',
                ],
            ];
        }
        public function actionIndex ()
        {
            return $this->render('index ');
        }
    }
```

Файл SpoController.php

```
    <?php
    namespace app\controllers;
    use app\models\Spo;
    use yii\data\ActiveDataProvider;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    class SpoController extends Controller
    {
        public function behaviors()
        {
            return array_merge(
                parent::behaviors(),
                [
                    'verbs' => [
                        'class' => VerbFilter::class,
                        'actions' => [
                            'delete' => ['POST'],
                        ],
                    ],
                ]
            );
        }
        public function actionIndex ()
        {
            $dataProvider = new ActiveDataProvider([
                'query' => Spo::find(),
                'sort' => [
                    'defaultOrder' => [
                        'name_spo' => SORT_ASC,
                    ]
                ],
            ]);
            return $this->render('index ', [
                'dataProvider' => $dataProvider,
            ]);
        }
        public function actionCreate()
        {
            $model = new Spo();
            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');

                    return $this->redirect(['index ']);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);

            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
    {
                \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');

                return $this->redirect(['index ']);
            }
            return $this->render('update', [
                'model' => $model,
            ]);
        }
```

```
    public function actionDelete($id)
    {
        $model = $this->findModel($id);
        if ($model->delete())
            \Yii::$app->session->setFlash('success', 'Данные успешно удалены');
        return $this->redirect(['index ']);
    }
    protected function findModel($id)
    {
        if (($model = Spo::findOne(['id' => $id])) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
}
```

Файл TourCategoryController.php

```
<?php
namespace app\controllers;
use app\models\TourCategory;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class TourCategoryController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::class,
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => TourCategory::find(),
            'sort' => [
                'defaultOrder' => [
                    'name_category' => SORT_ASC
                ],
                'attributes' => ['name_category']
            ],
        ]);
        return $this->render('index ', [
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionCreate()
    {
        $model = new TourCategory();
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                return $this->redirect(['index ']);
            }
        } else {
            $model->loadDefaultValues();
        }
        return $this->render('create', [
            'model' => $model,
        ]);
    }
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
            \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');
            return $this->redirect(['index ']);
        }
        return $this->render('update', [
            'model' => $model,
```

```
        ]);
    }
    public function action*Delete*($id)
    {
        $model = $this->findModel($id);
        if ($model->*delete*())
            \Yii::$app->session->setFlash('success', 'Данные успешно удалены');
        return $this->redirect(['*index* ']);
    }
    protected function findModel($id)
    {
        if (($model = TourCategory::findOne(['id' => $id])) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
}
```

Файл PurposeOfPaymentController.php

```
<?php
namespace app\controllers;
use app\models\PurposeOfPayment;
use yii\data\ActiveDataProvider;
use yii\*web*\Controller;
use yii\*web*\NotFoundHttpException
```

Файл TourController.php

```
<?php
namespace app\controllers;
use app\models\Tour;
use app\models\search\TourSearch;
use yii\*web*\Controller;
use yii\*web*\NotFoundHttpException;
use yii\filters\VerbFilter;
class TourController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        '*delete*' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function action*Index* ()
    {
        $searchModel = new TourSearch();
        $dataProvider = $searchModel->search($this->request->queryParams);

        return $this->render('*index* ', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    public function action*View*($id)
    {
        return $this->render('*view*', [
            'model' => $this->findModel($id),
        ]);
    }
    public function action*Create*()
    {
        $model = new Tour();
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                return $this->redirect(['*view*', 'id' => $model->id]);
            }
        } else {
            $model->loadDefaultValues();
        }

        return $this->render('*create*', [
```

```
                'model' => $model,
            ]);
        }
        public function action*Update*($id)
        {
            $model = $this->findModel($id);
            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                return $this->redirect(['*view*', 'id' => $model->id]);
            }
            return $this->render('*update*', [
                'model' => $model,
            ]);
        }
        public function action*Delete*($id)
        {
            $model = $this->findModel($id);
            if ($model->*delete*())
                \Yii::$app->session->setFlash('success', 'Данные успешно удалены');
            return $this->redirect(['*index* ']);
        }
        protected function findModel($id)
        {
            if (($model = Tour::findOne(['id' => $id])) !== null) {
                return $model;
            }
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл TouroperatorController.php

```
<?php
namespace app\controllers;
use app\models\Touroperator;
use app\models\search\TouroperatorSearch;
use yii\*web*\Controller;
use yii\*web*\NotFoundHttpException;
use yii\filters\VerbFilter;
class TouroperatorController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        '*delete*' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function action*Index* ()
    {
        $searchModel = new TouroperatorSearch();
        $dataProvider = $searchModel->search($this->request->queryParams);
        return $this->render('*index* ', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    public function action*View*($id)
    {
        return $this->render('*view*', [
            'model' => $this->findModel($id),
        ]);
    }
    public function action*Create*()
    {
        $model = new Touroperator();
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');

                return $this->redirect(['*view*', 'id' => $model->id]);
            }
```

```
                } else {
                    $model->loadDefaultValues();
                }
                return $this->render('create', [
                    'model' => $model,
                ]);
            }
            public function actionUpdate($id)
            {
                $model = $this->findModel($id);
                if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                    \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                    return $this->redirect(['view', 'id' => $model->id]);
                }
                return $this->render('update', [
                    'model' => $model,
                ]);
            }
            public function actionDelete($id)
            {
                $this->findModel($id)->delete();
                return $this->redirect(['index ']);
            }
            protected function findModel($id)
            {
                if (($model = Touroperator::findOne(['id' => $id])) !== null) {
                    return $model;
                }
                throw new NotFoundHttpException('The requested page does not exist.');
            }
        }
```

Файл TypeOfPaymentController.php

```
        <?php
namespace app\controllers;
use app\models\TypeOfPayment;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;.
class TypeOfPaymentController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => TypeOfPayment::find(),
            'sort' => [
                'defaultOrder' => [
                    'name_type_of_payment' => SORT_ASC,
                ]
            ],
        ]);

        return $this->render('index ', [
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionCreate()
    {
        $model = new TypeOfPayment();

        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                return $this->redirect(['index ']);
```

```
            }
        } else {
            $model->loadDefaultValues();
        }
        return $this->render('create', [
            'model' => $model,
        ]);
    }
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        if ($this->request->isPost && $model->load($this->request->post()) && $model->save()) {
            \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
            return $this->redirect(['index ']);
        }
        return $this->render('update', [
            'model' => $model,
        ]);
    }
    public function actionDelete($id)
    {
        $model = $this->findModel($id);
        if ($model->delete())
            \Yii::$app->session->setFlash('success', 'Данные успешно удалены');

        return $this->redirect(['index ']);
    }
    protected function findModel($id)
    {
        if (($model = TypeOfPayment::findOne(['id' => $id])) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
}
```

Файл TypeOfServiceController.php

```
<?php
namespace app\controllers;
use app\models\TypeOfService;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class TypeOfServiceController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::class,
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => TypeOfService::find(),
            'sort' => [
                'defaultOrder' => [
                    'name_type_of_service' => SORT_ASC,
                ]
            ],
        ]);
        return $this->render('index ', [
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionCreate()
    {
        $model = new TypeOfService();
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
```

```
                \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');

                return $this->redirect(['index ']);
            }
        } else {
            $model->loadDefaultValues();
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
            \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');

            return $this->redirect(['index ']);
        }
        return $this->render('update', [
            'model' => $model,
        ]);
    }
    public function actionDelete($id)
    {
        $model = $this->findModel($id);
        if ($model->delete())
            \Yii::$app->session->setFlash('success', 'Данные успешно удалены');

        return $this->redirect(['index ']);
    }
    protected function findModel($id)
    {
        if (($model = TypeOfService::findOne(['id' => $id])) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
}
```

Файл TypeOfTourController.php

```
<?php
namespace app\controllers;
use app\models\TypeOfTour;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class TypeOfTourController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::class,
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => TypeOfTour::find(),
            'sort' => [
                'defaultOrder' => [
                    'name_type_of_tour' => SORT_ASC
                ]
            ],
        ]);
        return $this->render('index ', [
            'dataProvider' => $dataProvider,
        ]);
```

```
        }
        public function actionCreate()
        {
            $model = new TypeOfTour();
            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                    return $this->redirect(['index ']);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);
            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');

                return $this->redirect(['index ']);
            }
            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $model = $this->findModel($id);
            if ($model->delete())
                \Yii::$app->session->setFlash('success', 'Данные успешно удалены');
            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = TypeOfTour::findOne(['id' => $id])) !== null) {
                return $model;
            }
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл WayController.php

```
<?php
namespace app\controllers;
use app\models\Way;
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class WayController extends Controller
{
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::class,
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }
    public function actionIndex ()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => Way::find(),
            'sort' => [
                'defaultOrder' => [
                    'name_way' => SORT_ASC,
                ]
            ],
        ]);
```

```
            return $this->render('index ', [
                'dataProvider' => $dataProvider,
            ]);
        }
        public function actionCreate()
        {
            $model = new Way();
            if ($this->request->isPost) {
                if ($model->load($this->request->post()) && $model->save()) {
                    \Yii::$app->session->setFlash('success', 'Данные успешно сохранены');
                    return $this->redirect(['index ']);
                }
            } else {
                $model->loadDefaultValues();
            }
            return $this->render('create', [
                'model' => $model,
            ]);
        }
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);

            if ($this->request->isPost && $model->load($this->request->post()) && $model->save())
{
                \Yii::$app->session->setFlash('success', 'Данные успешно обновлнены');
                return $this->redirect(['index ']);
            }
            return $this->render('update', [
                'model' => $model,
            ]);
        }
        public function actionDelete($id)
        {
            $model = $this->findModel($id);
            if ($model->delete())
                \Yii::$app->session->setFlash('success', 'Данные успешно удалены');


            return $this->redirect(['index ']);
        }
        protected function findModel($id)
        {
            if (($model = Way::findOne(['id' => $id])) !== null) {
                return $model;
            }
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
```

Файл m230501_193549_*create*_client_table.php

```
<?php
use yii\db\Migration;
class m230501_193549_create_client_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%client}}', [
            'id' => $this->primaryKey()->comment('Личный номер клиента'),
            'surname' => $this->string()->notNull()->comment('Фамилия'),
            'name' => $this->string()->notNull()->comment('Имя'),
            'middle_name' => $this->string()->comment('Отчество'),
            'date' => $this->date()->notNull()->comment('Дата рождения'),
            'num_ser_pas' => $this->integer()->notNull()->comment('Серия и номер паспорта'),
            'address' => $this->string()->notNull()->comment('Адрес'),
            'telephone' => $this->string()->notNull()->comment('Телефон'),
            'skan' => $this->binary(16777215)->comment('Скан паспорта')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%client}}');
    }
}
```

Файл m230501_193706_*create*_tour_category_table.php

```
<?php
use yii\db\Migration;
class m230501_193706_create_tour_category_table extends Migration
{
```

```
            public function safeUp()
            {
                $this->createTable('{{%tour_category}}', [
                    'id' => $this->primaryKey()->comment('Код категории тура'),
                    'name_category'  =>  $this->string()->notNull()->comment('Наименование  категории
тура')
                ]);
            }
            public function safeDown()
            {
                $this->dropTable('{{%tour_category}}');
            }
        }
```

Файл m230501_193717_*create*_tour_table.php

```
        <?php
        use yii\db\Migration;
        class m230501_193717_create_tour_table extends Migration
        {
            public function safeUp()
            {
                $this->createTable('{{%tour}}', [
                    'id' => $this->primaryKey(),
                    'date_start' => $this->date()->notNull()->comment('Дата начала'),
                    'date_finish' => $this->date()->notNull()->comment('Дата окончания'),
                    'hotel' => $this->string()->notNull()->comment('Место проживания'),
                    'type_food' => $this->string()->notNull()->comment('Тип питания'),
                    'tour_category_id'=> $this->integer()->notNull()->comment('Категории тура'),
                    'way_id'=> $this->integer()->notNull()->comment('Направление тура'),
                    'touroperator_id'=> $this->integer()->notNull()->comment('Туроператор'),
                    'type_id' => $this->integer()->notNull()->comment('Вид тура'),
                ]);
            }
            public function safeDown()
            {
                $this->dropTable('{{%tour}}');
            }
        }
```

Файл m230501_194428_*create*_type_of_tour_table.php

```
        <?php
        use yii\db\Migration;
        class m230501_194428_create_type_of_tour_table extends Migration
        {
            public function safeUp()
            {
                $this->createTable('{{%type_of_tour}}', [
                    'id' => $this->primaryKey(),
                    'name_type_of_tour'  =>  $this->string()->notNull()->comment('Наименование  вида
тура'),
                ]);
            }
            public function safeDown()
            {
                $this->dropTable('{{%type_of_tour}}');
            }
        }
```

Файл m230501_194618_*create*_service_table.php

```
        <?php
        use yii\db\Migration;
        class m230501_194618_create_service_table extends Migration
        {
            public function safeUp()
            {
                $this->createTable('{{%service}}', [
                    'id' => $this->primaryKey(),
                    'type_of_service_id' => $this->integer()->notNull()->comment('Код типа услуги'),
                    'name_service' => $this->string()->notNull()->comment('Наименование услуги'),
                    'description_service' => $this->string()->notNull()->comment('Описание услуги'),
                ]);
            }
            public function safeDown()
            {
                $this->dropTable('{{%service}}');
            }
        }
```

Файл m230501_194656_*create*_type_of_service_table.php

```
        <?php
```

```
use yii\db\Migration;
class m230501_194656_create_type_of_service_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%type_of_service}}', [
            'id' => $this->primaryKey(),
            'name_type_of_service' => $this->string()->notNull()->comment('Наименование вида
услуги')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%type_of_service}}');
    }
}
```

Файл m230501_195809_*create*_services_tour_table.php

```
<?php
use yii\db\Migration;
class m230501_195809_create_services_tour_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%services_tour}}', [
            'id' => $this->primaryKey(),
            'tour_id' => $this->integer()->notNull()->comment('Код тура'),
            'service_id' => $this->integer()->notNull()->comment('Код услуги')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%services_tour}}');
    }
}
```

Файл m230501_195836_*create*_touroperator_table.php

```
<?php
use yii\db\Migration;
class m230501_195836_create_touroperator_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%touroperator}}', [
            'id' => $this->primaryKey(),
            'name' => $this->string()->notNull()->comment('Наименование туроператора'),
            'address' => $this->string()->notNull()->comment('Адрес'),
            'telephone' => $this->string()->notNull()->comment('Телефон'),
            'num_contract' => $this->integer()->notNull()->comment('Номер договора'),
            'date_contract' => $this->date()->notNull()->comment('Дата догвора'),
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%touroperator}}');
    }
}
```

Файл m230501_195905_*create*_journal_of_contracts_table.php

```
<?php
use yii\db\Migration;
class m230501_195905_create_journal_of_contracts_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%journal_of_contracts}}', [
            'id' => $this->primaryKey(),
            'tour_id' => $this->integer()->notNull()->comment('Код тура'),
            'client_id' => $this->integer()->notNull()->comment('Код клиента'),
            'date_contract' => $this->date()->notNull()->comment('Дата заключения договора'),
            'date_finish_contract'   =>   $this->date()->notNull()->comment('Дата   закрытия
договора'),
            'close_marker' => $this->binary()->comment('Отметка о закрытии'),
            'note' => $this->string()->comment('Примечание')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%journal_of_contracts}}');
    }
}
```

Файл m230501_195931_*create*_payment_table.php

```
<?php
use yii\db\Migration;
class m230501_195931_create_payment_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%payment}}', [
            'id' => $this->primaryKey(),
            'contract_id' => $this->integer()->notNull()->comment('Код договора'),
            'amount' => $this->integer()->notNull()->unsigned()->comment('Сумма оплаты'),
            'date' => $this->date()->notNull()->comment('Дата оплаты'),
            'type_of_payment_id' => $this->integer()->notNull()->comment('Тип оплаты'),
            'purpose_of_payment_id'    =>    $this->integer()->notNull()->comment('Назначение
оплаты'),
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%payment}}');
    }
}
```

Файл m230501_195955_*create*_type_of_payment_table.php

```
<?php
use yii\db\Migration;
class m230501_195955_create_type_of_payment_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%type_of_payment}}', [
            'id' => $this->primaryKey(),
            'name_type_of_payment' => $this->string()->notNull()->comment('Наименование типа
оплаты')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%type_of_payment}}');
    }
}
```

Файл m230501_200023_*create*_purpose_of_payment_table.php

```
<?php
use yii\db\Migration;
class m230501_200023_create_purpose_of_payment_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%purpose_of_payment}}', [
            'id' => $this->primaryKey(),
            'name_purpose_of_payment'  =>  $this->string()->notNull()->comment('Наименование
назначения оплаты')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%purpose_of_payment}}');
    }
}
```

Файл m230501_200036_*create*_way_table.php

```
<?php
use yii\db\Migration;
class m230501_200036_create_way_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%way}}', [
            'id' => $this->primaryKey(),
            'name_way' => $this->string()->notNull()->comment('Наименование направления'),
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%way}}');
    }
}
```

Файл m230501_200149_*create*_journal_of_applications_table.php

```
<?php
use yii\db\Migration;
class m230501_200149_create_journal_of_applications_table extends Migration
```

```
{
    public function safeUp()
    {
        $this->createTable('{{%journal_of_applications}}', [
            'id' => $this->primaryKey(),
            'client_id' => $this->integer()->notNull()->comment('Код клиента'),
            'way_id' => $this->integer()->notNull()->comment('Код направления'),
            'date_reg' => $this->date()->notNull()->comment('Дата регистрации заявки'),
            'date_exe' => $this->date()->notNull()->comment('Дата выполнения заявки'),
            'name_manager' => $this->string()->comment('Имя менеджера'),
            'note' => $this->string()->comment('Примечание')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%journal_of_applications}}');
    }
}
```

Файл m230501_200220_*create*_spo_table.php

```
<?php
use yii\db\Migration;
class m230501_200220_create_spo_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%spo}}', [
            'id' => $this->primaryKey(),
            'name_spo' => $this->string()->notNull()->comment('Наименование СПО'),
            'description_spo' => $this->string()->notNull()->comment('Описание СПО')
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%spo}}');
    }
}
```

Файл m230501_200258_*create*_spo_on_the_tour_table.php

```
<?php
use yii\db\Migration;
class m230501_200258_create_spo_on_the_tour_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%spo_on_the_tour}}', [
            'id' => $this->primaryKey(),
            'spo_id' => $this->integer()->notNull()->comment('Код СПО'),
            'tour_id' => $this->integer()->notNull()->comment('Код тура')
        ]);
        $this->addForeignKey('{{%fk-tour-tour_category_id}}', '{{%tour}}',
'tour_category_id', '{{%tour_category}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-tour-way_id}}', '{{%tour}}', 'way_id', '{{%way}}', 'id',
'CASCADE');
        $this->addForeignKey('{{%fk-tour-touroperator_id}}', '{{%tour}}', 'touroperator_id',
'{{%touroperator}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-tour-type_id}}', '{{%tour}}', 'type_id',
'{{%type_of_tour}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-service-type_of_service_id}}', '{{%service}}',
'type_of_service_id', '{{%type_of_service}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-services_tour-tour_id}}', '{{%services_tour}}',
'tour_id', '{{%tour}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-services_tour-service_id}}', '{{%services_tour}}',
'service_id', '{{%service}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-journal_of_contracts-tour_id}}',
'{{%journal_of_contracts}}', 'tour_id', '{{%tour}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-journal_of_contracts-client_id}}',
'{{%journal_of_contracts}}', 'client_id', '{{%client}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-payment-contract_id}}', '{{%payment}}', 'contract_id',
'{{%journal_of_contracts}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-payment-type_of_payment_id}}', '{{%payment}}',
'type_of_payment_id', '{{%type_of_payment}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-payment-purpose_of_payment_id}}', '{{%payment}}',
'purpose_of_payment_id', '{{%purpose_of_payment}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-journal_of_applications-client_id}}',
'{{%journal_of_applications}}', 'client_id', '{{%client}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-journal_of_applications-way_id}}',
'{{%journal_of_applications}}', 'way_id', '{{%way}}', 'id', 'CASCADE');
        $this->addForeignKey('{{%fk-spo_on_the_tour-tour_id}}', '{{%spo_on_the_tour}}',
'tour_id', '{{%tour}}', 'id', 'CASCADE');
```

```
            $this->addForeignKey('{{%fk-spo_on_the_tour-spo_id}}',       '{{%spo_on_the_tour}}',
'spo_id', '{{%spo}}', 'id', 'CASCADE');
        }
        public function safeDown()
        {
            $this->dropTable('{{%spo_on_the_tour}}');
        }
    }
```

Файл ClientSearch.php

```
    <?php
    namespace app\models\search;
    use yii\base\Model;
    use yii\data\ActiveDataProvider;
    use app\models\Client;
    class ClientSearch extends Client
    {
        public function rules()
        {
            return [
                [['id', 'num_ser_pas'], 'integer'],
                [['surname',  'name',  'middle_name',  'date',  'address',  'telephone',  'skan'],
'safe'],
            ];
        }
        public function search($params)
        {
            $query = Client::find();
            $dataProvider = new ActiveDataProvider([
                'query' => $query,
            ]);
            $this->load($params);
            if (!$this->validate()) {
                return $dataProvider;
            }
            $query->andFilterWhere([
                'id' => $this->id,
                'date' => $this->date,
                'num_ser_pas' => $this->num_ser_pas,
            ]);
            $query->andFilterWhere(['like', 'surname', $this->surname])
                ->andFilterWhere(['like', 'name', $this->name])
                ->andFilterWhere(['like', 'middle_name', $this->middle_name])
                ->andFilterWhere(['like', 'address', $this->address])
                ->andFilterWhere(['like', 'telephone', $this->telephone])
                ->andFilterWhere(['like', 'skan', $this->skan]);

            return $dataProvider;
        }
    }
```

Файл JournalOfApplicationsSearch.php

```
    <?php
    namespace app\models\search;
    use yii\base\Model;
    use yii\data\ActiveDataProvider;
    use app\models\JournalOfApplications;
    class JournalOfApplicationsSearch extends JournalOfApplications
    {
        public function rules()
        {
            return [
                [['id', 'client_id', 'way_id'], 'integer'],
                [['date_reg', 'date_exe', 'name_manager', 'note'], 'safe'],
            ];
        }
        public function search($params)
        {
            $query = JournalOfApplications::find();
            $dataProvider = new ActiveDataProvider([
                'query' => $query,
            ]);
            $this->load($params);
            if (!$this->validate()) {
                return $dataProvider;
            }
            $query->andFilterWhere([
                'id' => $this->id,
                'client_id' => $this->client_id,
                'way_id' => $this->way_id,
                'date_reg' => $this->date_reg,
```

```
                'date_exe' => $this->date_exe,
            ]);
            $query->andFilterWhere(['like', 'name_manager', $this->name_manager])
                ->andFilterWhere(['like', 'note', $this->note]);
            return $dataProvider;
        }
    }
```

Файл JournalOfContractsSearch.php

```
<?php
namespace app\models\search;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\JournalOfContracts;
class JournalOfContractsSearch extends JournalOfContracts
{
    public function rules()
    {
        return [
            [['id', 'tour_id', 'client_id'], 'integer'],
            [['date_contract', 'date_finish_contract', 'close_marker', 'note'], 'safe'],
        ];
    }
    public function search($params)
    {
        $query = JournalOfContracts::find();
        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);
        $this->load($params);
        if (!$this->validate()) {
            return $dataProvider;
        }
        $query->andFilterWhere([
            'id' => $this->id,
            'tour_id' => $this->tour_id,
            'client_id' => $this->client_id,
            'date_contract' => $this->date_contract,
            'date_finish_contract' => $this->date_finish_contract,
        ]);
        $query->andFilterWhere(['like', 'close_marker', $this->close_marker])
            ->andFilterWhere(['like', 'note', $this->note]);
        return $dataProvider;
    }
}
```

Файл PaymentSearch.php

```
<?php
namespace app\models\search;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\Payment;
class PaymentSearch extends Payment
{
    public function rules()
    {
        return [
            [['id', 'contract_id', 'amount', 'type_of_payment_id', 'purpose_of_payment_id'],
'integer'],
            [['date'], 'safe'],
        ];
    }
    public function search($params)
    {
        $query = Payment::find();
        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);
        $this->load($params);
        if (!$this->validate()) {
            return $dataProvider;
        }
        $query->andFilterWhere([
            'id' => $this->id,
            'contract_id' => $this->contract_id,
            'amount' => $this->amount,
            'date' => $this->date,
            'type_of_payment_id' => $this->type_of_payment_id,
            'purpose_of_payment_id' => $this->purpose_of_payment_id,
        ]);
        return $dataProvider;
```

```
    }
}
```

Файл ServiceSearch.php

```
<?php
namespace app\models\search;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\Models\Service;
class ServiceSearch extends Service
{
    public function rules()
    {
        return [
            [['id', 'type_of_service_id'], 'integer'],
            [['name_service', 'description_service'], 'safe'],
        ];
    }
    public function scenarios()
    {
        return Model::scenarios();
    }
    public function search($params)
    {
        $query = Service::find();
        $dataProvider = new ActiveDataProvider([
            'query' => $query,
            'sort' => ['defaultOrder' => ['name_service' => SORT_ASC]]
        ]);
        $this->load($params);
        if (!$this->validate()) {
            return $dataProvider;
        }
        $query->andFilterWhere([
            'id' => $this->id,
            'type_of_service_id' => $this->type_of_service_id,
        ]);
        $query->andFilterWhere(['like', 'name_service', $this->name_service])
            ->andFilterWhere(['like', 'description_service', $this->description_service]);
        return $dataProvider;
    }
}
```

Файл TouroperatorSearch.php

```
<?php
namespace app\models\search;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\Touroperator;
class TouroperatorSearch extends Touroperator
{
    public function rules()
    {
        return [
            [['id', 'num_contract'], 'integer'],
            [['name', 'address', 'telephone', 'date_contract'], 'safe'],
        ];
    }
    public function search($params)
    {
        $query = Touroperator::find();
        $dataProvider = new ActiveDataProvider([
            'query' => $query,
            'sort' => [
                'defaultOrder' => ['name' => SORT_ASC]
            ]
        ]);
        $this->load($params);
        if (!$this->validate()) {
            return $dataProvider;
        }
        $query->andFilterWhere(['like', 'name', $this->name])
            ->andFilterWhere(['like', 'telephone', $this->telephone]);
        return $dataProvider;
    }
}
```

Файл TourSearch.php

```
<?php
namespace app\models\search;
use yii\base\Model;
use yii\data\ActiveDataProvider;
```

```
use app\models\Tour;
class TourSearch extends Tour
{
    public function rules()
    {
        return [
            [['id', 'tour_category_id', 'way_id', 'touroperator_id', 'type_id'], 'integer'],
            [['date_start', 'date_finish', 'hotel', 'type_food'], 'safe'],
        ];
    }
    public function search($params)
    {
        $query = Tour::find();
        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);
        $this->load($params);
        if (!$this->validate()) {
            return $dataProvider;
        }
        $query->andFilterWhere([
            'date_start' => $this->date_start,
            'date_finish' => $this->date_finish,
            'tour_category_id' => $this->tour_category_id,
            'way_id' => $this->way_id,
            'touroperator_id' => $this->touroperator_id,
            'type_id' => $this->type_id,
        ]);
        $query->andFilterWhere(['like', 'hotel', $this->hotel])
            ->andFilterWhere(['like', 'type_food', $this->type_food]);
        return $dataProvider;
    }
}
```

Файл Client.php

```
<?php
namespace app\models;
use Yii;
use yii\web\UploadedFile;
class Client extends \yii\db\ActiveRecord
{
    public $file;
    public static function tableName()
    {
        return 'client';
    }
    public function rules()
    {
        return [
            [['surname', 'name', 'date', 'num_ser_pas', 'address', 'telephone'], 'required'],
            [['date'], 'safe'],
            [['num_ser_pas'], 'integer'],
            [['skan'], 'safe'],
            [['surname', 'name', 'middle_name', 'address', 'telephone'], 'string', 'max' =>
255],
            [['file'], 'file']
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Личный номер клиента',
            'surname' => 'Фамилия',
            'name' => 'Имя',
            'middle_name' => 'Отчество',
            'date' => 'Дата рождения',
            'num_ser_pas' => 'Серия и номер паспорта',
            'address' => 'Адрес',
            'telephone' => 'Телефон',
            'skan' => 'Скан паспорта',
        ];
    }
    public function getJournalOfApplications()
    {
        return $this->hasMany(JournalOfApplications::class, ['client_id' => 'id']);
    }
    public function getJournalOfContracts()
    {
        return $this->hasMany(JournalOfContracts::class, ['client_id' => 'id']);
    }
```

```
            public function beforeSave($insert)
            {
                if($file=UploadedFile::getInstance($this,'file'))
                {
                    $this->skan = file_get_contents($file->tempName);
                    $this->skan  = base64_encode($this->skan );
                }
                return parent::beforeSave($insert);
            }
        }
```

Файл JournalOfApplications.php

```
        <?php
        namespace app\models;
        use Yii;
        class JournalOfApplications extends \yii\db\ActiveRecord
        {
            public static function tableName()
            {
                return 'journal_of_applications';
            }
            public function rules()
            {
                return [
                    [['client_id', 'way_id', 'date_reg', 'date_exe'], 'required'],
                    [['client_id', 'way_id'], 'integer'],
                    [['date_reg', 'date_exe'], 'safe'],
                    [['name_manager', 'note'], 'string', 'max' => 255],
                    [['client_id'], 'exist', 'skipOnError' => true, 'targetClass' => Client::class,
'targetAttribute' => ['client_id' => 'id']],
                    [['way_id'], 'exist', 'skipOnError' => true, 'targetClass' => Way::class,
'targetAttribute' => ['way_id' => 'id']],
                ];
            }
            public function attributeLabels()
            {
                return [
                    'id' => 'Номер заявки',
                    'client_id' => 'Код клиента',
                    'way_id' => 'Код направления',
                    'date_reg' => 'Дата регистрации заявки',
                    'date_exe' => 'Дата выполнения заявки',
                    'name_manager' => 'Имя менеджера',
                    'note' => 'Примечание',
                ];
            }
            public function getClient()
            {
                return $this->hasOne(Client::class, ['id' => 'client_id']);
            }
            public function getWay()
            {
                return $this->hasOne(Way::class, ['id' => 'way_id']);
            }
        }
```

Файл JournalOfContracts.php

```
        <?php
        namespace app\models;
        use Yii;
        class JournalOfContracts extends \yii\db\ActiveRecord
        {
            public static function tableName()
            {
                return 'journal_of_contracts';
            }
            public function rules()
            {
                return [
                    [['tour_id', 'client_id', 'date_contract', 'date_finish_contract'], 'required'],
                    [['tour_id', 'client_id'], 'integer'],
                    [['date_contract', 'date_finish_contract'], 'safe'],
                    [['close_marker'], 'string'],
                    [['note'], 'string', 'max' => 255],
                    [['client_id'], 'exist', 'skipOnError' => true, 'targetClass' => Client::class,
'targetAttribute' => ['client_id' => 'id']],
                    [['tour_id'], 'exist', 'skipOnError' => true, 'targetClass' => Tour::class,
'targetAttribute' => ['tour_id' => 'id']],
                ];
            }
```

```
            public function attributeLabels()
            {
                return [
                    'id' => 'Номер договора',
                    'tour_id' => 'Код тура',
                    'client_id' => 'Код клиента',
                    'date_contract' => 'Дата заключения договора',
                    'date_finish_contract' => 'Дата закрытия договора',
                    'close_marker' => 'Отметка о закрытии',
                    'note' => 'Примечание',
                ];
            }
            public function getClient()
            {
                return $this->hasOne(Client::class, ['id' => 'client_id']);
            }
            public function getPayments()
            {
                return $this->hasMany(Payment::class, ['contract_id' => 'id']);
            }
            public function getTour()
            {
                return $this->hasOne(Tour::class, ['id' => 'tour_id']);
            }
        }
```

Файл PurposeOfPayment.php

```
        <?php
        namespace app\models;
        use Yii;
        class PurposeOfPayment extends \yii\db\ActiveRecord
        {
            public static function tableName()
            {
                return 'purpose_of_payment';
            }
            public function rules()
            {
                return [
                    [['name_purpose_of_payment'], 'required'],
                    [['name_purpose_of_payment'], 'string', 'max' => 255],
                ];
            }
            public function attributeLabels()
            {
                return [
                    'id' => 'Код назначения оплаты',
                    'name_purpose_of_payment' => 'Наименование назначения оплаты',
                ];
            }
            public function getPayments()
            {
                return $this->hasMany(Payment::class, ['purpose_of_payment_id' => 'id']);
            }
        }
```

Файл Payment.php

```
        <?php
        namespace app\models;
        use Yii;
        class Payment extends \yii\db\ActiveRecord
        {
            public static function tableName()
            {
                return 'payment';
            }
            public function rules()
            {
                return [
                    [['contract_id', 'amount', 'date', 'type_of_payment_id', 'purpose_of_payment_id'],
'required'],
                    [['contract_id',   'amount',   'type_of_payment_id',   'purpose_of_payment_id'],
'integer'],
                    [['date'], 'safe'],
                    [['contract_id'],   'exist',   'skipOnError'   =>   true,   'targetClass'   =>
JournalOfContracts::class, 'targetAttribute' => ['contract_id' => 'id']],
                    [['purpose_of_payment_id'], 'exist', 'skipOnError' => true, 'targetClass' =>
PurposeOfPayment::class, 'targetAttribute' => ['purpose_of_payment_id' => 'id']],
```

```
                [['type_of_payment_id'], 'exist', 'skipOnError' => true, 'targetClass' => TypeOfPayment::class, 'targetAttribute' => ['type_of_payment_id' => 'id']],
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'Код оплаты',
                'contract_id' => 'Код договора',
                'amount' => 'Сумма оплаты',
                'date' => 'Дата оплаты',
                'type_of_payment_id' => 'Тип оплаты',
                'purpose_of_payment_id' => 'Назначение оплаты',
            ];
        }
        public function getContract()
        {
            return $this->hasOne(JournalOfContracts::class, ['id' => 'contract_id']);
        }
        public function getPurposeOfPayment()
        {
            return $this->hasOne(PurposeOfPayment::class, ['id' => 'purpose_of_payment_id']);
        }
        public function getTypeOfPayment()
        {
            return $this->hasOne(TypeOfPayment::class, ['id' => 'type_of_payment_id']);
        }
    }
```

Файл Service.php

```
    <?php
    namespace app\models;
    use Yii;
    class Service extends \yii\db\ActiveRecord
    {
        public static function tableName()
        {
            return 'service';
        }
        public function rules()
        {
            return [
                [['type_of_service_id', 'name_service', 'description_service'], 'required'],
                [['type_of_service_id'], 'integer'],
                [['name_service', 'description_service'], 'string', 'max' => 255],
                [['type_of_service_id'], 'exist', 'skipOnError' => true, 'targetClass' => TypeOfService::class, 'targetAttribute' => ['type_of_service_id' => 'id']],
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'ID',
                'type_of_service_id' => 'Код типа услуги',
                'name_service' => 'Наименование услуги',
                'description_service' => 'Описание услуги',
            ];
        }
        public function getServicesTours()
        {
            return $this->hasMany(ServicesTour::class, ['service_id' => 'id']);
        }
        public function getTypeOfService()
        {
            return $this->hasOne(TypeOfService::class, ['id' => 'type_of_service_id']);
        }
    }
```

Файл PurposeOfPaymentController.php

```
    <?php
    namespace app\controllers;
    use app\models\PurposeOfPayment;
    use yii\data\ActiveDataProvider;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException
```

Файл ServicesTour.php

```
    <?php
    namespace app\models;
    use Yii;
```

```
class ServicesTour extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'services_tour';
    }
    public function rules()
    {
        return [
            [['tour_id', 'service_id'], 'required'],
            [['tour_id', 'service_id'], 'integer'],
            [['service_id'], 'exist', 'skipOnError' => true, 'targetClass' => Service::class,
'targetAttribute' => ['service_id' => 'id']],
            [['tour_id'], 'exist', 'skipOnError' => true, 'targetClass' => Tour::class,
'targetAttribute' => ['tour_id' => 'id']],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'tour_id' => 'Код тура',
            'service_id' => 'Код услуги',
        ];
    }
    public function getService()
    {
        return $this->hasOne(Service::class, ['id' => 'service_id']);
    }
    public function getTour()
    {
        return $this->hasOne(Tour::class, ['id' => 'tour_id']);
    }
}
```

Файл Spo.php

```
<?php
namespace app\models;
use Yii;
class Spo extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'spo';
    }
    public function rules()
    {
        return [
            [['name_spo', 'description_spo'], 'required'],
            [['name_spo', 'description_spo'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name_spo' => 'Наименование СПО',
            'description_spo' => 'Описание СПО',
        ];
    }
    public function getSpoOnTheTours()
    {
        return $this->hasMany(SpoOnTheTour::class, ['spo_id' => 'id']);
    }
}
```

Файл SpoOnTheTour.php

```
<?php
namespace app\models;
use Yii;
class SpoOnTheTour extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'spo_on_the_tour';
    }
    public function rules()
    {
        return [
            [['spo_id', 'tour_id'], 'required'],
            [['spo_id', 'tour_id'], 'integer'],
```

```
                [['spo_id'], 'exist', 'skipOnError' => true, 'targetClass' => Spo::class,
'targetAttribute' => ['spo_id' => 'id']],
                [['tour_id'], 'exist', 'skipOnError' => true, 'targetClass' => Tour::class,
'targetAttribute' => ['tour_id' => 'id']],
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'ID',
                'spo_id' => 'Spo ID',
                'tour_id' => 'Tour ID',
            ];
        }
        public function getSpo()
        {
            return $this->hasOne(Spo::class, ['id' => 'spo_id']);
        }
        public function getTour()
        {
            return $this->hasOne(Tour::class, ['id' => 'tour_id']);
        }
    }
```

Файл Tour.php

```
    <?php
    namespace app\models;
    use yii\helpers\ArrayHelper;
    class Tour extends \yii\db\ActiveRecord
    {
        private $_serviceItems;
        private $_spoItems;
        public static function tableName()
        {
            return 'tour';
        }
        public function rules()
        {
            return [
                [['date_start', 'date_finish', 'hotel', 'type_food', 'tour_category_id', 'way_id',
'touroperator_id', 'type_id'], 'required'],
                [['date_start', 'date_finish'], 'safe'],
                [['tour_category_id', 'way_id', 'touroperator_id', 'type_id'], 'integer'],
                [['hotel', 'type_food'], 'string', 'max' => 255],
                [['tour_category_id'], 'exist', 'skipOnError' => true, 'targetClass' =>
TourCategory::class, 'targetAttribute' => ['tour_category_id' => 'id']],
                [['touroperator_id'], 'exist', 'skipOnError' => true, 'targetClass' =>
Touroperator::class, 'targetAttribute' => ['touroperator_id' => 'id']],
                [['type_id'], 'exist', 'skipOnError' => true, 'targetClass' => TypeOfTour::class,
'targetAttribute' => ['type_id' => 'id']],
                [['way_id'], 'exist', 'skipOnError' => true, 'targetClass' => Way::class,
'targetAttribute' => ['way_id' => 'id']],
                [['serviceItems', 'spoItems'], 'safe']
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'Код тура',
                'date_start' => 'Дата начала',
                'date_finish' => 'Дата окончания',
                'hotel' => 'Место проживания',
                'type_food' => 'Тип питания',
                'tour_category_id' => 'Категории тура',
                'way_id' => 'Направление тура',
                'touroperator_id' => 'Туроператор',
                'type_id' => 'Вид тура',
                'serviceItems' => 'Услуги тура',
                'spoItems' => 'СПО по туру'
            ];
        }
        public function getJournalOfContracts()
        {
            return $this->hasMany(JournalOfContracts::class, ['tour_id' => 'id']);
        }
        public function getServicesTours()
        {
            return $this->hasMany(ServicesTour::class, ['tour_id' => 'id']);
        }
        public function getServices()
```

```
    {
        return $this->hasMany(Service::class, ['id' => 'service_id'])->via('servicesTours');
    }
    public function getSpoOnTheTours()
    {
        return $this->hasMany(SpoOnTheTour::class, ['tour_id' => 'id']);
    }
    public function getSpo()
    {
        return $this->hasMany(Spo::class, ['id' => 'spo_id'])->via('spoOnTheTours');
    }
    public function getTourCategory()
    {
        return $this->hasOne(TourCategory::class, ['id' => 'tour_category_id']);
    }
    public function getTouroperator()
    {
        return $this->hasOne(Touroperator::class, ['id' => 'touroperator_id']);
    }
    public function getType()
    {
        return $this->hasOne(TypeOfTour::class, ['id' => 'type_id']);
    }
    public function getWay()
    {
        return $this->hasOne(Way::class, ['id' => 'way_id']);
    }
    public function getServiceItems()
    {
        $items = [];
        if ($this->servicesTours) {
            $items = ArrayHelper::getColumn($this->servicesTours, 'service_id');
        }
        return $items;
    }
    public function setServiceItems($serviceItems): void
    {
        $this->_serviceItems = $serviceItems;
    }
    public function getSpoItems()
    {
        $items = [];
        if ($this->spoOnTheTours) {
            $items = ArrayHelper::getColumn($this->spoOnTheTours, 'spo_id');
        }
        return $items;
    }
    public function setSpoItems($spoItems): void
    {
        $this->_spoItems = $spoItems;
    }
    public function saveServiceItems($serviceItems){
        $this->unlinkAll('servicesTours', true);
        if ($serviceItems) {
            foreach ($serviceItems as $service_id) {
                $this->link('servicesTours', new ServicesTour(['service_id' => $service_id]));
            }
        }
    }
    public function saveSpoItems($spoItems){
        $this->unlinkAll('spoOnTheTours', true);
        if ($spoItems) {
            foreach ($spoItems as $spo_id) {
                $this->link('spoOnTheTours', new SpoOnTheTour(['spo_id' => $spo_id]));
            }
        }
    }
    public function afterSave($insert, $changedAttributes)
    {
        $this->saveServiceItems($this->_serviceItems);
        $this->saveSpoItems($this->_spoItems);

        parent::afterSave($insert, $changedAttributes); // TODO: Change the autogenerated stub
    }
}
```

Файл PurposeOfPaymentController.php

```
<?php
namespace app\controllers;
use app\models\PurposeOfPayment;
```

```
use yii\data\ActiveDataProvider;
use yii\web\Controller;
use yii\web\NotFoundHttpException
```

Файл TourCategory.php

```
<?php
namespace app\models;
use Yii;
class TourCategory extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'tour_category';
    }
    public function rules()
    {
        return [
            [['name_category'], 'required'],
            [['name_category'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Код категории тура',
            'name_category' => 'Наименование категории тура',
        ];
    }
    public function getTours()
    {
        return $this->hasMany(Tour::class, ['tour_category_id' => 'id']);
    }
}
```

Файл Touroperator.php

```
<?php

namespace app\models;
use Yii;
class Touroperator extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'touroperator';
    }
    public function rules()
    {
        return [
            [['name', 'address', 'telephone', 'num_contract', 'date_contract'], 'required'],
            [['num_contract'], 'integer'],
            [['date_contract'], 'safe'],
            [['name', 'address', 'telephone'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Код туроператора',
            'name' => 'Наименование туроператора',
            'address' => 'Адрес',
            'telephone' => 'Телефон',
            'num_contract' => 'Номер договора',
            'date_contract' => 'Дата догвора',
        ];
    }
    public function getTours()
    {
        return $this->hasMany(Tour::class, ['touroperator_id' => 'id']);
    }
}
```

Файл TypeOfPayment.php

```
<?php
namespace app\models;
use Yii;
class TypeOfPayment extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'type_of_payment';
    }
```

```
    public function rules()
    {
        return [
            [['name_type_of_payment'], 'required'],
            [['name_type_of_payment'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Код типа оплаты',
            'name_type_of_payment' => 'Наименование типа оплаты',
        ];
    }
    public function getPayments()
    {
        return $this->hasMany(Payment::class, ['type_of_payment_id' => 'id']);
    }
}
```

Файл TypeOfService.php

```
<?php
namespace app\models;
use Yii;
class TypeOfService extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'type_of_service';
    }
    public function rules()
    {
        return [
            [['name_type_of_service'], 'required'],
            [['name_type_of_service'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name_type_of_service' => 'Наименование вида услуги',
        ];
    }
    public function getServices()
    {
        return $this->hasMany(Service::class, ['type_of_service_id' => 'id']);
    }
}
```

Файл TypeOfTour.php

```
<?php
namespace app\models;
use Yii;
class TypeOfTour extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'type_of_tour';
    }
    public function rules()
    {
        return [
            [['name_type_of_tour'], 'required'],
            [['name_type_of_tour'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name_type_of_tour' => 'Наименование вида тура',
        ];
    }
    public function getTours()
    {
        return $this->hasMany(Tour::class, ['type_id' => 'id']);
    }
}
```

Файл Way.php

```php
<?php
namespace app\models;
use Yii;
class Way extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'way';
    }
    public function rules()
    {
        return [
            [['name_way'], 'required'],
            [['name_way'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name_way' => 'Наименование направления',
        ];
    }
    public function getJournalOfApplications()
    {
        return $this->hasMany(JournalOfApplications::class, ['way_id' => 'id']);
    }
    public function getTours()
    {
        return $this->hasMany(Tour::class, ['way_id' => 'id']);
    }
}
```

Файл *index* .php

```php
<?php
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');
require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/../vendor/yiisoft/yii2/Yii.php';
$config = require __DIR__ . '/../config/web.php';
(new yii\web\Application($config))->run();
```

Приложение В

(Обязательное)

Информационная автоматизированная система туристического агентства

Презентация

Тема работы:
” Информационная автоматизированная система туристического агентства.”

Автор работы: студент группы 19ВВП1 Пронин В.А.

Научный руководитель: к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.

вка задачи

ы: разработать автоматизированную информ
нтра туристического сервиса.

адачи:

едметной области.

ование системы.

ия информационной автоматизированной сис
ского агентства.

ічи

інка готовых программных продуктов;

струментов разработки;

а требований к автоматизированной информ

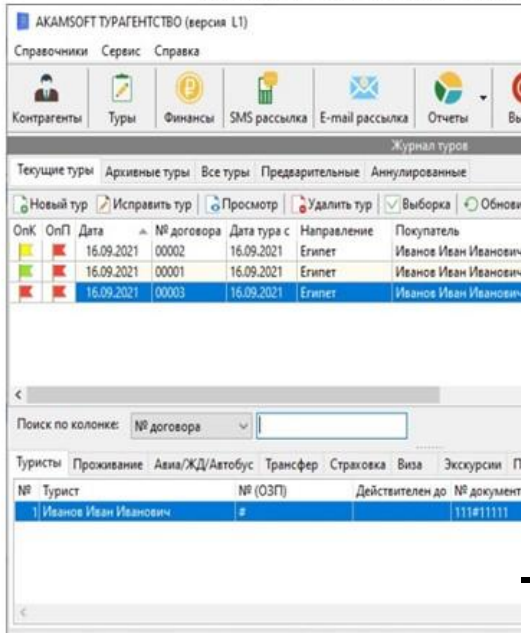
ование структуры базы данных;

а программного обеспечения.

вые программные продукты

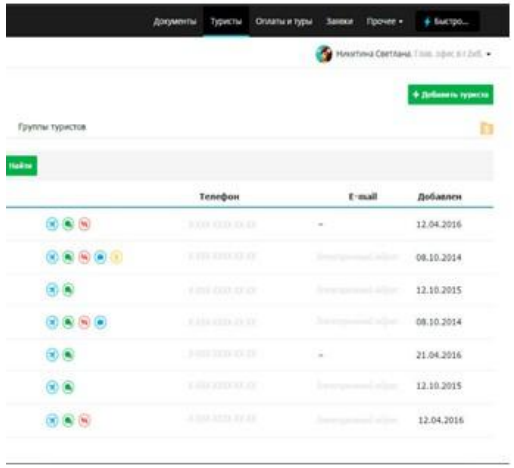
Само-ТурАгент

AKAMSOFT ТУРАГЕНТ

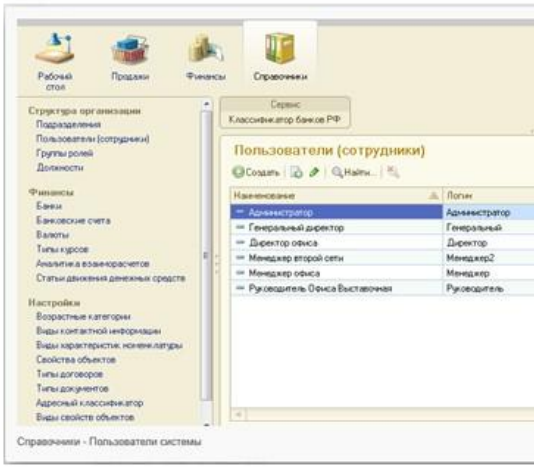


товые программные продукты

окументы-Туризм



ERP.travel



и системы

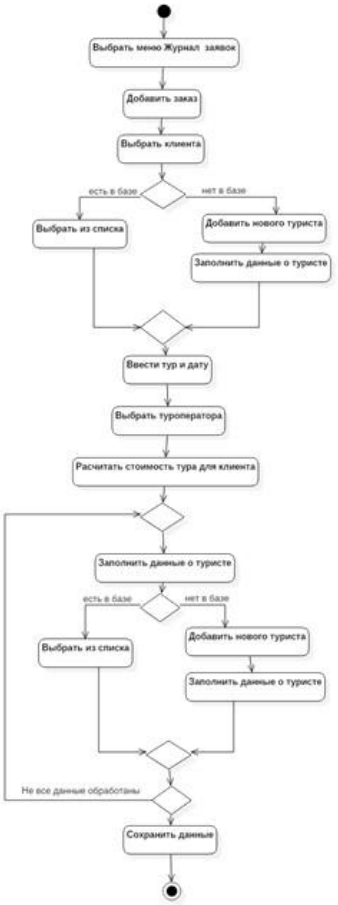
авторизации пользователей системы.
ний о клиентах.
ний о заявках.
ы заказов.
ний о турах (видах, направлениях туров, ти
туров).
ний об услугах по туру.
ний о спецпредложениях.
ний о туроператорах туров.
рнала договоров (контрактов).
ы за тура (оплата, тип оплаты, назначение опл

ма
ов
ования

UseCaseSubject4
Назначение прав доступа
Регистрация договоров
Управление информацией о заявках
Управление информацией о клиентах
Управление информацией о спецпредложениях
Управление информацией о турах
«include»
Указать направление тура
Указать тип тура
Указать категорию тура
Регистрация оплаты
«include»
«include»
Указать тип оплаты
Указать назначение оплаты
Сотрудник

с заказами

заказа



Добавле

лма
овательности

администратора

ıа
ательности

ı сотрудника

Диаграмма классов

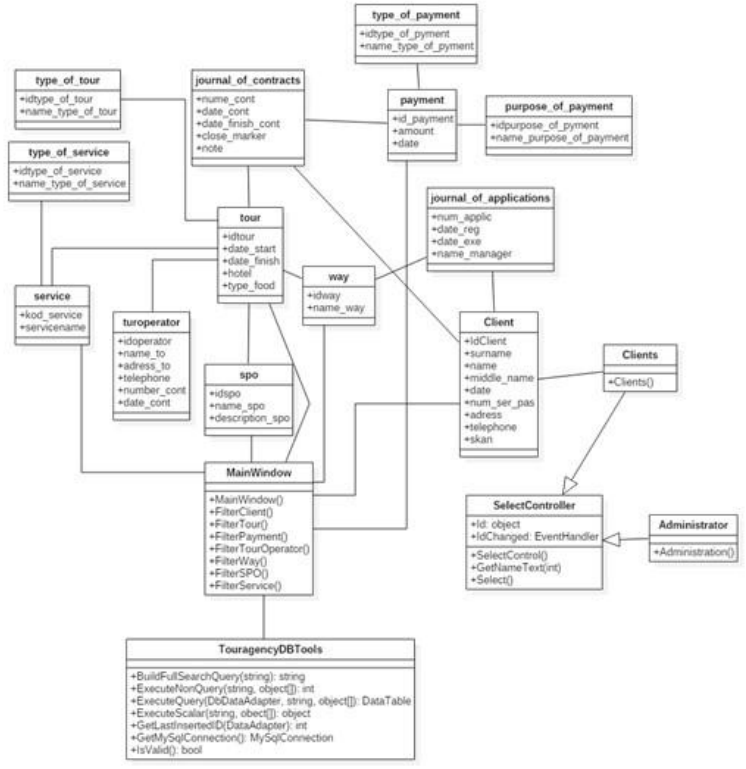
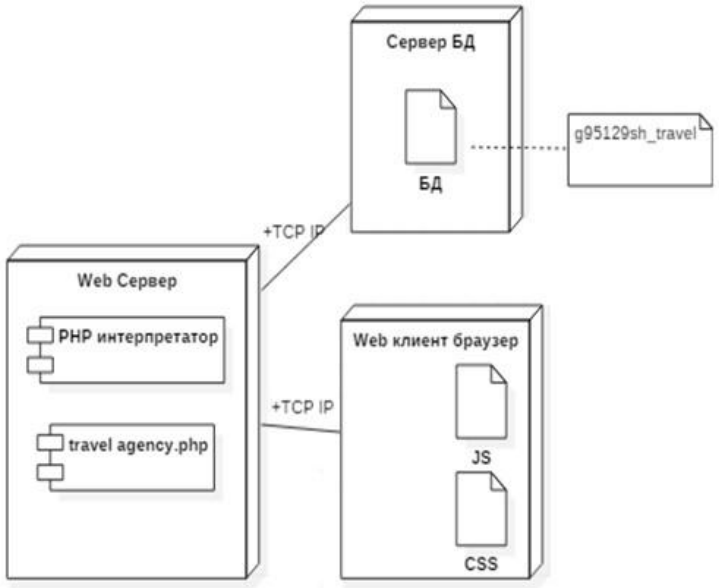
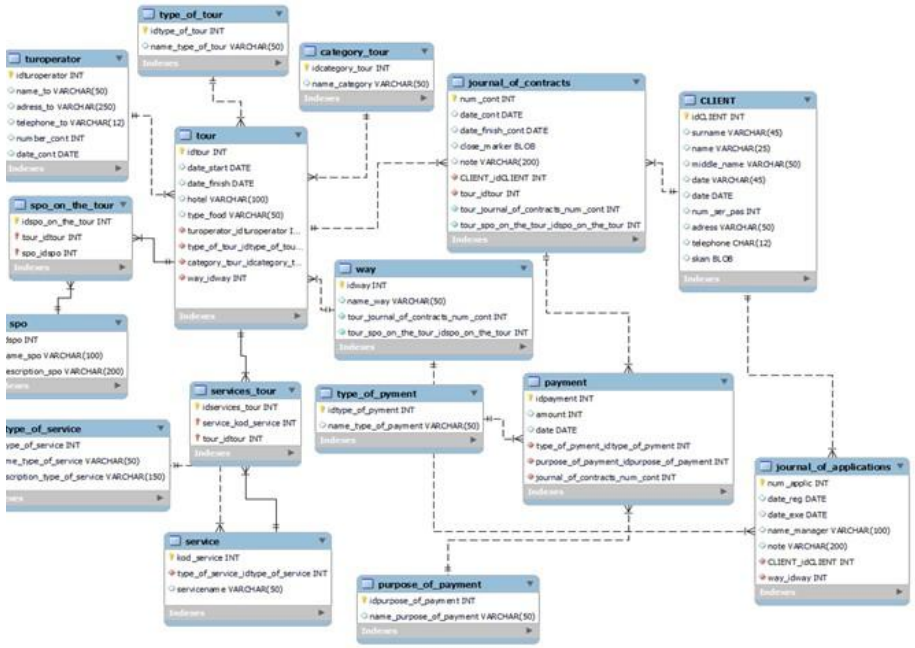


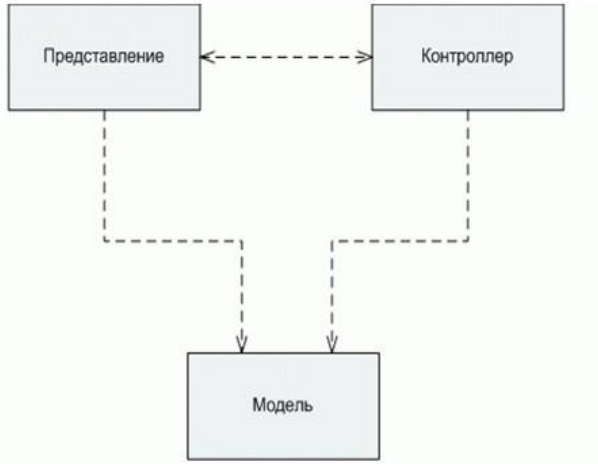
Диаграмма развертывания



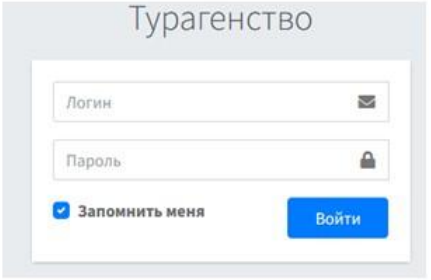
Физическая модель базы данн



MVC-паттерн



Экранные формы приложения



Форма авториз
системе

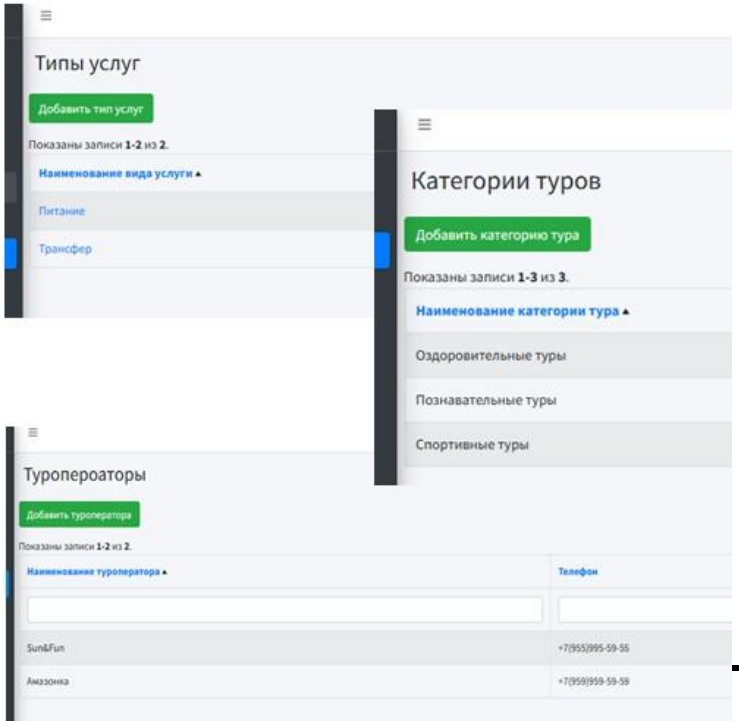
ная форма программы

ы

Туры
Добавить тур

Главная / Журнал договоров
Дата заключения договора
Дата закрытия договора

Экранные формы приложения



ение

ыл выполнен анализ предметной области. Раз
и физическое представление базы данных си
UML - диаграммы системы. Разработана и от
ированная информационная система центра
кого сервиса.