Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа № 4.

«Объектно-ориентированное моделирование. UML – диаграммы поведения»

Студент: Подобед В.Г.

ФИТ 4 курс 6 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# Постановка задачи

# Тема и цель работы

Темой данной лабораторной работы является построение UML-диаграммы вариантов использования, необходимое для графического представления взаимосвязи ролей конечных пользователей и вариантов взаимодействий с системой, представленной в первой лабораторной работе по разработке электронной библиотеки.

Целью лабораторной работы является изучение методологии объектно-ориентированного моделирования средствами UML, ознакомление с основными принципами объектно-ориентированного проектирования программного обеспечения и получение навыков проектирования функциональности информационной системы с применением UML. Создание диаграмм вариантов использования позволит визуализировать сценарии взаимодействия пользователей с системой, что особенно важно для выявления потенциальных проблем и улучшения пользовательского опыта.

Процесс построения диаграмм вариантов использования также помогает выявить возможные сценарии, которые могли быть упущены на ранних этапах, тем самым минимизируя риски во время разработки. Также стоит отметить, что использование UML в проектировании позволяет создать общую визуальную основу для всех участников проекта, включая разработчиков, тестировщиков и бизнес-аналитиков. Это облегчает коммуникацию между ними, так как все могут опираться на одну и ту же модель системы.

В результате успешное выполнение данной лабораторной работы не только улучшит навыки работы с UML, но и поможет в дальнейшем развитии системы, обеспечивая её гибкость и адаптивность к изменяющимся условиям.

# Описание функциональных требований

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

Интерфейс веб-приложения должен быть тщательно продуманным и интуитивно понятным, чтобы пользователи могли легко ориентироваться в системе и эффективно использовать все предоставленные функции. Интерфейс должен способствовать комфортному взаимодействию, минимизируя необходимость в дополнительных усилиях и упрощая процесс выполнения задач. Также интерфейс обязан обеспечивать следующий функционал:

Функционал для гостя:

* регистрация;
* авторизация;
* поиск книги по определенным критериям;
* просмотр информации о книге;
* просмотр рейтинга книг.

Функционал для пользователя:

* скачивание книги;
* Оставление отзывов на книги;
* Добавление книги в избранное;
* Удаление книги из избранного;
* Сортировка каталога по фильтрам;
* Просмотр и обновление личных данных;
* Управление своими предпочтениями и избранными книгами.

Функционал для администратора:

* Управление пользователями (добавление, удаление пользователей);
* Управление книгами (добавление, изменение и удаление книг);
* Просмотр статистики использования системы;
* Управление профилями пользователей.

Основные страницы веб-приложения:

* Страница регистрации;
* Страница авторизации;
* Главная страница с поиском книг;
* Страница управления избранными книгами;
* Страница списка книг с фильтрацией;
* Страница с информацией о книге;
* Страница обратной связи;
* Страница редактирования профиля пользователя.

Таким образом, система электронной библиотеки не только предлагает удобный и разнообразный функционал для каждой роли, но и способствует повышению эффективности работы с книгами и улучшению пользовательского опыта.

Интуитивный интерфейс позволяет легко ориентироваться в системе, а высокий уровень безопасности обеспечивает защиту личных данных пользователей. Благодаря гибкости настройки каждый пользователь может персонализировать взаимодействие с библиотекой, а администраторам предоставляется возможность отслеживать статистику и управлять контентом.

Масштабируемость системы гарантирует стабильную работу при увеличении нагрузки, что делает электронную библиотеку надежным и функциональным инструментом для всех категорий пользователей.

# Описание программных средств

Для построения моделей было использовано программное средство Draw.io (также известное как diagrams.net). Draw.io — это многофункциональный инструмент, предназначенный для создания разнообразных графических схем, таких как диаграммы классов, диаграммы баз данных, блок-схемы, диаграммы деятельности, диаграммы процессов и многие другие.

Разработчиком Draw.io является компания JGraph Ltd. На момент использования проекта была задействована актуальная веб-версия программного обеспечения. Для доступа и использования данного инструмента можно посетить официальный сайт: <https://app.diagrams.net>.

Draw.io поддерживает интеграцию с облачными сервисами, такими как Google Drive, OneDrive и GitHub, что позволяет легко сохранять и управлять проектами. Кроме того, программное обеспечение поддерживает работу с локальными файлами и экспорт моделей в популярные форматы (PNG, PDF, SVG, XML), что делает его универсальным и удобным в использовании.

Инструмент поддерживает интеграцию с такими популярными сервисами, как Google Workspace, Microsoft Office 365, Confluence и Jira. Это позволяет пользователям бесшовно внедрять диаграммы и схемы в документы.   
Draw.io доступен в двух режимах:

1. Веб-версия — основной способ использования программного средства, работающий через любой современный браузер. Поддерживается на всех платформах, включая Windows, macOS, Linux, а также мобильные устройства под управлением Android и iOS.
2. Десктопная версия — доступна для скачивания и установки на операционные системы Windows, macOS и Linux. Обе версии имеют идентичный функционал, однако настольная версия может быть предпочтительна для пользователей, работающих с чувствительными данными, так как она поддерживает полностью автономную работу без подключения к интернету.

Draw.io предоставляет все необходимые средства для работы с различными моделями и диаграммами, применяемыми в инженерии программного обеспечения, таких как:

* UML диаграммы классов, последовательностей, прецедентов;
* ERD диаграммы (сущность-связь) для моделирования баз данных;
* BPMN диаграммы бизнес-процессов;
* диаграммы архитектуры программных систем и сетевых решений;
* блок-схемы для описания алгоритмов и процессов;
* диаграммы потоков данных и управления;
* схемы организации структуры данных.

Использование данного программного средства значительно упростило процесс проектирования и предоставило возможность создания наглядных схем, что способствует лучшему пониманию логики работы системы всеми заинтересованными сторонами проекта.

# Описание практического задания

В ходе выполнения практического задания по системе электронной библиотеки необходимо построить диаграмму вариантов использования. Основными составляющими диаграммы являются акторы, прецеденты и связи. Акторы представляют определённую роль, которую выполняют конечные пользователи системы, а прецеденты отображают варианты действий, которые могут выполнять акторы в системе.

Связи между элементами диаграммы можно разделить на использование (между актором и прецедентом), обобщение (прецеденты, свойственные нескольким акторам), включение (повторяющийся элемент в нескольких вариантах использования), и расширение (детализация основного процесса). Связи обобщения позволяют наследовать базовые функции между акторами, демонстрируя схожие возможности. Включение позволяет повторно использовать прецеденты в разных сценариях, а расширение добавляет дополнительную функциональность к уже существующему процессу.

Гость — незарегистрированный пользователь, который может выполнять базовые действия, такие как регистрация, авторизация, поиск книг, просмотр информации о книгах. Однако гостям не предоставляется возможность скачивать книги или оставлять отзывы.

Пользователь — зарегистрированный и авторизованный человек, который может управлять личным кабинетом, скачивать книги, добавлять книги в избранное, а также оставлять отзывы о книгах.

Администратор — пользователь с расширенными правами, управляющий всей системой. Он может добавлять, удалять или редактировать книги, а также управлять профилями пользователей.

Отношения обобщения на диаграмме показывают, что каждый последующий актор включает функционал предыдущего с добавлением уникальных возможностей. Например, пользователь имеет все возможности гостя, но с доступом к дополнительным функциям, таким как скачивание книг и управление избранным. Администратор включает функции пользователя и добавляет управление системой.

Кроме того, в системе можно использовать связи типа include и extend для определения деталей действий. Например, процесс регистрации может включать такие действия, как ввод данных пользователя и подтверждение по электронной почте. Связь extend может быть использована для детализации таких действий, как управление профилем пользователя, что может включать редактирование личных данных, изменение пароля и настройку уведомлений. Это гарантирует, что библиотека остаётся актуальной, а пользователи могут получать доступ только к разрешённым материалам. Для пользователя список доступных действий значительно расширяется.

Таким образом, система электронной библиотеки структурирована таким образом, чтобы каждый актор выполнял свою роль в соответствии с предоставленными правами и доступом. А использование различных видов связей позволяет проектировать более гибкие и многоуровневые процессы, что способствует удобству использования системы и её расширению по мере необходимости.

Диаграмма вариантов использования представлена на рисунке 3.1.

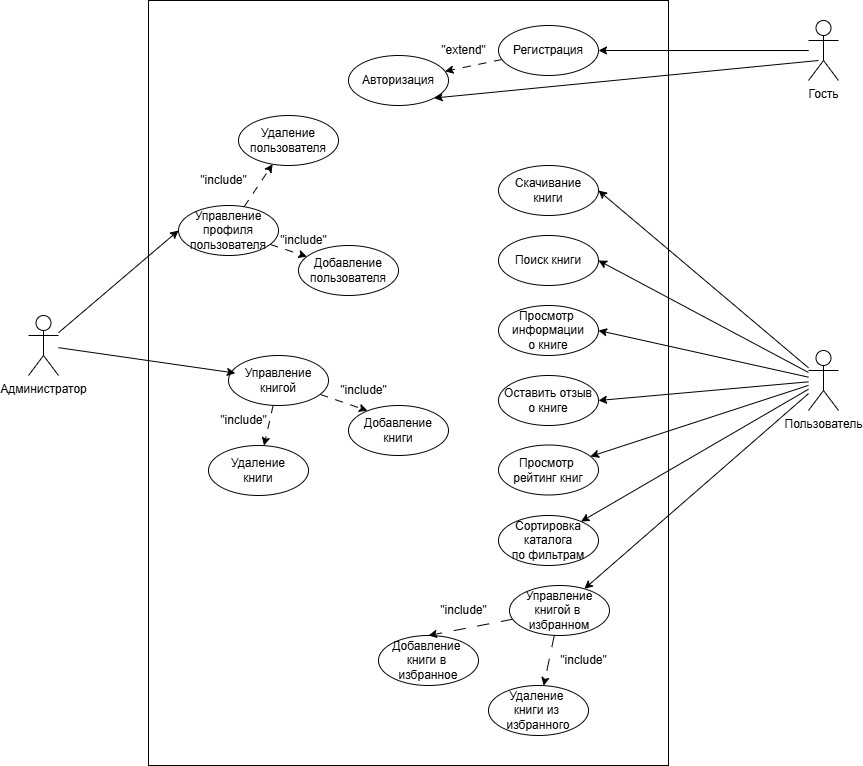


Рисунок 3.1 – Диаграмма вариантов использования

Рассмотрим подробнее прецеденты, выделенные в данной системе:

* регистрация (ассоциация с Гостем);
* авторизация (ассоциация со всеми акторами);
* поиск книги по критериям (ассоциация с Гостем и Пользователем);
* просмотр информации о книге (ассоциация с Гостем и Пользователем);
* скачивание книги (ассоциация с Пользователем);
* оставление отзыва (ассоциация с Пользователем);
* добавление книги в избранное (ассоциация с Пользователем);
* удаление книги из избранного (ассоциация с Пользователем);
* добавление книги в систему (ассоциация с Администратором);
* удаление книги из системы (ассоциация с Администратором);
* редактирование профиля пользователя (ассоциация с Администратором).

Связи include и extend используются для обозначения повторяющихся действий или детализации процесса. Например, прецедент «Скачивание книги» может быть включён в более сложные процессы, связанные с управлением библиотекой.

Таким образом, диаграмма вариантов использования отражает распределение ролей в системе, предоставляя каждому актору соответствующий набор прав и возможностей. Администратор имеет полный контроль над системой, а пользователи и гости могут взаимодействовать с библиотекой в соответствии с их уровнем доступа.

Прецеденты, связанные с расширением функций через extend, позволяют детализировать более сложные действия и указывать их зависимости от других процессов. Например, управление профилем пользователя может быть расширено за счёт таких опций, как изменение пароля или настройка уведомлений, что позволяет системе более гибко реагировать на запросы пользователей. Это делает систему электронной библиотеки более масштабируемой и позволяет добавлять новые функции без изменения базовой структуры.

Диаграмма вариантов использования также иллюстрирует иерархию ролей в системе. Каждая роль, будь то гость, пользователь или администратор, имеет определённый набор прав и возможностей. Связь обобщения между ролями указывает на то, что каждый последующий актор наследует возможности предыдущего. Это позволяет гостям системы просматривать книги, а пользователям — выполнять дополнительные действия, такие как скачивание книг и добавление их в избранное. Администратор, в свою очередь, получает полный доступ к системе с возможностью управлять как контентом, так и пользователями.

Прецеденты, относящиеся к роли Администратора, включают действия, необходимые для поддержания актуальности и безопасности библиотеки. Администратор может добавлять новые книги, удалять устаревшие записи, а также управлять профилями пользователей, обеспечивая соответствие библиотечных правил. Это гарантирует, что пользователи получают доступ к только актуальным материалам, а система остаётся безопасной и организованной.

Таким образом, система электронной библиотеки предоставляет гибкую и хорошо структурированную платформу, где каждая роль имеет чётко определённый набор обязанностей и прав. Применение связей include и extend помогает оптимизировать процессы, избегая избыточности и облегчая внесение изменений в систему. Диаграмма вариантов использования не только облегчает понимание структуры системы, но и позволяет разработчикам эффективно проектировать и масштабировать систему, ориентируясь на потребности пользователей и поддерживая высокий уровень контроля администратора.

# Ответы на теоретические вопросы

# Перечислите структурные диаграммы, которые входят в UML 2.0.

Структурные диаграммы в UML 2.0: В UML 2.0 существует несколько типов структурных диаграмм, которые описывают статические аспекты системы, такие как классы, объекты, компоненты и их взаимосвязи. Эти диаграммы включают:

* диаграмма классов (Class Diagram);
* диаграмма объектов (Object Diagram);
* диаграмма компонентов (Component Diagram);
* диаграмма развертывания (Deployment Diagram);
* диаграмма пакетов (Package Diagram);
* диаграмма композитной структуры (Composite Structure Diagram);
* диаграмма профилей (Profile Diagram).

# Укажите назначение структурных диаграммы

Структурные диаграммы UML предназначены для отображения статической структуры системы на различных уровнях детализации. Их цель — показать классы, компоненты, пакеты и другие элементы системы, а также их взаимосвязи. Основное назначение каждой диаграммы:

* диаграммы классов — отображают структуры классов и их отношения (ассоциации, обобщения и композиции);
* диаграммы объектов — представляют конкретные экземпляры классов в определенном состоянии;
* диаграммы компонентов — помогают визуализировать, как модули системы взаимодействуют друг с другом;
* диаграммы развертывания — описывают физическое развертывание системы на оборудовании;
* диаграммы пакетов — группируют элементы системы, обеспечивая модульную структуру;
* диаграммы композитной структуры — отображают внутренние части классов и их взаимодействия;
* диаграммы профилей — позволяют добавлять специализированные элементы для конкретных областей приложения.

# Опишите нотации, которые используются для построения Classes диаграмм

В диаграммах классов используется несколько ключевых нотаций:

* класс — прямоугольник, разделенный на три части: верхняя часть содержит имя класса, средняя — атрибуты (с переменными или данными), нижняя — методы или операции;
* ассоциация — линия, соединяющая классы, которая показывает, как они взаимодействуют. Могут быть добавлены кратности (например, 1..\*, 0..1), чтобы указать количество объектов;
* обобщение (наследование) — пустая треугольная стрелка, указывающая на суперкласс;
* реализация (интерфейс) — пунктирная линия с треугольной стрелкой, указывающей на интерфейс, который класс реализует;
* агрегация — линия с пустым ромбом на стороне класса-владельца, показывающая отношение «часть-целое»;
* композиция — линия с закрашенным ромбом, указывающая, что объект полностью владеет своими частями;
* зависимость — пунктирная линия со стрелкой, показывающая, что один элемент зависит от другого;
* модификаторы доступа — используются для обозначения уровня видимости атрибутов и методов: + (public), - (private), # (protected), ~ (package).

# Для чего применяются расширения диаграмм UML?

Расширения в UML используются для добавления специфических возможностей и элементов, которые не предусмотрены стандартными диаграммами. Они позволяют адаптировать UML для различных доменов и бизнес-сценариев.

Расширения включают:

* стереотипы — для добавления новых видов элементов в диаграммы, которые могут иметь специальные свойства;
* тегированные значения — для хранения дополнительных данных или атрибутов, не включенных в стандартные элементы;
* ограничения — для выражения условий или правил, которые должны быть выполнены (например, на атрибуты или ассоциации).

# 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. Стереотипы визуально обозначаются угловыми скобками (например, <<controller>>, <<entity>>). Они могут использоваться для спецификации, например, ролей элементов, таких как «Контроллер» или «Сущность» в архитектурных моделях.

Тегированное значение (tagged value) — это дополнительный атрибут или метка, которую можно прикрепить к элементу модели (например, классу или ассоциации) для хранения дополнительных данных. Эти значения могут использоваться для хранения информации, важной для домена, но не предусмотренной в стандартных элементах UML (например, «версия», «автор»).

Стереотипы и тегированные значения позволяют расширять функциональность UML и адаптировать его под специфические нужды проекта.

# Заключение

В рамках данной лабораторной работы была разработана диаграмма вариантов использования для системы электронной библиотеки. Основной целью было получение практических навыков построения диаграмм с использованием унифицированного языка моделирования (UML) для проектирования функциональной структуры системы. Эти навыки важны для успешного проектирования и разработки информационных систем, так как UML предоставляет стандартизированный подход к моделированию процессов и сценариев взаимодействия пользователей с системой.

На этапе проектирования были смоделированы основные сценарии взаимодействия пользователей с системой, включая три ключевых актера: Гость, Пользователь и Администратор. Каждый актор обладает своими правами и выполняет определённые действия. Например, Гость может искать и просматривать книги, а также фильтровать и сортировать результаты. Однако для скачивания книг или оставления отзыва ему необходимо зарегистрироваться и авторизоваться, став Пользователем. Администратор, в свою очередь, имеет расширенные права, позволяющие ему управлять пользователями и книгами в системе.

Моделирование различных вариантов использования помогло лучше понять взаимодействие пользователей с системой. Например, процесс управления книгами для администратора включает добавление новой книги, редактирование информации о ней, а также удаление книг из каталога. При моделировании были выявлены связи обобщения и включения между прецедентами. Например, Пользователь наследует базовые возможности Гостя (такие как поиск и просмотр книг), но дополняется дополнительными функциями — скачиванием книг и управлением избранным. Это упрощает структуру системы и снижает дублирование действий.

Диаграмма вариантов использования структурирует функциональность системы и помогает организовать бизнес-процессы. Она предоставляет визуальное представление о взаимодействиях между пользователями и системой, что помогает разработчикам и менеджерам проектов лучше понимать работу системы. Это позволяет заранее выявить возможные проблемы, оптимизировать процесс разработки и более точно определить требования к системе. Таким образом, диаграмма вариантов использования помогает разработчикам заранее выявить возможные ошибки и недочеты.

Таким образом, построение диаграммы вариантов использования для системы электронной библиотеки стало важным этапом в проектировании системы. Этот подход позволил учесть все сценарии взаимодействия пользователей с системой, что значительно упростило процесс разработки и обеспечило высокое качество конечного продукта, ориентированного на удобство использования и эффективность.