Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Проектирование программного обеспечения»

Тема: Интерфейс бронирования номеров

**Цель:** изучить методологии объектно-ориентированного моделирования средствами UML. Ознакомление с основными принципами объектно-ориентированного проектирования программного обеспечения, получение навыков проектирования структуры информационной системы с применением UML.

Студент: Жук С.С.

ФИТ 3 курс 2 группа

Преподаватель: Курилец А.В.

# **Описание функциональных требований**

Функциональны требования к системе можно разделить на требования к функционалу для различных ролей приложения – пользователя, гостя и администратора.

Функционал для пользователя:

* регистрация;
* авторизация;
* поиск и фильтрация по критериям;
* бронирование номера в отеле;
* добавление отзыва и рейтинга на отель;
* просмотр всей истории бронирования;
* просмотр и обновление личных данных;
* отмена бронирования номера.
* просмотр списка отелей и номеров.

Функционал для администратора:

* авторизация;
* поиск и фильтрация по критериям;
* отправка уведомлений о бронировании;
* управление пользователями (добавление, удаление, редактирование, блокировка пользователей);
* управление объявлениями;
* проверка объявлений и отзывов.
* просмотр списка отелей и номеров.

Функционал для гостя:

* регистрация;
* авторизация;
* просмотр списка отелей и номеров.
* поиск и фильтрация по критериям;

# **2. Описание программных средств**

Для построения диаграммы вариантов использования применен веб-ресурс Draw.io, разрабатываемый компанией JGraph Ltd и направленный на построение диаграмм. Адрес веб-ресурса – <https://www.drawio.com>. Данный ресурс доступен на всех платформах, имеющих веб-браузер и доступ в Интернет.

В функционал веб-ресурса Draw.io входит построение графиков, смысловых карт, UML-диаграмм, диаграммы Венна, Agile и Kanban досок и многое другое. Он также поддерживает импорт и экспорт файлов в различных форматах, включая PNG, JPEG, PDF и SVG.

# **3. Описание практического задания**

В ходе выполнения практического задания необходимо построить диаграмму классов. Данная диаграмма описывает типы объектов системы и различного рода статические отношения, которые существуют между ними.

В каждом прямоугольнике, отражающем класс, содержатся имя класса, его атрибуты (поля и свойства) и операции (методы).

Связи разделяются на зависимость (если изменения одного вызовут изменения другого), ассоциацию (в случае, когда один класс пользуется функционалом другого), обобщения (наследование всех открытых полей, свойств и методов), агрегации (включение в другой класс в виде составной части, при которой зависимый класс не может существовать без основного) и композиции (также включение в виде составной части другого класса, но составной класс не может существовать обособленно от основного).

Диаграмма классов для интерфейса бронирования номеров «HotelBooking» представлена на рисунке 1.1.

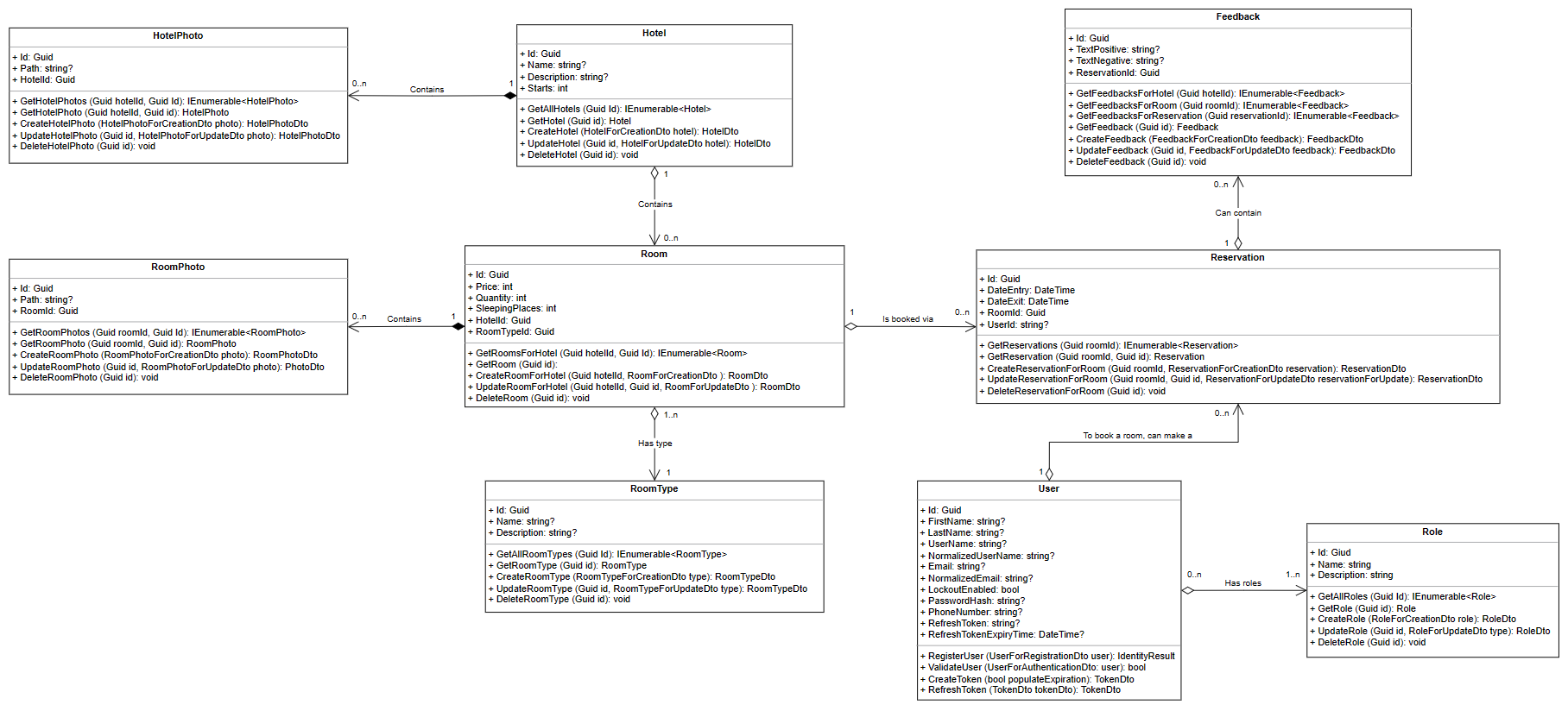


Рисунок 1.1 – Диаграмма классов

Рассмотрим подробнее составные части данной диаграммы. В ней содержится 9 классов со следующими атрибутами и операциями:

* Класс: User.  
  Атрибуты: Id, FirstName, LastName, UserName, NormalizedUserName, Email, NormalizedEmail, LockoutEnabled, PasswordHash, PhoneNumber, RefreshToken, RefreshTokenExpiryTime.  
  Операции: RegisterUser, ValidateUser, CreateToken, RefreshToken.
* Класс: Role.  
  Атрибуты: Id, Name, Description.  
  Операции: GetAllRoles, GetRole, CreateRole, UpdateRole, DeleteRole.
* Класс: RoomType.  
  Атрибуты: Id, Name, Description.  
  Операции: GetAllRoomTypes, GetRoomType, CreateRoomType, UpdateRoomType, DeleteRoomType.
* Класс: Room.  
  Атрибуты: Id, Price, Quantity, SleepingPlaces, HotelId, RoomTypeId.  
  Операции: GetRoomsForHotel, GetRoom, CreateRoomForHotel, UpdateRoomForHotel, DeleteRoom.
* Класс: Hotel.  
  Атрибуты: Id, Name, Description, Stars.  
  Операции: GetAllHotels, GetHotel, CreateHotel, UpdateHotel, DeleteHotel.
* Класс: RoomPhoto.  
  Атрибуты: Id, Path, RoomId.  
  Операции: GetRoomPhotosForRoom, GetRoomPhoto, CreateRoomPhotoForRoom, UpdateRoomPhotoForRoom, DeleteRoomPhoto.
* Класс: HotelPhoto.  
  Атрибуты: Id, Path, HotelId.  
  Операции: GetHotelPhotosForHotel, GetHotelPhoto, CreateHotelPhotoForHotel, UpdateHotelPhotoForHotel, DeleteHotelPhoto.
* Класс: Reservation.  
  Атрибуты: Id, DateEntry, DateExit, RoomId, UserId.  
  Операции: GetReservations, GetReservation, CreateReservationForRoom, UpdateReservationForRoom, DeleteReservationForRoom.
* Класс: Feedback.  
  Атрибуты: Id, TextPositive, TextNegative, ReservationId.  
  Операции: GetFeedbacksForHotel, GetFeedbacksForRoom, GetFeedback, CreateFeedback, UpdateFeedback, DeleteFeedback.

Данные сущности имеют следующие связи:

* Пользователи User связаны с ролями Role агрегационной связью один-ко-многим, причем у каждого пользователя может быть 1 или несколько ролей, но каждая роль принадлежит 0 или нескольким пользователям.
* Пользователи User бронируют номера через класс Reservation, связанный агрегацией. У одного пользователя может быть 0 или несколько резерваций – экземпляров класса Reservation.
* Комнаты Room связаны с типами комнат RoomType агрегационной связью один-ко-многим, то есть у каждой из комнат есть один из многи типов.
* Отели Hotel связаны с комнатами Room агрегацией один-ко-многим, то есть в одном отеле может быть несколько комнат, но каждая комната может принадлежать только одному отелю.
* Отзывы Feedback связаны с бронями Reservation, то есть если пользователь пользовался услугами отеля посредством резервации, то он может оставить отзыв, причем на каждую бронь может не быть ни одного или быть несколько отзывов.
* Фото комнат RoomPhoto связаны с комнатами Room композицией, так как фотографии комнат могут существовать только в контексте существования самих комнат, и у каждой комнаты может быть 0 или несколько фотографий, а каждая фотография относится только к одной из комнат.
* Фото отелей HotelPhoto связаны с отелями Hotel также композицией. У каждого отеля может быть 0 или несколько фотографий, а каждая фотография относится только к одному конкретному отелю.

Если сгруппировать всё вышеперечисленное в пакеты, то получится следующее – рисунок 2.

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 2 – Диаграмма пакетов

**Теоретические вопросы:**

**1) Перечислите структурные диаграммы, которые входят в UML 2.0.**

В UML 2.0 структурные диаграммы включают следующие типы:

1. Диаграмма классов (Class Diagram): Представляет классы, их атрибуты, методы и отношения между классами.
2. Диаграмма объектов (Object Diagram): Показывает экземпляры классов (объекты) и их связи в конкретный момент времени.
3. Диаграмма компонентов (Component Diagram): Определяет структуру системы в виде компонентов и их взаимодействий.
4. Диаграмма развертывания (Deployment Diagram): Отображает физическую архитектуру системы, включая узлы и артефакты.
5. Диаграмма пакетов (Package Diagram): Группирует связанные элементы модели в пакеты, показывая их зависимости.
6. Диаграмма структур (Composite Structure Diagram): Показывает внутреннююструктуру классов и их взаимодействие.

**2) Укажите назначение структурных диаграммы.**

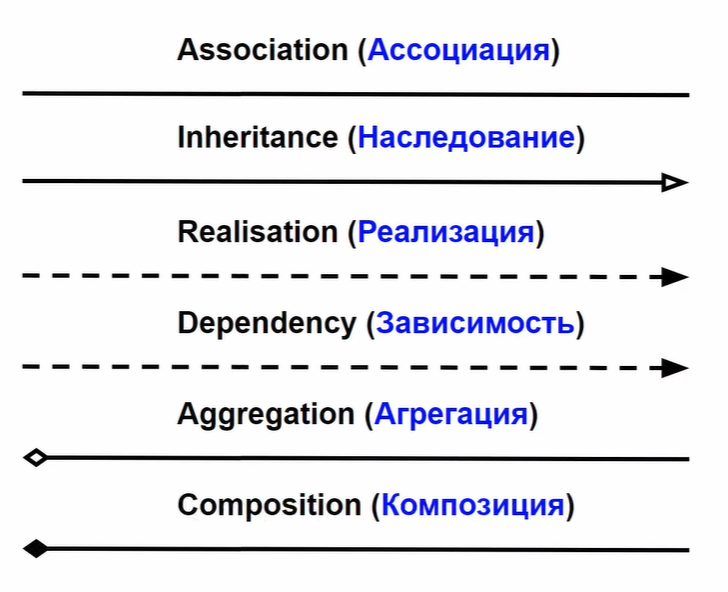
Структурные диаграммы служат для визуализации и описания статической структуры системы. Их назначение включает:

* Представление классов, объектов, компонентов и их отношений.
* Упрощение понимания архитектуры системы и её модульности.
* Помощь в проектировании и анализе системы, выявлении зависимостей и взаимодействий.

**3) Опишите нотации, которые используются для построения Classes диаграмм.**

Основные нотации для диаграмм классов:

* Класс: Представляется в виде прямоугольника, разделенного на три части: имя класса (вверху), атрибуты (в середине) и методы (внизу).
* Атрибуты: Записываются в формате visibility name: type, где visibility может быть + (public), - (private), или # (protected).
* Методы: Записываются аналогично атрибутам, но с добавлением параметров: visibility name(parameter: type): returnType.
* Связи:
  + Ассоциация: Простая линия, показывающая связь между классами.
  + Зависимость: Пунктирная линия с открытым треугольником на конце.
  + Реализация: Пунктирная линия с открытым треугольником на конце.
  + Наследование: Сплошная линия с открытым треугольником на конце.
  + Агрегация: Сплошная линия с пустым ромбом на конце.
  + Композиция: Сплошная линия с заполненным ромбом на конце.



**4) Для чего применяются расширения диаграмм UML?**

Расширения диаграмм UML применяются для:

* Уточнения и дополнения стандартных элементов UML, чтобы учесть специфические требования и особенности конкретной области или системы.
* Внедрения новых элементов и отношений, которые не предусмотрены в стандартной нотации UML.

**5) Что означают понятия «стереотип» и «тегированное значение» в контексте расширенных диаграмм?**

**Стереотип**: Это механизм расширения UML, который позволяет создавать новые типы элементов на основе существующих, добавляя к ним дополнительные характеристики или поведение. Стереотип обозначается угловыми скобками (например, <<interface>>, <<service>>). Стереотипы помогают уточнить назначение и роль элементов в модели.

**Тегированное значение**: Это пара "имя-значение", которая может быть присвоена элементу модели (например, классам или стереотипам) для добавления дополнительных свойств или метаданных. Тегированные значения позволяют описывать элементы более детально и могут включать информацию, специфичную для домена, в котором используется модель. Например, можно использовать тегированное значение для указания версии компонента или автора класса.