Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование программного обеспечения»

Лабораторная работа №5

«Структурные диаграммы UML»

Студент: Стрелковская В. А.

ФИТ 3 курс 2 группа

Преподаватель: Якубенко К. Д.

# **Тема и цель работы**

В ходе данной лабораторной работы необходимо разработать диаграмму классов и диаграмму пакетов с использованием нотации UML (Unified Modeling Language):

* диаграмма классов – визуальное представление системы, в котором отображаются основные классы, их атрибуты и методы, а также взаимосвязи между классам;
* диаграмма пакетов – служит для организации и группировки классов и других элементов модели в логически связанные пакеты.

Эти диаграммы являются ключевыми инструментами для визуализации и проектирования программных систем, позволяя разработчикам и другим заинтересованным сторонам лучше понять структуру системы и ее компоненты.

Основной целью данной лабораторной работы является создание диаграммы классов, которая будет представлять структуру системы, включая классы, их атрибуты, методы и взаимосвязи между ними. Диаграмма классов позволяет разработчикам увидеть, как различные компоненты системы взаимодействуют друг с другом, а также понять, какие данные хранятся и как они обрабатываются.

Помимо диаграммы классов, необходимо также разработать диаграмму пакетов. Диаграмма пакетов используется для организации элементов модели в логические группы или пакеты, что позволяет упростить восприятие структуры системы и облегчить управление ее компонентами. Пакеты могут содержать классы, интерфейсы и другие элементы, и они помогают структурировать проект, делая его более модульным и удобным для работы.

В результате выполнения лабораторной работы необходимо иметь визуализированные диаграммы классов и пакетов, а также подробный отчет по проделанной работе. Это позволит не только продемонстрировать понимание структурных диаграмм UML, но и подтвердить способность применять эти знания на практике в процессе разработки программного обеспечения.

# **Описание функциональных требований**

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

1. Функционал для пользователя:

Для зарегистрированных пользователей приложение предлагает широкий спектр возможностей, направленных на удобство и улучшение пользовательского опыта:

* просмотр расписания сеансов и фильмов;
* просмотр информации о фильмах;
* онлайн покупка билетов на сеансы;
* выбор мест в зале при покупке билетов;
* просмотр истории бронирований и приобретенных билетов;
* авторизация пользователя.

1. Функционал для администратора:

Администраторы играют ключевую роль в управлении и поддержании работы приложения. Их функционал включает:

* управление расписанием сеансов и фильмами;
* добавление, редактирование и удаление информации о фильмах (включая название, описание, жанр и актеров);
* управление ценами на билеты.

1. Функционал для гостя:

Гостям предоставляется доступ к ограниченному набору функций, что позволяет им ознакомиться с приложением и принять решение о регистрации:

* регистрация;
* просмотр расписания сеансов и фильмов;
* просмотр информации о фильмах;
* поиск фильмов и сеансов по критериям.

# **Описание программных средств**

Для построения диаграмм IDEF0 использовался 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.

Draw.io предлагает интуитивно понятный интерфейс, который позволяет легко создавать и редактировать диаграммы. В функционал данного ресурса входит широкий спектр возможностей, включая:

* построение графиков и смысловых карт: пользователи могут визуализировать свои идеи и концепции, создавая понятные и наглядные схемы;
* UML-диаграммы: этот инструмент поддерживает создание различных типов UML-диаграмм, что делает его полезным для разработчиков программного обеспечения и системных аналитиков;
* диаграммы Венна: draw.io позволяет создавать диаграммы Венна для визуального представления пересечений и различий между наборами данных;
* Agile и Kanban доски: инструмент поддерживает методологии Agile, что позволяет командам эффективно управлять проектами и отслеживать прогресс;
* графики мозговых штурмов: пользователи могут организовывать свои идеи и генерировать новые концепции в удобной и визуально привлекательной форме;
* диаграммы архитектур технических систем: draw.io предоставляет возможности для создания сложных архитектурных схем, что полезно для инженеров и проектировщиков.

Одним из ключевых преимуществ draw.io является возможность совместной работы в реальном времени, что позволяет командам эффективно сотрудничать и вносить изменения одновременно. Пользователи могут делиться своими диаграммами с коллегами и работать над проектами совместно, не беспокоясь о лицензиях или ограничениях платформ.

Кроме того, draw.io обеспечивает высокий уровень конфиденциальности и безопасности данных, позволяя пользователям хранить свои диаграммы локально или в облачных хранилищах по своему выбору. Это делает инструмент идеальным для команд, стремящихся к безопасному управлению своей информацией.

Таким образом, использование draw.io для построения диаграмм IDEF0 не только облегчает процесс визуализации, но и предоставляет мощные инструменты для поддержки различных методологий и повышения эффективности работы команды.

# **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).

Диаграмма классов для сервиса «REALFILM» представлена на рисунке 4.1.

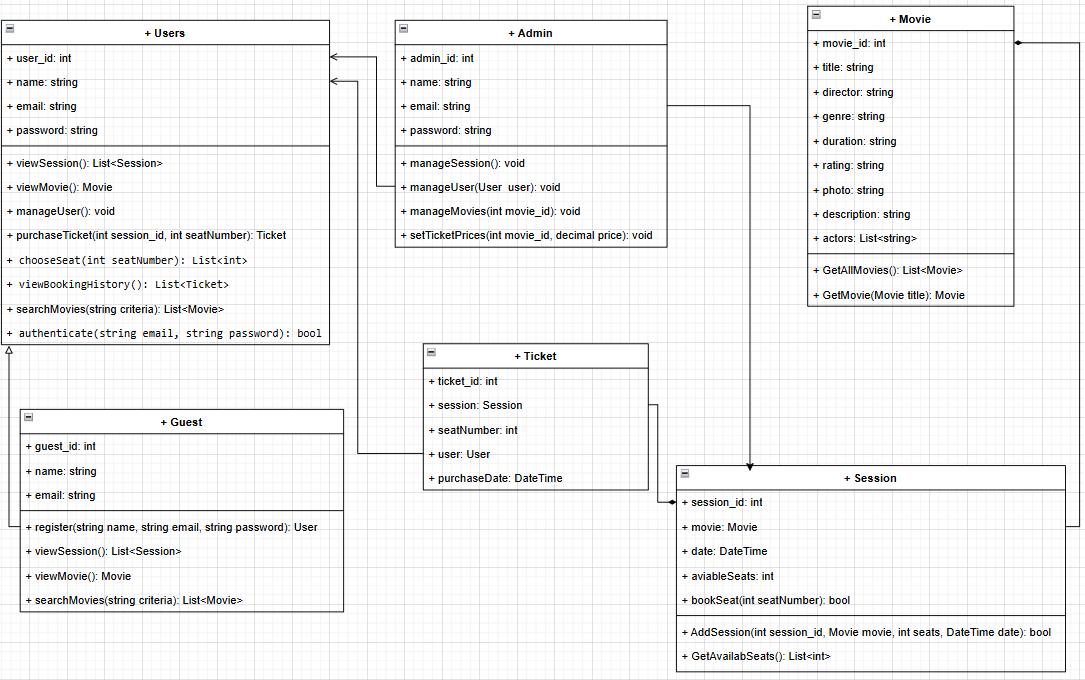


Рисунок 4.1 – Диаграмма классов

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 пакетов является мощным инструментом для визуализации и организации компонентов системы. Она помогает разработчикам лучше понимать архитектуру, управлять зависимостями и облегчает процесс разработки, делая проект более модульным и структурированным. Использование диаграмм пакетов способствует созданию гибких и легко поддерживаемых приложений, что в конечном итоге ведет к повышению качества программного обеспечения.

Диаграмма пакетов представлена на рисунке 4.2.

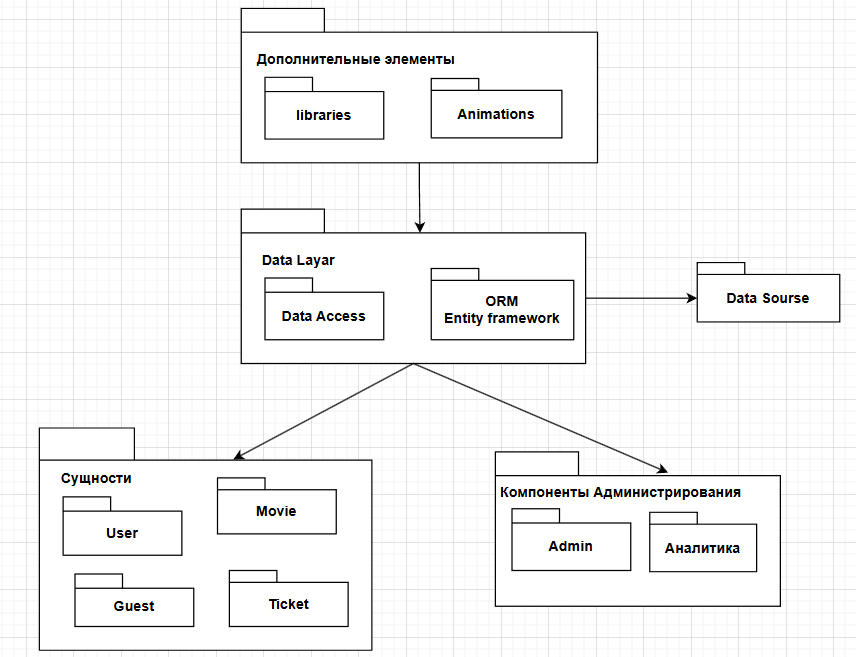
. 

Рисунок 4.2 – Диаграмма пакетов

Кроме того, диаграмма пакетов позволяет эффективно документировать структуру проекта, что упрощает процесс командной разработки и передачи знаний новым участникам. Визуальное представление архитектуры помогает быстрее ориентироваться в больших и сложных системах, а четкое разделение ответственности между пакетами снижает риск возникновения ошибок, связанных с нарушением модульности.

# **Ответы на вопросы**

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

В UML 2.0 включено несколько структурных диаграмм, каждая из которых фокусируется на описании статической архитектуры системы:

* диаграмма классов (Class Diagram) – описывает классы, их атрибуты, методы и связи между классами;
* диаграмма объектов (Object Diagram) – показывает конкретные экземпляры классов и их связи в определённый момент времени;
* диаграмма пакетов (Package Diagram) – группирует элементы системы (классы, компоненты) в пакеты для организации архитектуры;
* диаграмма компонентов (Component Diagram) – описывает физические модули системы и взаимодействие между ними;
* диаграмма развёртывания (Deployment Diagram) – демонстрирует распределение компонентов системы по узлам физической инфраструктуры;
* диаграмма композитной структуры (Composite Structure Diagram) – описывает внутреннюю структуру компонентов и классов и их взаимосвязи на более детальном уровне;
* диаграмма профилей (Profile Diagram) – используется для создания расширений UML и применяет пользовательские стереотипы и другие механизмы для конкретных областей.

1. Укажите назначение структурных диаграмм.

Структурные диаграммы в UML предназначены для моделирования статической архитектуры системы. Они фокусируются на описании:

* элементов системы (классов, объектов, компонентов, узлов и пакетов);
* иерархии и связей между элементами;
* взаимодействий и зависимостей между различными компонентами системы на уровне классов и физических модулей.

Эти диаграммы помогают разработчикам и архитекторам:

* чётко понимать структуру системы и её составляющих;
* описывать модели данных и взаимосвязи между объектами;
* разрабатывать архитектуру программного обеспечения с учётом требований к физической и логической структуре системы;

1. Опишите нотации, которые используются для построения Classes диаграмм.

Для построения диаграмм классов (Class Diagram) используются следующие основные нотации:

* 1. Класс – представляется прямоугольником, который разделён на три части: верхняя часть – имя класса, средняя часть – атрибуты (свойства) класса, нижняя часть – методы (операции) класса.
  2. Связи между классами:
* Ассоциация – линия между классами, которая обозначает, что один класс взаимодействует с другим. Может быть направленной (стрелка) или недиректной.
* Агрегация – полая ромбовидная стрелка, указывающая на «целое» и «части» (например, класс «Автомобиль» состоит из класса «Колесо»).
* Композиция – закрашенная ромбовидная стрелка, показывающая, что «часть» не может существовать без «целого» (например, «Комната» является частью «Дома»).
* Наследование (генерализация) – линия с треугольной стрелкой, указывающая на родительский класс, от которого наследуются дочерние классы.
* Зависимость – пунктирная линия со стрелкой, обозначающая, что один класс зависит от другого (например, использование одного класса в качестве параметра метода другого класса).
  1. Множественность – обозначает количество экземпляров, которые могут участвовать в связи. Например:
* «1..\*» – связь одного объекта с несколькими;
* «0..1» – связь с нулём или одним объектом.
  1. Интерфейс – изображается как прямоугольник с надписью <<interface>> или как круг (класс реализует интерфейс). Абстрактные классы и методы – обозначаются курсивом.

1. Для чего применяются расширения диаграмм UML?

Расширения UML применяются для того, чтобы адаптировать или настраивать язык UML под специфические потребности проекта или предметной области. Стандарт UML предоставляет гибкие механизмы для создания собственных обозначений и терминов, которые могут расширять его базовые элементы. Эти расширения используются для:

* добавления дополнительных смыслов к существующим элементам диаграмм;
* создания специальных профилей, которые адаптируют UML под конкретные технологии или предметные области (например, профиль для моделирования баз данных или веб-сервисов);
* оптимизации проектирования сложных систем, обеспечивая гибкость и поддержку конкретных требований.

1. Что означают понятия «стереотип» и «тегированное значение» в контексте расширенных диаграмм?

Стереотип – это один из ключевых механизмов расширения UML. Он используется для добавления дополнительной информации к существующим элементам диаграммы. Стереотипы позволяют классифицировать элементы, предоставляя им специфические свойства и поведение, присущие определённой области применения. Например, можно создать стереотип <<controller>> для обозначения классов, которые управляют взаимодействиями между другими объектами в MVC-архитектуре.

Класс с <<controller>> может представлять элемент системы, который управляет логикой обработки запросов.

Тегированное значение (tagged value) – это дополнительный атрибут или метка, которая присваивается элементу диаграммы для уточнения его характеристик. Тегированное значение дополняет информацию, которую нельзя выразить стандартными средствами UML. Каждый тег имеет имя и значение, которые можно назначить элементам модели.

Стереотипы и тегированные значения вместе создают мощный механизм для кастомизации и адаптации UML под нужды конкретных проектов.