МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.О. Пятлина |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ СРЕДСТВАМИ ЯЗЫКА UML |
| по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4631 |  |  |  | К.С.Ларионов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Содержание**

|  |  |
| --- | --- |
| Введение……………………………………………...................................... | 3 |
| 1. Описание основных функций Информационной системы «Онлайн-бронирование туров»………………………………………………………. | 3 |
| 2. Краткая информация об аппарате объектно-ориентированного проектирования программного обеспечения информационной систем... | 3 |
| 3.CASE средства, используемые для проектирования ПО ИС, их возможности, достоинства, особенности применения…………………... | 4 |
| 4. Разработка ПО информационной системы «Онлайн-бронирование туров»........................................................................................................... | 5 |
| 4.1. Use-case диаграмма…………………………………………………...… | 5 |
| 4.2. Диаграмма классов……………..………………………………………. | 7 |
| 4.3. Диаграммы последовательностей…………..…………………………. | 10 |
| 4.4. Диаграммы состояний……………………………………...…………….. | 14 |
| 4.5 Диаграммы видов деятельности……………………………………………………… | 18 |
| 4.6.Диаграмма размещений…………………………….. | 23 |
| Заключение…………………………………………………… | 24 |
| Приложение…………………………………………………………... | 25 |

**Введение.**

В данной работе приведен проект структуры программного обеспечения информационной системы. В качестве предметной области рассмотрена «Онлайн-бронирование туров». Проектирование производилось с помощью специализированного программного обеспечения Software Ideas Modeler – CASE пакета для проектирования диаграмм UML. Приложение полностью совместимо с версией UML 2.2 и включает всевозможные типы диаграмм языка UML 2.2, используемых в этой системе. В приложении к пояснительной записке приведены результаты автоматической генерации кода программ.

**1. Описание основных функций Информационной системы «Онлайн-бронирование туров».**

Информационная система «Онлайн-бронирование туров» применяется для упрощения работы турагентства и более удобного бронирования туров клиентом. Система позволяет клиенту контактировать с турагентством через сайт турагентства, на котором он может выбрать нужный тур, забронировать его и оплатить любым удобным способом.

* Проверка наличия путёвок
* Работа администратора с базой данных
  + Добавление новых туров в базу данных
  + Обновление и редактирование информации о доступных турах
* Регистрация нового пользователя
  + Регистрация по электронной почте
  + Регистрация по номеру телефона
  + Подписаться на новостную рассылку при регистрации
* Поиск тура
  + Выбор страны
  + Выбор города
  + Выбор отеля
* Заказ путёвки
  + Заказ через Интернет
  + Заказ в отделении турагентства
  + Заказ по телефону
  + Подключение СМС-оповещений о состоянии заказа
* Покупка путёвки
  + Выбор способа оплаты покупки
  + Возможность использования скидки
* Выбор способа оплаты заказа
  + Наличными
  + Банковской картой
* Составление отчетности турфирмы
  + Получение отчётов о проданных турах
  + Получение отчётов о планируемых турах

**2. Краткая информация об аппарате объектно-ориентированного проектирования программного обеспечения информационной систем.**

Важное место в моделировании информационных систем занимает методология и системы, использующие UML — унифицированный язык моделирования (Unified Modeling Language).  
  
UML — язык для спецификации, визуализации, конструирования и документирования сложных информационно-насыщенных объектных систем. В настоящее время зарегистрирован как международный стандарт ISO/IEC 19501:2005 «Information technology - Open Distributed Processing - Unified Modeling Language (UML)».

Словарь языка UML включает три вида строительных блоков: сущности, отношения, диаграммы.

Сущности в UML — это абстракции, являющиеся основными элементами модели. Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

В UML имеется четыре типа сущностей: структурные, поведенческие, группирующие, аннотационные.

Структурные сущности — это имена существительные в моделях на языке UML. Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует семь разновидностей структурных сущностей: Класс, Интерфейс, Кооперация, Прецедент, Активный класс, Компонент, Узел.  
Поведенческие сущности являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей: Взаимодействие и Автомат.  
Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель.   
Аннотационные сущности — пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов — примечание.  
Все разновидности сущностей UML в диаграммах имеют свой способ графического изображения.

В языке UML определены четыре типа отношений: зависимость, ассоциация, обобщение, реализация.

Диаграмма в UML — это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций, соответствующих пяти наиболее употребительным видам, которые составляют архитектуру ИС.

**3.CASE средства, используемые для проектирования ПО ИС, их возможности, достоинства, особенности применения.**

CASE средства, поддерживающие методологию UML, — Rational Rose (Rational Software), Paradigm Plus (CA/Platinum), ARIS (IDS Sheer AG), Together Designer (Borland) и др.

Система Rational Rose позволяет строить восемь типов диаграмм UML: диаграммы прецедентов, диаграммы классов, диаграммы последовательностей, диаграммы кооперации, диаграммы состояний, диаграммы действий, компонентные диаграммы, диаграммы развертывания. Основным типом диаграмм, своеобразным ядром моделирования в UML являются диаграммы классов. Кроме UML, предусмотрено использование и других методов (Booch, OMT).

Система Paradigm Plus ориентирована на методологию OOCL (Object Oriented Change and Learning) и компонентную технологию проектирования и разработки. Она поддерживает диаграммы различных методов (UML, CLIPP, TeamFusion, OMT, Booch, OOCL, Martin/Odell, Shlaer/Mellor, Coad/Yourdon).

Система ARIS обеспечивает четыре различных «взгляда» на моделирование и анализ: Процессы, Функции (с Целями), Данные, Организация. Для каждого «взгляда» поддерживаются три уровня анализа (требования, спецификации, внедрение). Каждый из уровней анализа состоит из своего комплекта моделей различных типов, в том числе диаграмм UML, диаграмм SAP/R3 и др. Каждый объект моделей ARIS имеет множество атрибутов, которые позволяют контролировать процесс разработки моделей, определять условия для выполнения функционально-стоимостного анализа, имитационного моделирования, взаимодействия с workflow-системами и т. д.

На этапе создания клиентского и серверного кода все современные средства UML-моделирования могут осуществлять генерацию кода на различных языках программирования.

При проектирование информационной системы была использована система Ideas Software Modeler позволяет строить все типы диаграмм UML: диаграммы прецедентов, диаграммы классов, диаграммы последовательностей, диаграммы кооперации, диаграммы состояний, диаграммы действий, компонентные диаграммы, диаграммы развертывания. Выбранное CASE средство отлично подходит для решения поставленной задачи.

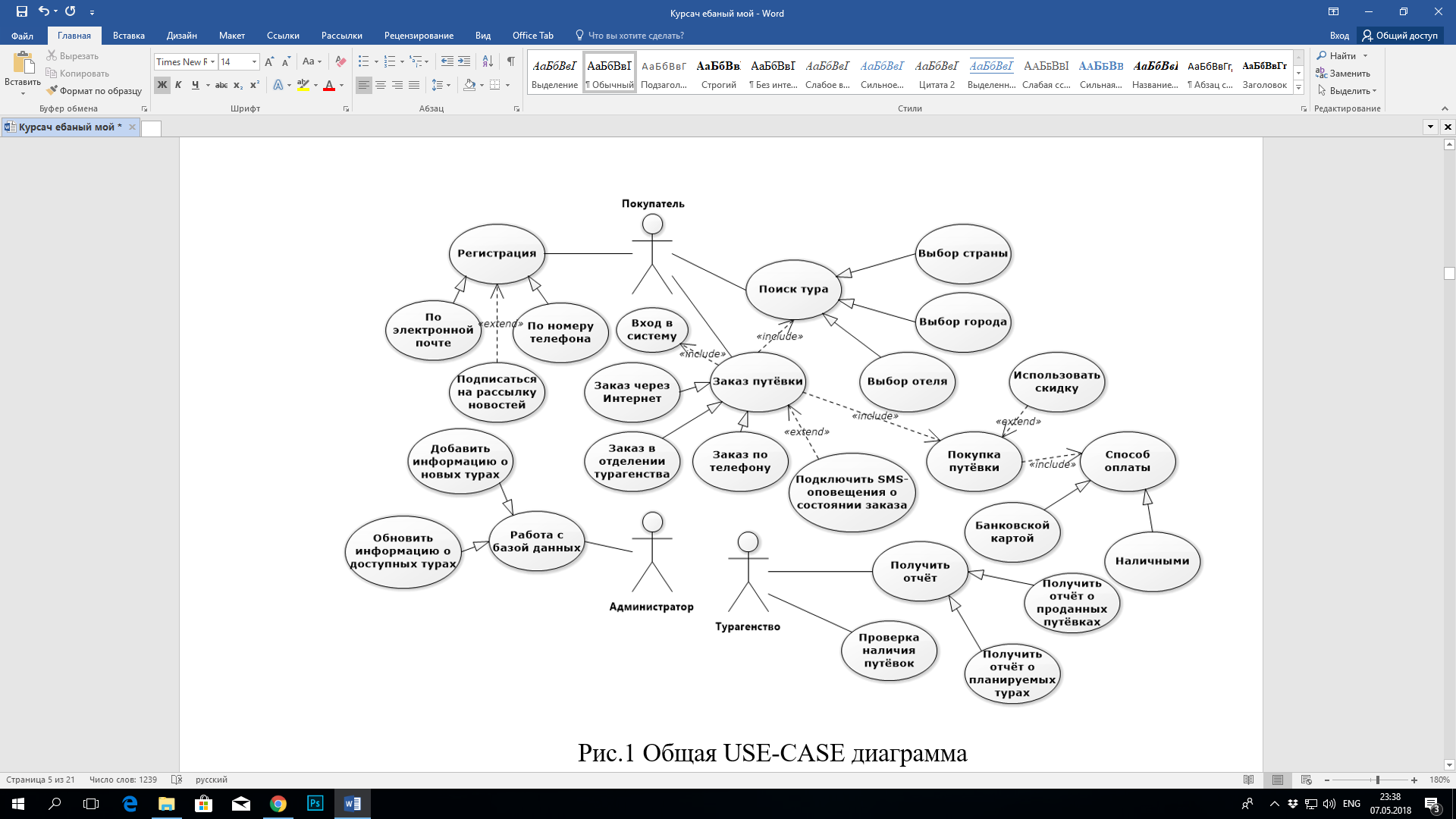
**4. Разработка ПО информационной системы «Онлайн-бронирование туров».**

**4.1. Use-case диаграмма (диаграмма вариантов использования, сценариев, прецедентов).** Диаграммы позволяют наглядно представить ожидаемое поведение системы. Элементы, используемые на диаграмме:

* Сценарий. (Определяет один фрагмент поведения системы, без раскрытия внутреннего содержания)

Актер. (Внешняя по отношению к главной системе сущность, которая участвует в сценариях, является инициатором, источником или приемником информации, внутреннее содержание не рассматривается)  
В  созданной диаграмме использовались все возможные виды связей между  элементами  диаграммы:

* коммуникация (communication),
* включение (include),
* расширение (extend),
* обобщение (generalization).

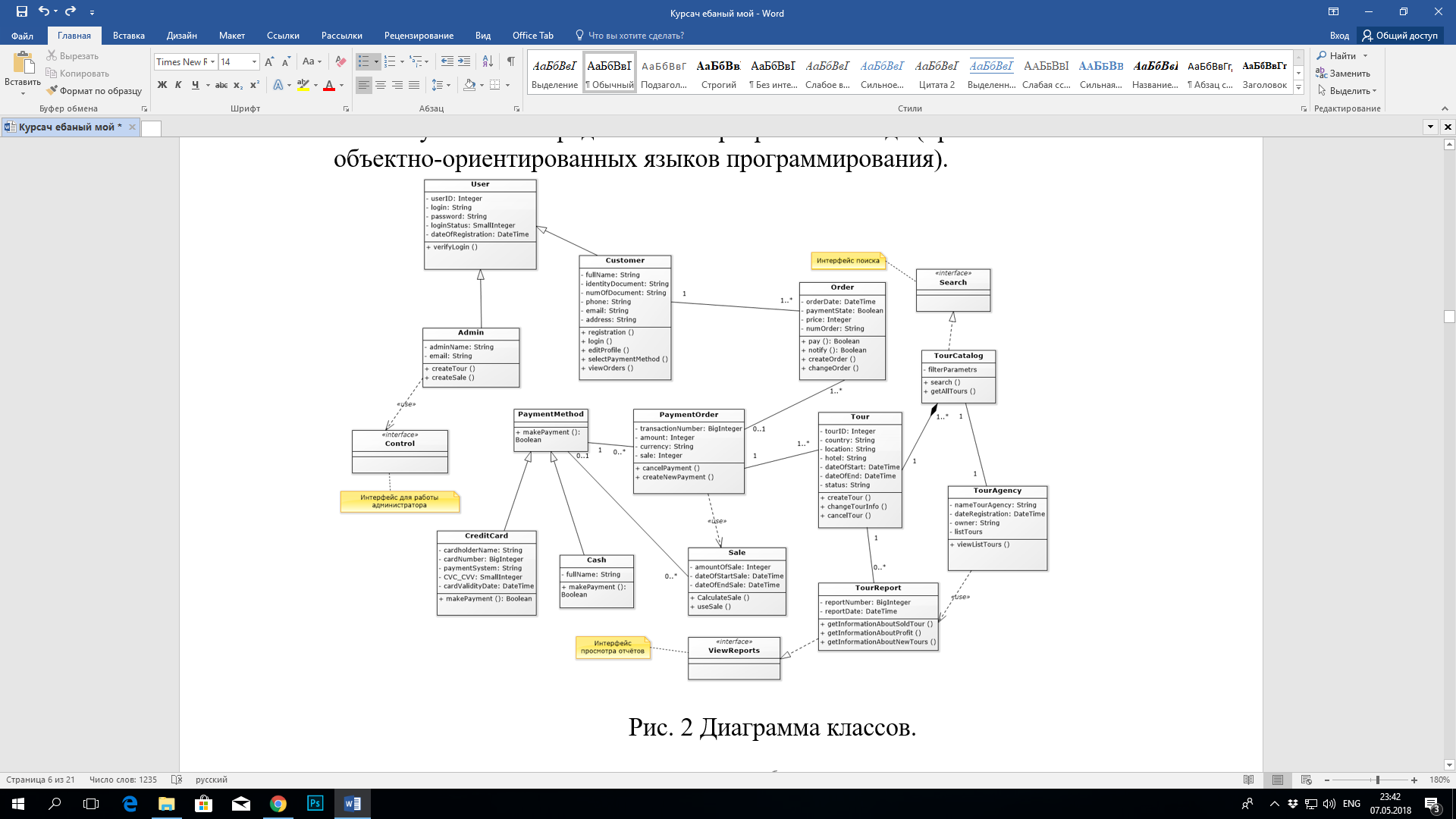
Диаграммы Use Case по концепции напоминают классические DFD диаграммы, также применяемые для структурного анализа. Use Case также отображают границы исследуемой системы, её функциональность и определяют сущности и процессы, а также пользователей системы. 

**Расчет оценки диаграммы.   
**

где *Sobj*-оценка элемента на диаграмме, *Slink*- оценка связей, *Оbj*- кол-во объектов на диаграмме, *Tobj*–количество типов объектов, *Tlink*- количество типов связи.

**4.2. Диаграмма классов.**

[**Диаграмма классов**](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2) (Class diagram) — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами. Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

* концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
* точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
* точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).
* 

**Расчет оценки диаграммы.**

где *Sobj*-оценка элемента на диаграмме, *Slink* - оценка связей, *Оbj* - кол-во объектов на диаграмме, *Tobj*–количество типов объектов, *Tlink*- количество типов связи.

где *Op*- количество операций, *Atr*- количество атрибутов.

**Расчет оценки диаграммы классов с атрибутами и операциями:**

**4.3. Диаграммы последовательностей.**

**Диаграммы последовательностей *(***[*англ.*](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*Sequence diagram)* — [диаграмма](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основными элементами диаграммы последовательностей являются обозначения объектов (прямоугольники), вертикальные линии ([*англ.*](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *lifeline*), отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

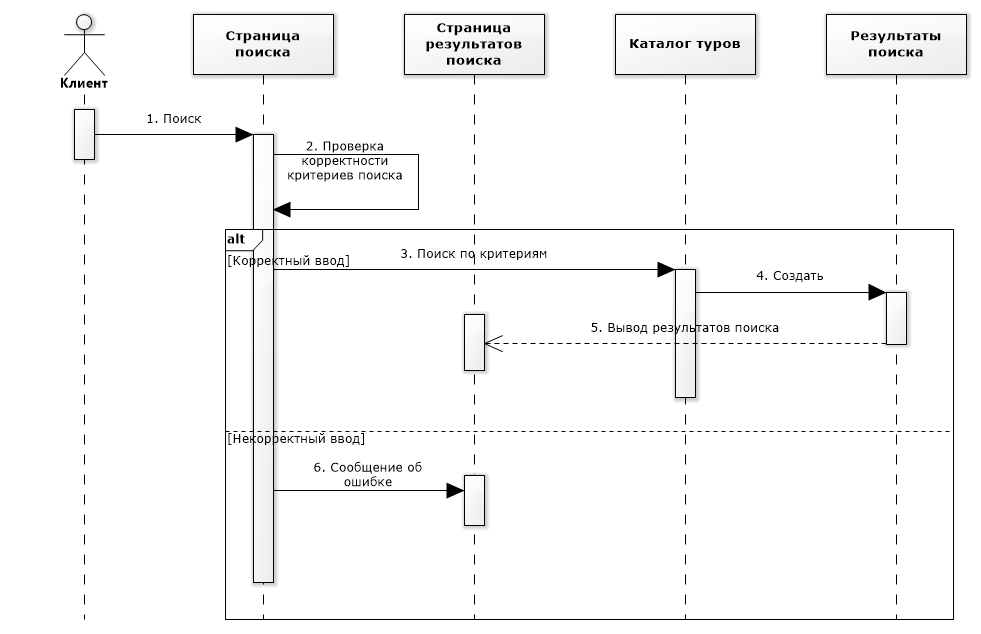


Рис.3 Диаграмма последовательностей для сценария «Поиск тура».

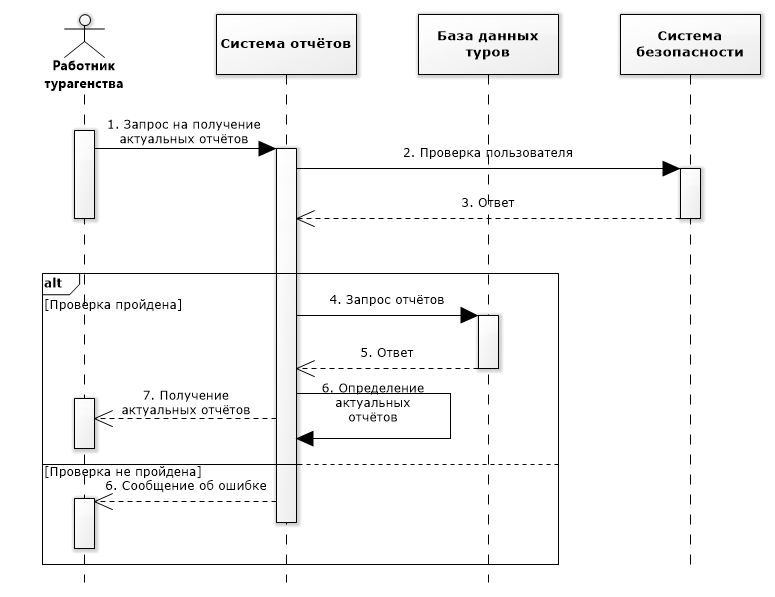
****

Рис.4 Диаграмма последовательностей для сценария «Работа с отчётами турфирмы».

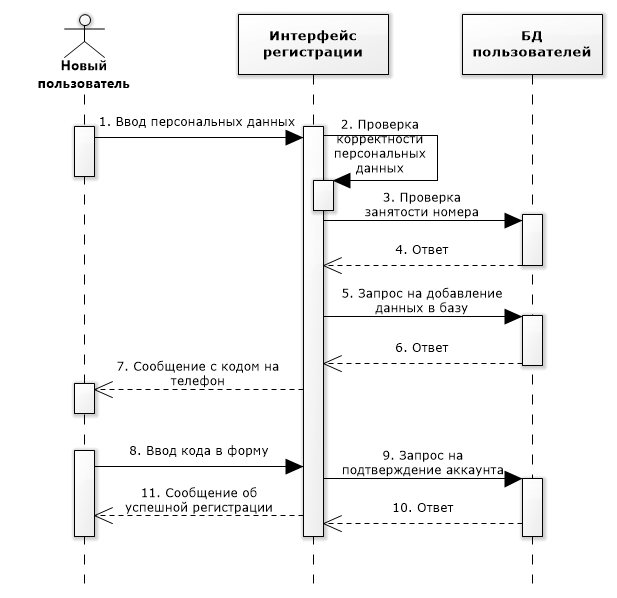
****

Рис.5 Диаграмма последовательностей для сценария «Регистрация пользователя по e-mail».

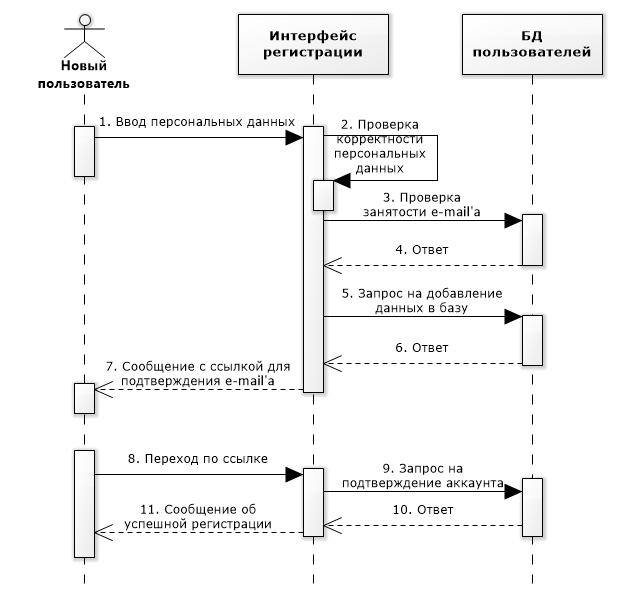
****

Рис.6 Диаграмма последовательностей для сценария «Регистрация пользователя по номеру телефона».

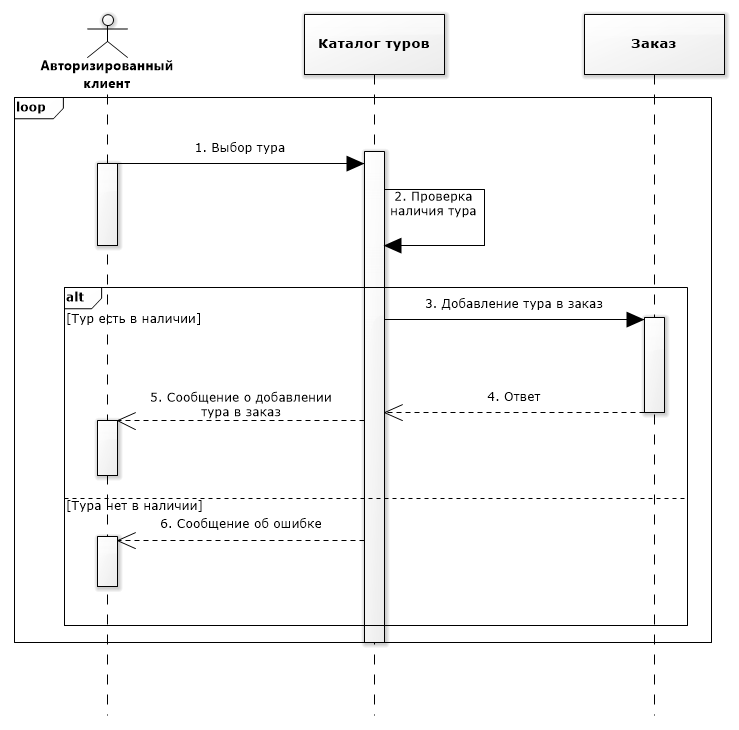
****

Рис.7 Диаграмма последовательностей для сценария «Заказ тура».

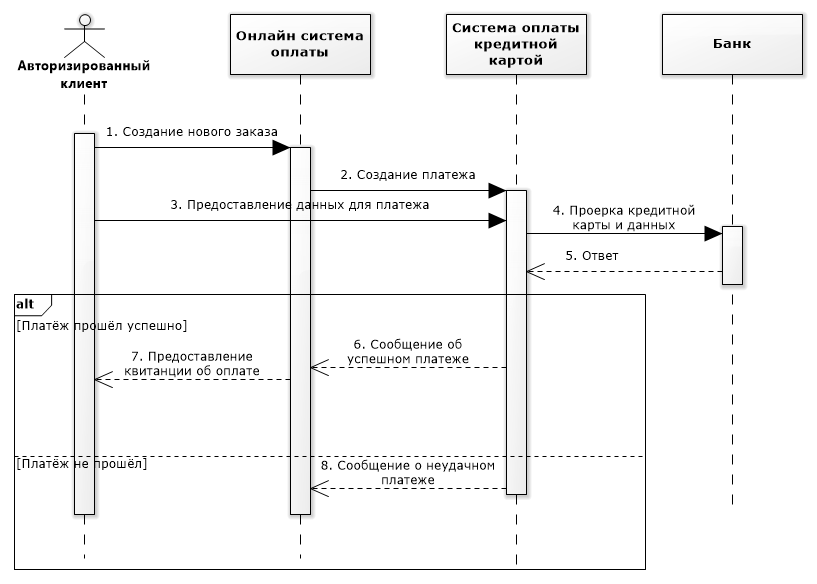
****

Рис.8 Диаграмма последовательностей для сценария «Оплата заказа».

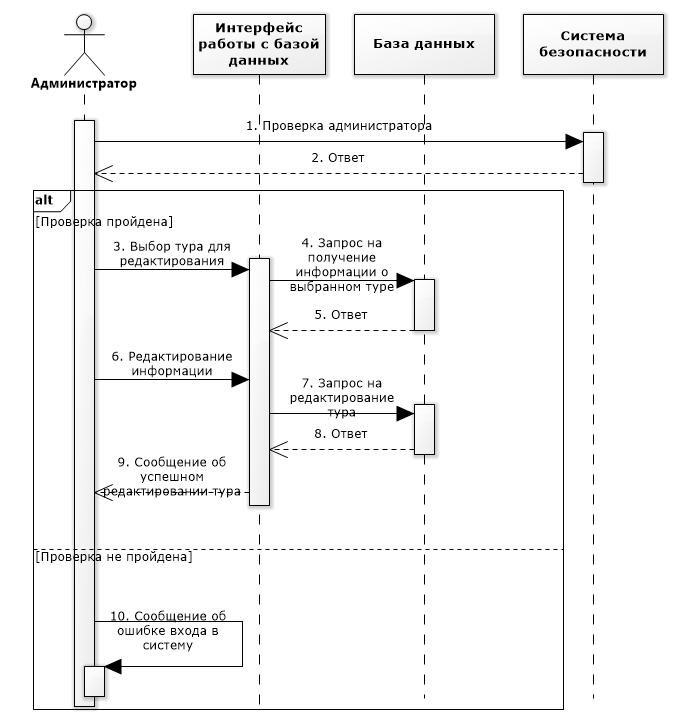
****

Рис.9 Диаграмма последовательностей для сценария «Работа с базой данных».

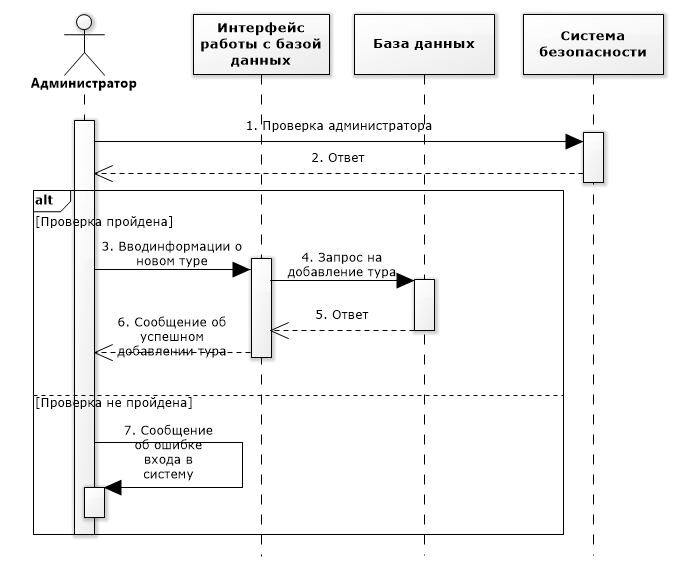
****

Рис.10 Диаграмма последовательностей для альтернативного сценария «Работа с базой данных».

**4.4. Диаграммы состояний (Statechar diagram)**

Диаграмма состояний (Statechar diagram) определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На диаграмме имеются два специальных состояния – начальное (start) и конечное(stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).



Рис 11. Диаграмма состояний для класса «User».

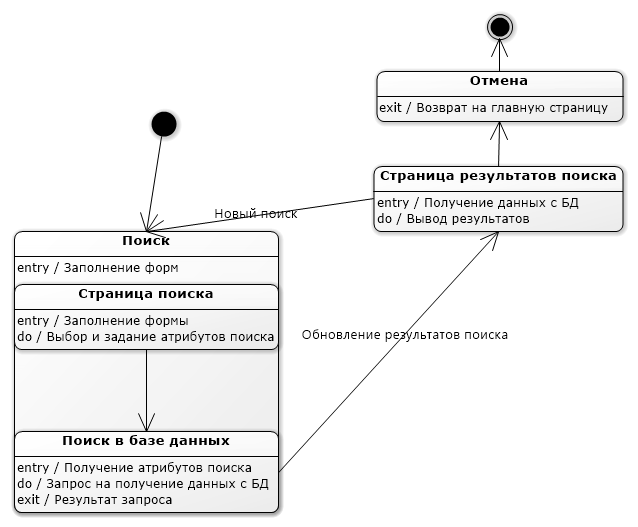


Рис 12. Диаграмма состояний для класса «Search».

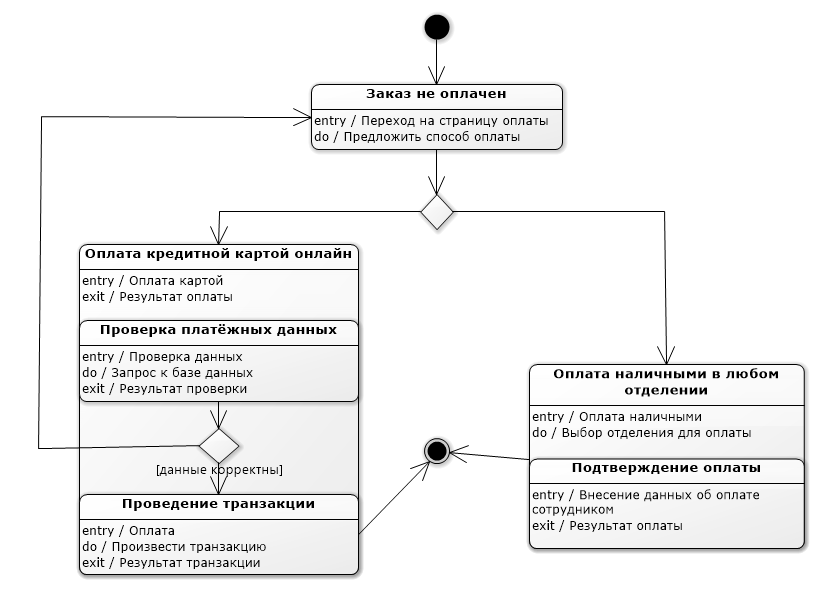


Рис 13. Диаграмма состояний для класса «PaymentOrder».

Рис 14. Диаграмма состояний для класса «CreditCard».

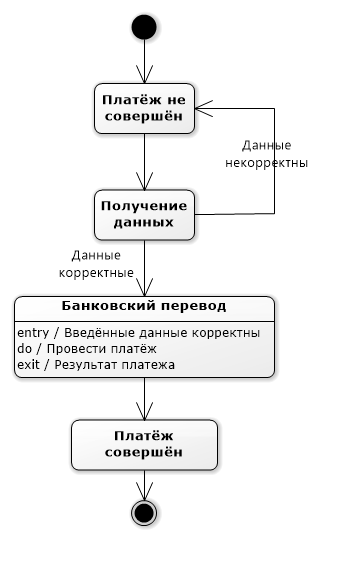


Рис 15. Диаграмма состояний для класса «Cash».

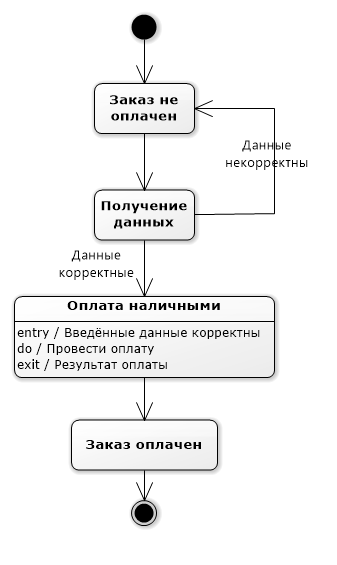


Рис 16. Диаграмма состояний для класса «Sale».



Рис 17. Диаграмма состояний для класса «Tour».

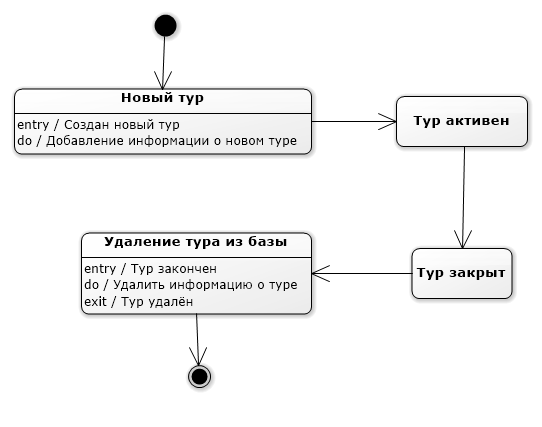
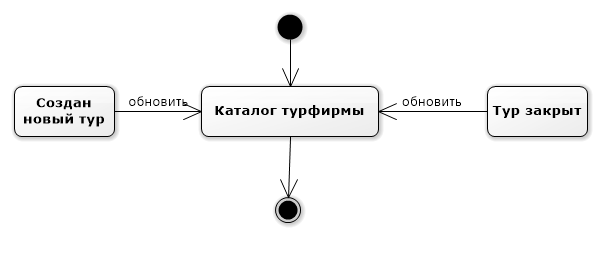
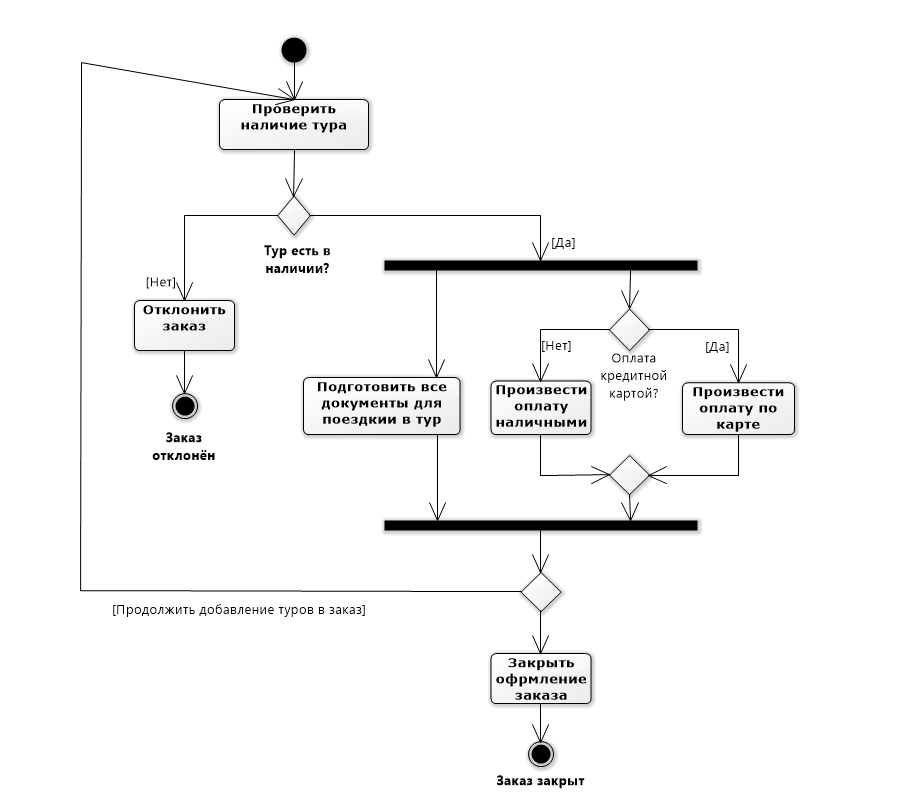


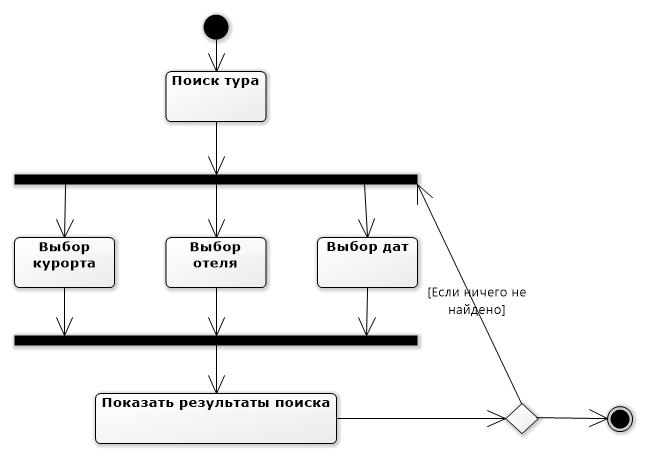
Рис 18. Диаграмма состояний для класса «TourCatalog».

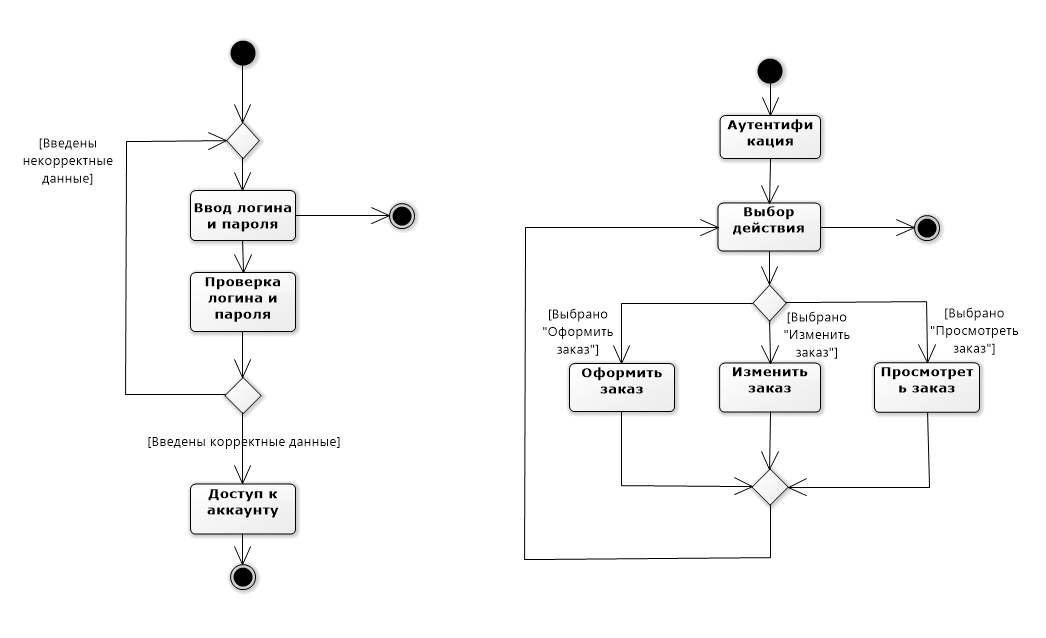


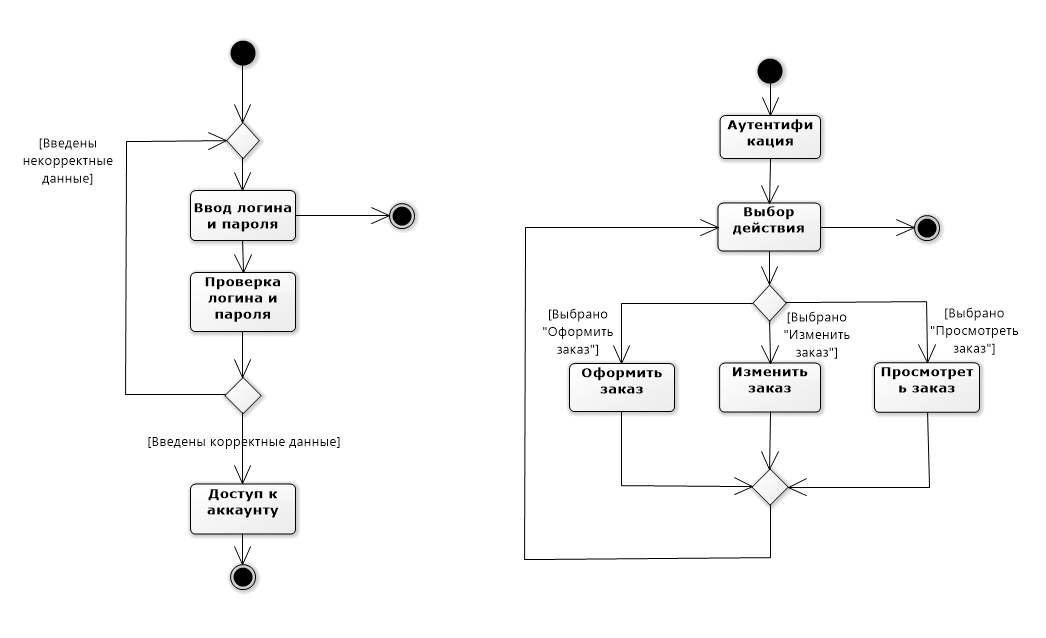
**4.5 Диаграммы видов деятельности(Activity diagram)**

Диаграммы видов деятельности(Activity diagram)-диаграмма, на которой показано разложение некоторой деятельности на ее составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов - вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого.

  
Рис 19. Диаграмма деятельности для метода «Оформление заказа» класса «Order».

  
Рис 20. Диаграмма деятельности для метода «Поиск» класса «TourCatalog».

  
Рис 21. Диаграмма деятельности для метода «Вход в систему» класса «Customer».

  
Рис 22. Диаграмма деятельности для метода «Действия с заказами» класса «Order».

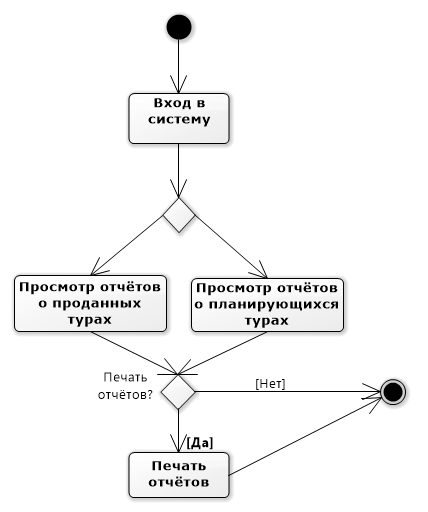
  
Рис 23. Диаграмма деятельности для метода «Получение отчётов» класса «TourReport».

Рис 24. Диаграмма деятельности для метода «Произведение платежа» класса «PaymentMethod».

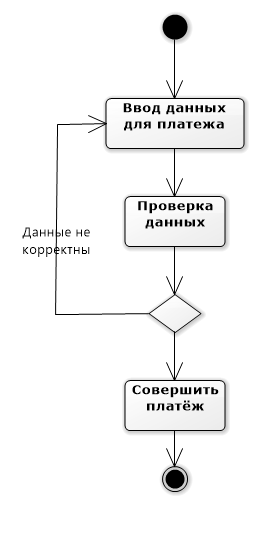


Рис 25. Диаграмма деятельности для метода «Просмотр списка туров» класса «TourAgency».



Рис 26. Диаграмма деятельности для метода «Расчёт скидки» класса «Sale».

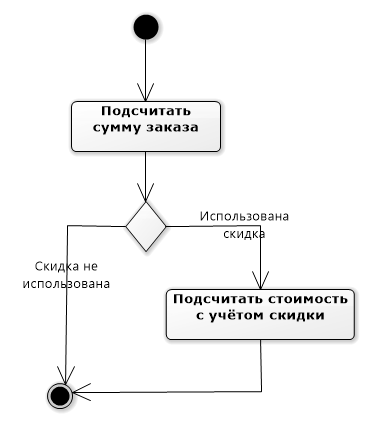
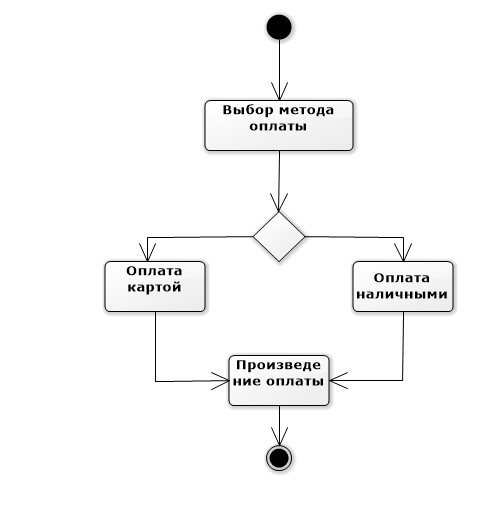


Рис 27. Диаграмма деятельности для метода «Выбор способа оплаты» класса «Customer».



**4.6.Диаграмма размещений (Диаграмма развертывания).**

Диаграмма развертывания (Deployment diagram)- предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. Диаграмма развертывания отражает физические взаимосвязи между программными и аппаратными компонентами разрабатываемой системы. Каждый узел на диаграмме развертывания представляет собой некоторый тип вычислительного устройства - в большинстве случаев самостоятельную часть аппаратуры.

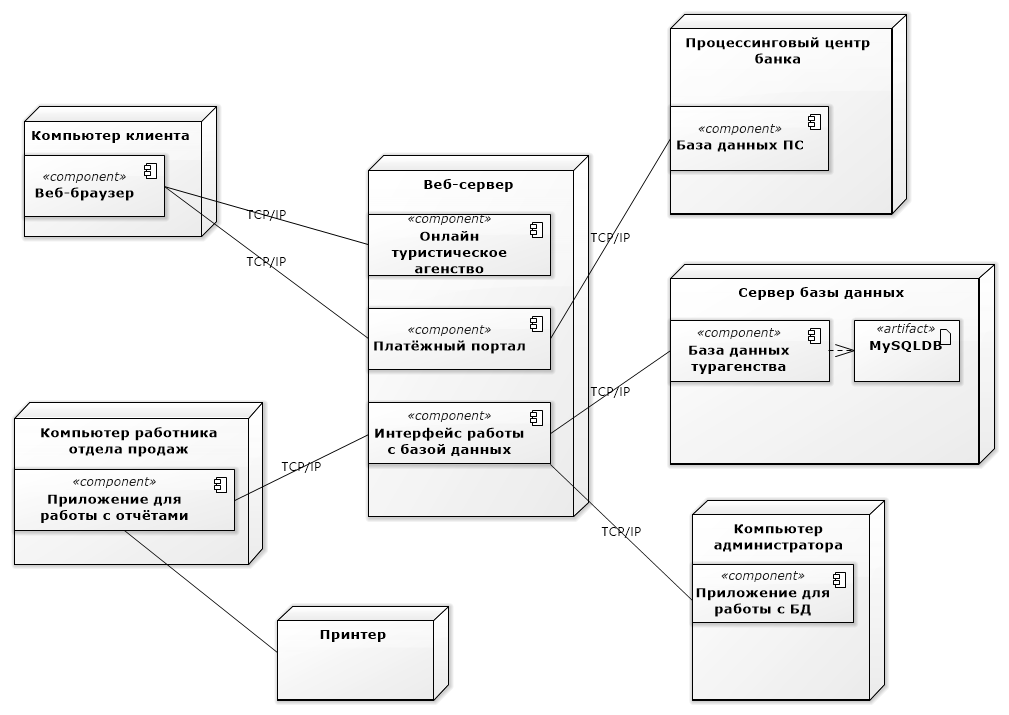


Рис 27. Диаграмма размещения

**ЗАКЛЮЧЕНИЕ**

В данной курсовой работе был приведен проект структуры программного обеспечения информационной системы «Онлайн-бронирование туров» в нотации UML с использованием СASE-средства для проектирования диаграмм UML 2.2 Ideas Software Modeler.

В пояснительной записке приведено 28 диаграмм с оценками их информационной наполненности.

Система описана практически со всех возможных точек зрения, рассмотрены разные аспекты поведения системы, методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках.

На основе спроектированной диаграммы классов автоматически сгенерирован код программы.

Данная работа дает возможность организовать качественное функционирование описанной информационной системы, позволяет автоматизировать процессы, что сэкономит и время, и средства на её разработку.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Рамбо, Якобсон, Буч: Введение в UML от создателей языка – ДМК-Пресс, 2015. – 496c.
2. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования – Диалектика, 2016. –736c.
3. Лешек Мацяшек: Анализ и проектирование информационных систем с помощью UML 2.0 – Вильямс, 2016 – 816c.
4. Хассан Гома: UML Проектирование систем реального времени, распределенных и параллельных приложений – ДМК-Пресс, 2016. – 700c
5. Пайлон, Питмен: UML 2 для программистов – Питер, 2012 – 240c.

**Приложение**

|  |
| --- |
| class Order  {  private:  DateTime orderDate;  bool paymentState;  int price;  string numOrder;  string phone;  long int transactionNumber;  public:  bool pay ();  bool notify ();  void createOrder ();  void changeOrder ();  };  bool Order::pay ()  {    }  bool Order::notify ()  {    }  void Order::createOrder ()  {    }  void Order::changeOrder ()  {    }  class Customer : public User  {  private:  string fullName;  string identityDocument;  string numOfDocument;  string phone;  string email;  string address;  bool paymentState;  public:  void registration ();  void login ();  void editProfile ();  void selectPaymentMethod ();  void viewOrders ();  };  void Customer::registration ()  {    }  void Customer::login ()  {    }  void Customer::editProfile ()  {    }  void Customer::selectPaymentMethod ()  {    }  void Customer::viewOrders ()  {    }  class PaymentOrder  {  private:  long int transactionNumber;  int amount;  string currency;  int sale;  string location;  public:  void cancelPayment ();  void createNewPayment ();  };  void PaymentOrder::cancelPayment ()  {    }  void PaymentOrder::createNewPayment ()  {    }  class PaymentMethod  {  private:  DateTime dateOfStartSale;  public:  bool makePayment ();  };  bool PaymentMethod::makePayment ()  {    }  class CreditCard : public PaymentMethod  {  private:  string cardholderName;  long int cardNumber;  string paymentSystem;  short int CVC\_CVV;  DateTime cardValidityDate;  public:  bool makePayment ();  };  bool CreditCard::makePayment ()  {    }  class Cash : public PaymentMethod  {  private:  string fullName;  public:  bool makePayment ();  };  bool Cash::makePayment ()  {    }  class Tour  {  private:  int tourID;  string country;  string location;  string hotel;  DateTime dateOfStart;  DateTime dateOfEnd;  string status;  public:  void createTour ();  void changeTourInfo ();  void cancelTour ();  };  void Tour::createTour ()  {    }  void Tour::changeTourInfo ()  {    }  void Tour::cancelTour ()  {    }  class TourReport  {  private:  long int reportNumber;  DateTime reportDate;  public:  void getInformationAboutSoldTour ();  void getInformationAboutProfit ();  void getInformationAboutNewTours ();  };  void TourReport::getInformationAboutSoldTour ()  {    }  void TourReport::getInformationAboutProfit ()  {    }  void TourReport::getInformationAboutNewTours ()  {    }  class TourAgency  {  private:  string nameTourAgency;  DateTime dateRegistration;  string owner;  object listTours;  public:  void viewListTours ();  };  void TourAgency::viewListTours ()  {    }  class Admin : public User  {  private:  string adminName;  string email;  public:  void createTour ();  void createSale ();  };  void Admin::createTour ()  {    }  void Admin::createSale ()  {    }  class Search  {  };  class TourCatalog  {  private:  object filterParametrs;  public:  void search ();  void getAllTours ();  };  void TourCatalog::search ()  {    }  void TourCatalog::getAllTours ()  {    }  class User  {  private:  int userID;  string login;  string password;  short int loginStatus;  DateTime dateOfRegistration;  public:  void verifyLogin ();  };  void User::verifyLogin ()  {    }  class Sale  {  private:  int amountOfSale;  DateTime dateOfStartSale;  DateTime dateOfEndSale;  public:  void CalculateSale ();  void useSale ();  };  void Sale::CalculateSale ()  {    }  void Sale::useSale ()  {    }  class Control  {  };  class ViewReports  {  }; |