**Министерство образования и науки Забайкальского края   
Государственное профессиональное образовательное учреждение  
«ЧИТИНСКИЙ ТЕХНИКУМ ОТРАСЛЕВЫХ ТЕХНОЛОГИЙ И БИЗНЕСА»  
(ГПОУ «ЧТОТиБ»)**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине «Технология разработки программного обеспечения»

На тему «ВИДЕОПРОКАТ»

|  |  |
| --- | --- |
|  | Выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Зубов И.И.,  студент группы ИСиП 19-1п  «\_\_\_» \_\_\_\_\_\_\_\_\_\_2022 г.  Проверил работу:  \_\_\_\_\_\_\_\_\_\_\_\_\_Ломов Д.А.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_2022 г. |

2022 г.

**Оглавление**

[Введение 3](#_Toc105085440)

[1. Теоретическая часть 4](#_Toc105085441)

[1.1. Диаграмма вариантов использования 4](#_Toc105085442)

[1.2. Диаграмма последовательности 5](#_Toc105085443)

[1.3. Диаграмма коммуникации 6](#_Toc105085444)

[1.4. Диаграмма деятельности 7](#_Toc105085445)

[1.5. Диаграмма классов 8](#_Toc105085446)

[1.6. Диаграмма состояний 9](#_Toc105085447)

[1.7. Диаграмма размещения 10](#_Toc105085448)

[1.8. Диаграмма компонентов 11](#_Toc105085449)

[2. Практическая часть 12](#_Toc105085450)

[2.1. Диаграмма вариантов использования 12](#_Toc105085452)

[2.2. Диаграмма последовательности  13](#_Toc105085453)

[2.3. Диаграмма коммуникации 14](#_Toc105085454)

[2.4. Диаграмма деятельности 15](#_Toc105085455)

[2.5. Диаграмма классов 16](#_Toc105085456)

[2.6. Диаграмма состояний 17](#_Toc105085457)

[2.7. Диаграмма размещения 18](#_Toc105085458)

[2.8. Диаграмма компонентов 19](#_Toc105085459)

[Заключение 20](#_Toc105085460)

[Список использованных источников 21](#_Toc105085461)

# **Введение**

UML - является самой современной и продвинутой технологией в области программной инженерии. И несомнено, является актуальной деятельностью в современном обществе. Ведь UML позволяет программистам представлять свое видение системы в виде набора стандартных диаграмм, которые, к тому же, служат отличным средством коммуникации в команде разработчиков и прекрасным помощником в общении с заказчиком. И при всем этом, UML - достаточно логичная и простая для изучения нотация, навыками использования которой, без сомнения, должен овладеть любой специалист, собирающийся работать в области программной инженерии. Знание UML нужно разработчикам, системным архитекторам, менеджерам и т.д.

Цель курсового проекта – закрепить полученные знания на парах, а также научиться создавать диаграммы UML на предоставленную тему.

Задачи курсового проекта:

* Создать диаграмму вариантов использования
* Создать диаграмму последовательности
* Создать диаграмму коммуникации
* Создать диаграмму деятельности
* Создать диаграмму классов
* Создать диаграмму состояний
* Создать диаграмму размещения
* Создать диаграмму компонентов
* Составить документацию

1. Теоретическая часть
   1. Диаграмма вариантов использования

Диаграмма прецедентов или диаграмма вариантов использования (англ. use case diagram) в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмму вариантов использования есть смысл строить во время изучения технического задания, она состоит из графической диаграммы, описывающей действующие лица и прецеденты, а также спецификации, представляющего собой текстовое описание конкретных последовательностей действий (потока событий), которые выполняет пользователь при работе с системой. Спецификация затем станет основой для тестирования и документации, а на следующих этапах проектирования она дополняется и оформляется в виде диаграммы (в рамках ICONIX используется диаграмма последовательности, но в UML для этого имеются также диаграммы деятельности). Кроме того, use-case диаграмма достаточно проста, чтобы ее мог понять заказчик, следовательно, вы можете использовать ее для согласования ТЗ (ведь диаграмма описывает функциональные требования к системе).

На диаграмме использования изображаются:

* акторы — группы лиц или систем, взаимодействующих с нашей системой;
* варианты использования (прецеденты) — сервисы, которые наша система предоставляет акторам;
* комментарии;
* отношения между элементами диаграммы.
  1. Диаграмма последовательности

Диаграмма последовательности (англ. sequence diagram) — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента.

Диаграммы последовательности следует применять тогда, когда требуется посмотреть на поведение нескольких объектов в рамках одного прецедента. Диаграммы последовательности хороши для представления взаимодействия объектов, но не очень подходят для точного определения поведения.

Одно из преимуществ диаграммы последовательности заключается в том, что почти не придется объяснять ее нотацию.

В UML-диаграмме описанной последовательности будут участвовать объекты:

* Пользователь;
* страница оплаты сайта учебного центра (Сайт УЦ);
* СУБД УЦ;
* Платежный шлюз;
* Антифрод-система.

Взаимодействие между ними на диаграмме будет отображено в виде прямых сигналов (сплошные стрелки) и ответов (пунктирные) в рамках линии жизни объекта, показанной как длинный вертикальный прямоугольник.

UML-диаграмма последовательности позволяет достаточно наглядно показать взаимодействие между разными объектами, детализируя какими сигналами прямыми и ответными они обмениваются. Чаще всего это требуется для иллюстрации интерактивного взаимодействия между разными сервисами или объектами одной системы, например, когда при регистрации клиентского обращения запускается задача ответственному сотруднику на его обработку.

* 1. Диаграмма коммуникации

Диаграмма коммуникации (англ. communication diagram, в UML 1.x — диаграмма кооперации, collaboration diagram) — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между объектами, а время как отдельное измерение не используется (применяются порядковые номера вызовов).

Диаграмма коммуникации моделирует взаимодействия между объектами или частями в терминах упорядоченных сообщений. Коммуникационные диаграммы представляют комбинацию информации, взятой из диаграмм классов, последовательности и вариантов использования, описывая сразу и статическую структуру, и динамическое поведение системы.

Коммуникационные диаграммы имеют свободный формат упорядочивания объектов и связей как в диаграмме объектов. Чтобы поддерживать порядок сообщений при таком свободном формате, их хронологически нумеруют. Чтение диаграммы коммуникации начинается с сообщения 1.0 и продолжается по направлению пересылки сообщений от объекта к объекту.

Диаграммы коммуникации обладают двумя признаками, которые отличают их от диаграмм последовательности. Во-первых, здесь отмечен путь (path), который отображается соответственно ассоциации. Впрочем, можно также показать его в соответствии с локальными переменными, параметрами, глобальными переменными и обращениями к самому себе. Путь представляет источник информации для объекта.

Во-вторых, здесь имеется порядковый номер. Чтобы указать порядок сообщения во времени, вы предваряете его номером начиная с 1 и далее – в арифметической прогрессии для каждого нового сообщения в потоке управления (2, 3, ...). Чтобы показать вложенность, применяется десятичная система классификации Дьюи (1 – первое сообщение, которое содержит в себе сообщения 1.1, 1.2 и т.д.). Количество уровней вложения не ограничивается. Кроме того, следует отметить, что на линии одной и той же ссылки можно показать множество сообщений (возможно, пересылаемых в разных направлениях), и каждое будет иметь уникальный порядковый номер.

* 1. Диаграмма деятельности

Диаграмма активности UML позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

Диаграммы активности UML также могут быть использованы для отображения потока событий в бизнес-процессе. Они могут быть использованы для изучения бизнес-процессов с целью определения их потока и требований.

Диаграммы деятельности можно считать частным случаем диаграмм состояний. Именно они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий. Метамодель UML предоставляет для этого необходимые термины и семантику. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции некоторого класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

В контексте языка UML деятельность (activity) представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения.

* 1. Диаграмма классов

Диаграмма классов (от англ. "class diagram") предназначена для представления внутренней структуры программы в виде классов и связей между ними.

Все сущности реального мира, с которыми собирается работать программист, должны быть представлены объектами классов в программе. При этом у каждого класса должно быть только одно назначение и уникально осмысленное имя, которое будет связано с этой целью.

Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

* имя класса
* атрибуты (свойства) класса
* операции (методы) класса.

Для атрибутов и операций может быть указан один из трех типов видимости:

* private (частный)
* protected (защищенный)
* public (общий)

Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента.

Каждый класс должен обладать именем, отличающим его от других классов. Имя – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк.

На практике обычно используются краткие имена классов, взятые из словаря моделируемой системы. Каждое слово в имени класса традиционно пишут с заглавной буквы (верблюжья конвенция), например, Sensor (Датчик) или TemperatureSensor (ДатчикТемпературы).

* 1. Диаграмма состояний

Диаграмма состояний является одной из пяти диаграмм UML, используемых для моделирования динамической природы системы. Они определяют различные состояния объекта в течение его жизни, и эти состояния изменяются событиями. Диаграммы состояний полезны для моделирования реактивных систем. Реактивные системы могут быть определены как система, которая реагирует на внешние или внутренние события.

Диаграмма состояний описывает поток управления из одного состояния в другое. Состояния определяются как состояние, при котором объект существует, и он изменяется при запуске какого-либо события. Наиболее важной целью диаграммы состояний является моделирование времени жизни объекта от создания до завершения.

Диаграммы состояний также используются для прямого и обратного проектирования системы. Однако основной целью является моделирование реактивной системы.

В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта.

Хотя диаграммы состояний чаще всего используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Понятие автомата в контексте UML обладает довольно специфической семантикой, основанной на теории автоматов. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели.

* 1. Диаграмма размещения

Диаграммы размещения – это один из двух видов диаграмм, используемых при моделировании физических аспектов объектно-ориентированной системы. Такая диаграмма представляет конфигурацию узлов, где производится обработка информации, и показывает, какие артефакты размещены на каждом узле.

Диаграммы размещения используются для моделирования статического представления системы с точки зрения размещения. В основном под этим понимается моделирование топологии аппаратных средств, на которых работает система. По существу, диаграммы размещения – это просто диаграммы классов, сосредоточенные на системных узлах.

Диаграммы размещения важны не только для визуализации, специфицирования и документирования встроенных, клиент-серверных и распределенных систем, но и для управления исполняемыми системами с использованием прямого и обратного проектирования.

При создании программной системы вы как разработчик программного обеспечения обращаете внимание в первую очередь на архитектуру и размещение своих программ в вычислительной среде. Но в качестве системного инженера вы заинтересованы главным образом в аппаратных и программных средствах системы и в том, как достичь оптимального их сочетания. Иными словами, разработчики программного обеспечения имеют дело с неосязаемыми артефактами вроде модели и кода, а разработчики систем – еще и с аппаратурой, вполне осязаемой.

На диаграмме размещения (deployment diagram) показана конфигурация обрабатывающих узлов и артефактов, размещенных в этих узлах. Диаграмма размещения представлена в виде графа с ребрами и вершинами.

Диаграмма размещения обладает общими свойствами, присущими всем диаграммам, – именем и графическим содержанием, которое отражает одну из проекций модели. От других диаграмм отличается своим специфическим наполнением.

* 1. Диаграмма компонентов

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

* Визуализации общей структуры исходного кода программной системы.
* Спецификации исполнимого варианта программной системы.
* Обеспечения многократного использования отдельных фрагментов программного кода.
* Представления концептуальной и физической схем баз данных.

В разработке диаграмм компонентов участвуют как системные аналитики и архитекторы, так и программисты. Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

1. Практическая часть
2. 1. Диаграмма вариантов использования

Первая диаграмма, с которой начинается всё проектирование, является диаграммой вариантов использования. Она представлена на рисунке 1.

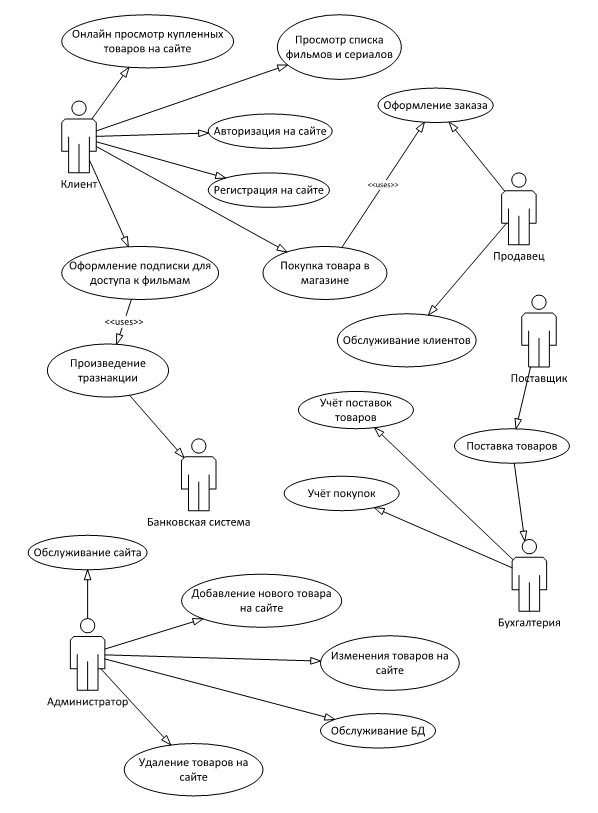


Рисунок 1 – Диаграмма вариантов использования

На ней можно выделить несколько акторов:

* Клиент или пользователь заинтересованный в товаре
* Администратор кампании
* Продавец кампании
* Поставщик товаров
* Бухгалтерия кампании

По диаграмме можно определить, что клиент может производить как регистрацию, так и авторизацию на сайте кампании. Также неспосредственно клиент может просматривать фильмы при оформлении подписки. Само оформление происходит через транзакцию, которую помогает осуществить банковская система. Ещё клиент может приобрести товар в магазине кампании. Для этого ему помогает продавец кампании, который осуществляет процесс покупки, т.е. создаёт заказ. Продавец, в частности, может помочь клиенту, в случае, если он нуждается в ней. Это называется, как обслуживание клиентов.

Администратор, в свою очередь, занимается осблуживанием сайта и базы данных, в которой хранятся все данные о товарах, покупателях, заказах и т.д. Также, он имеет право вносить изменения, добавлять и удалять товары и все данные, хранящихся в БД, в целом.

На данной диаграмме, мы можем также заметить и бухгалтерию, которая ведёт учёт товаров и поставок в магазине. Поставщики, непосредственно, помогают осуществлять поставки товаров.

* 1. Диаграмма последовательности

Вторая диаграмма, с которой происходит продолжение проектирования – это диаграмма последовательности. Работа представлена на рисунке 2.

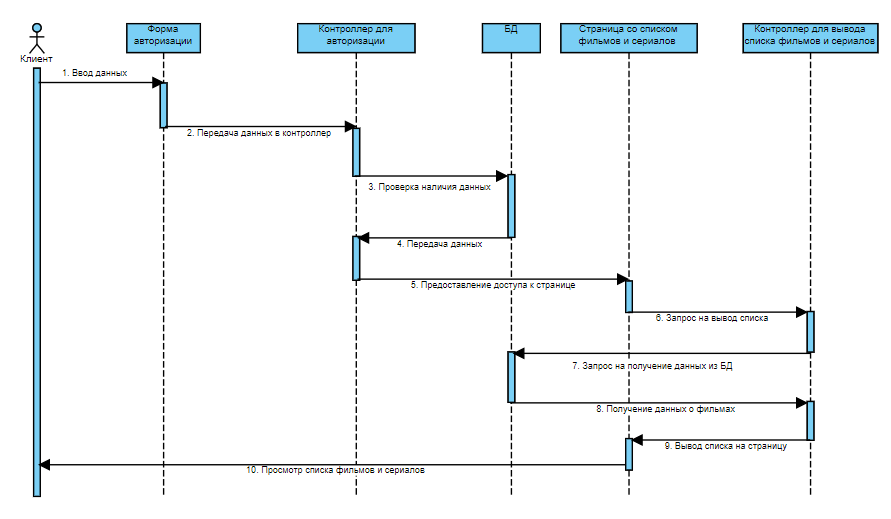


Рисунок 2 – Диаграмма последовательности

На ней представлен один из вариантов использования с небольшим дополнением, а именно это авторизация и просмотр списка фильмов. На этой диаграмме мы видим несколько линий жизни разных объектов. А конкретно это:

* Форма авторизации
* Контроллер метода авторизации
* База данных
* Страница со списком фильмов и сериалов
* Контроллер для вывода списков фильмов и сериалов

Также представлен клиент, как физическое лицо, взаимодействующее со всеми этими компонентами диаграммы.

Смотря на построенную диаграмму, можно понять действия определённых компонентов системы. Кто возвращает результат запроса, кто создаёт запрос и, кто как отсылается к друг другу. Видно также, последовательность действий, выполняющихся друг за другом.

* 1. Диаграмма коммуникации

Следующая диаграмма представлена на рисунке 3.

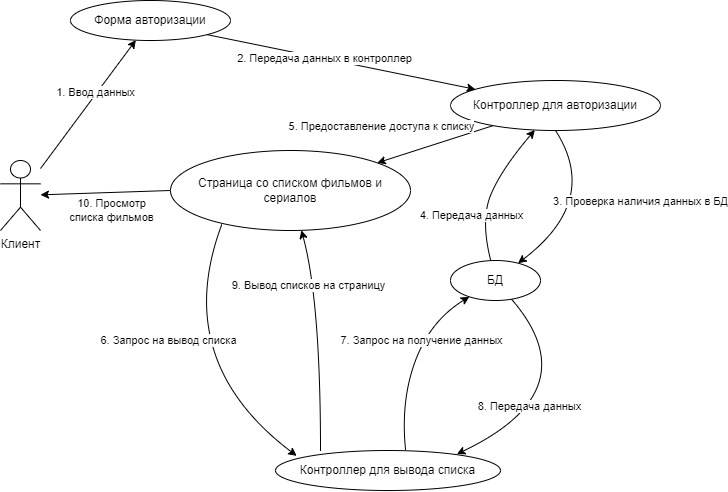


Рисунок 3 – Диаграмма коммуникации

Это диаграмма коммуникации. По своей сути, это тоже самое что и диаграмма последовательности, но в другой форме. В ней мы просматриваем объекты с точки зрения их нагруженности, т.е. на ней мы можем отследить какой элемент системы нагружен больше всего и как это можно исправить. Для этого и существует диаграмма коммуникации. И если в диаграмме последовательности важно время событий, то здесь это теряет смысл. А здесь мы можем увидеть, как отражается распределение процессов между объектами и их зависимости друг от друга.

* 1. Диаграмма деятельности

Следующая диаграмма – это диаграмма деятельности. На ней мы видим некое подобие блок-схемы, в которой описаны определённые действия, после определённых условий. Данная диаграмма помогает понять, как происходит процесс офрмления покупки в магазине видепрокатов. Кругами изображены начало и конец, а прямоугольниками – действие. Ромбами – условие, а стрелочками – переход к следующему действию. При проишествии одного условия, от него идут дорожки, которые в зависимости от разных вариантов, идут к разным действиям и соответсвенно выполняют разные операции.

Диаграмма деятельности представлена на рисунке 4.

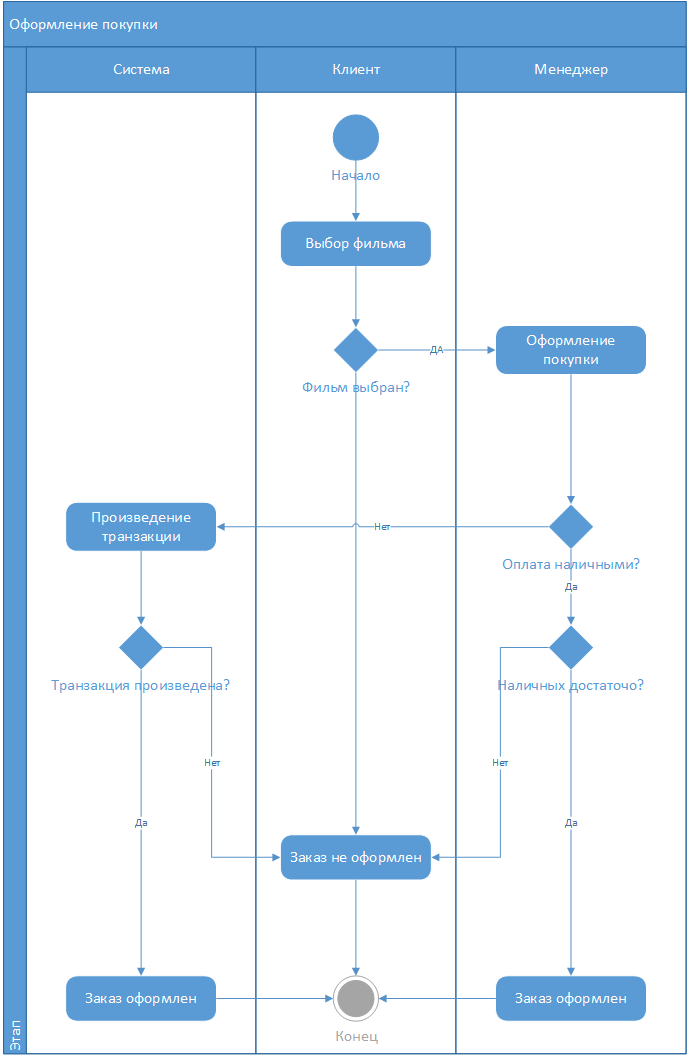


Рисунок 4 – Диаграмма деятельности

* 1. Диаграмма классов

Далее идёт процесс разработки диаграммы классов. Она представлена на рисунке 5. По ней можно понять внутренюю структуру приложения. В данной диаграмме описываются 5 классов приложения. Все классы связаны между собой разными зависимостями. А сами классы состоят из атрибутов и методов. Сверху квадратного блока представлены атрибуты, снизу методы. У каждого метода или поля есть модификатор доступа, «-» – это private, а «+» – public. Также у каждого атрибута указан тип данных. Всего их в данной диаграмме 4 вида (не считая специальных типов данных):

* String – поле будет являтся строкой.
* Int – поле будет являтся числом.
* Date – поле будет являтся датой.
* Time – поле будет являтся временем.

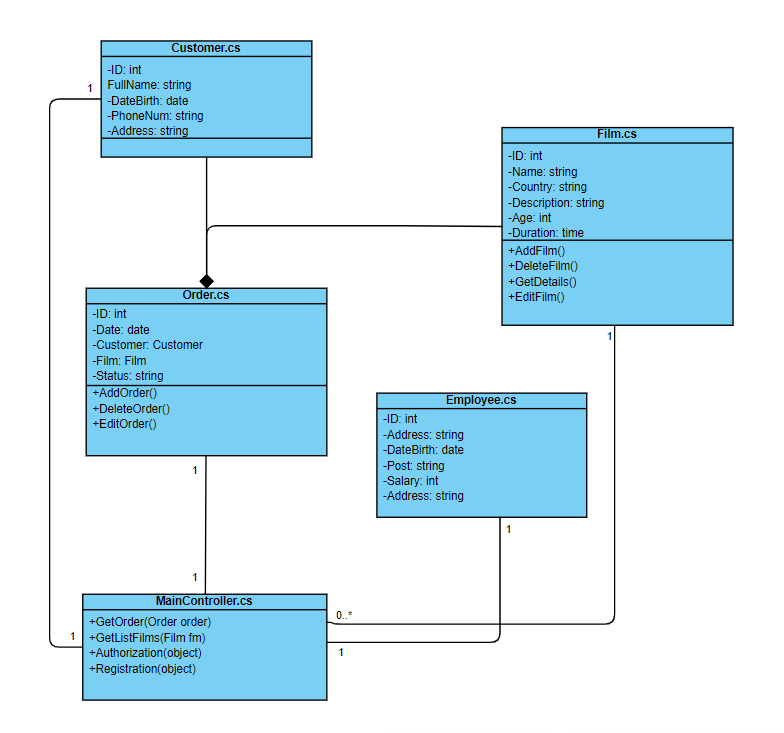


Рисунок 5 – Диаграмма классов

Специальными типами данных являются отдельные классы, использующиеся в определении атрибутов. Такой атрибут можно увидеть в классе «Order». Также в этой диаграмме отображаются зависимости между классами. Так, например, класс «Order» имеет отношение 1 ко 1 к классу «MainController». То есть один объект, к одному объекту. А вот уже в связи между классами «Film» и «MainController» имеется связь 1 ко многим (0..\*). Это связь показывает, что в классе «MainController» может быть несколько объектов из класса «Film».

С помощью этой диаграммы, мы можем понять какой класс являтся перегруженным, а также понять, какую зависимость будет целесообразнее использовать в данной системе.

* 1. Диаграмма состояний

Следлующей диаграммой является диаграмма состояний. Её представление изображено на рисунке 6.

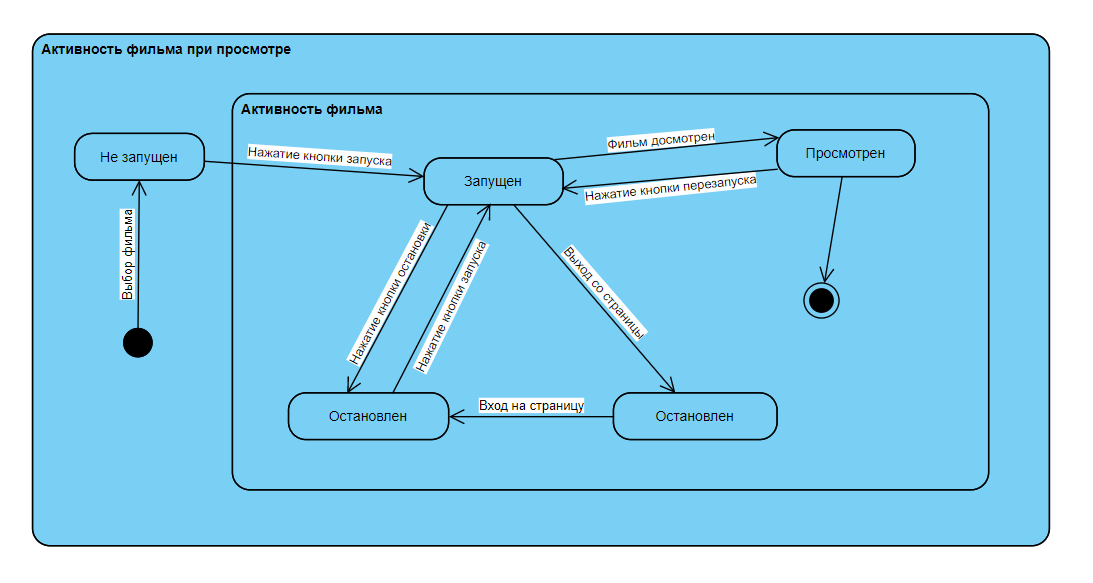


Рисунок 6 – Диаграмма состояний

С помощью этой диаграммы, можно отследить состояние какого-либо объекта. В данном случае это состояние фильма при просмотре. По ней можно определить детали процесса, связанными с запуском фильма и его просмотре. Когда фильм запущен, когда просмотрен, а когда вообще остановлен. По сути это описание работы определённого метода какого-либо класса, в котором выполняется этот процесс.

Сама диаграмма состоит из черных кругов, обозначающих начало и конец, прямоугольников, обозначающих состояние, и действий, изображенных на рисунке стрелками. То еть при нажатии кнопки запуска, фильм меняет своё состояние с «не запущенный» на «запущен». Таким образом мы отслеживаем состояние фильма при просмотре, что в дальнейшем поможет разработичкам правилно написать метод для просмотра фильма.

* 1. Диаграмма размещения

Создание следующей диаграммы подразумевает проектирование физического аппаратного и программного обеспечения системы. По ней мы понимаем, как система будет физически развернута на аппаратном обеспечении.

В данном случае у нас есть две стороны клиентской части. Одна часть – администраторская, то есть пользоваться и подключаться к общей сети будут администраторы кампании, через официальное приложение. А вторая часть – это клиентская, то есть уже пользуются обычные люди, используя сайт или приложение.

Структура диаграммы состоит из блоков, обозначающих определённый компонент сети, и стрелками, обозначающих соедеинение между компонентами. В нашем случае есть главный сервер-обработчик, в котором работает приложение и сайт, который в свою очередь подключается к серверу с базой, забирая нужные данные.

Суть диаграммы, показать топологию системы и распределение компонентов системы по её узлам, а также соединения - маршруты передачи информации между аппаратными узлами.

Диаграмма размещения показана на рисунке 7.

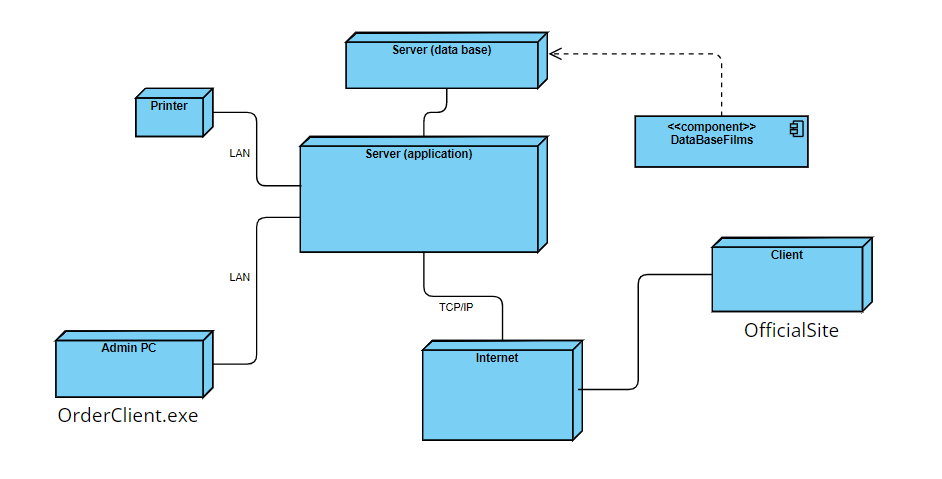


Рисунок 7 – Диаграмма размещения

* 1. Диаграмма компонентов

И последняя диаграмма в этой курсовой работе – это диаграмма компонентов. Её представление изображено на рисунке 8.

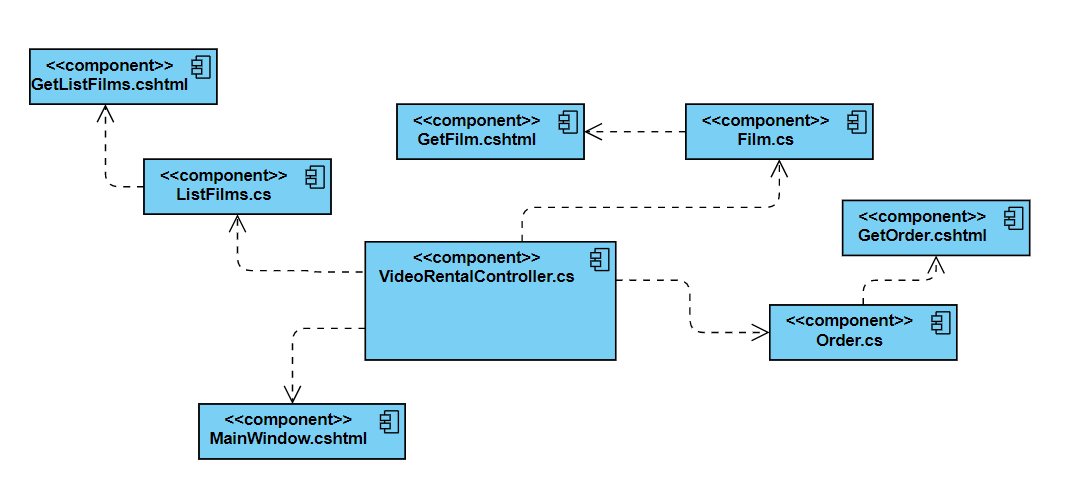


Рисунок 8 – Диаграмма компонентов

Диаграмма компонентов показывает, как выглядит модель на физическом уровне. На ней изображаются компоненты программного обеспечения системы и связи между ними.

В данной диаграмме, мы видим главный класс «VideoRentalController», от которого идут зависимоти ко всем другим классам. Каждый класс занимается своей задачей. Так, например, класс «ListFilms» выводит список фильмов через представление «GetlistFilms.cshtml».

Данная диаграмма обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения.

# **Заключение**

Значение языка UML существенно возрастает, поскольку он все более приобретает черты языка представления знаний. При этом наличие в языке UML изобразительных средств для представления структуры и поведения модели позволяет достичь адекватного представления декларативных и процедурных знаний и, что не менее важно, установить между этими формами знаний семантическое соответствие. Все эти особенности языка UML позволяют сделать вывод о том, что он имеет самые серьезные перспективы уже в ближайшем будущем.

Поставленные задачи во время выполнения курсового проекта выполнены:

* Создано 8 видов диаграмм UML на тему «Видеопрокат»
* Отработаны навыки в создании диаграмм
* Материал, предоставленный на парах, закреплён
* Отработаны навыки в написании документации

Во время выполнения курсового проекта, трудностей не возникло. Всё было предельно понятно и ясно.

Данный курсовой проект является актуальной деятельностью в нашем современном обществе. Полученные навыки, несомненно, пригодятся в нашей профессиональной деятельности.

# **Список использованных источников**

1. Онлайн библиотека гуманитарной и технической литературы TELENIR [Электронный ресурс] – Режим доступа: http://www.telenir.net/uchebniki/samouchitel\_uml/p10.php – свободный.
2. Онлайн блок о технологии разработки ПО. [Электронный ресурс] – Режим доступа: https://technologiarpo.blogspot.com/p/blog-page\_16.html - свободный.
3. Официальный сайт кампании Microsoft. [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling – свободный.
4. Сайт, на котором можно найти уроки, учебники и другие материалы по программированию. [Электронный ресурс] – Режим доступа: https://coderlessons.com/tutorials/akademicheskii/uchit-uml/uml-diagrammy-sostoianii# - свободный.
5. Сайт об UML для бизнес-моделирования. [Электронный ресурс]. – Режим доступа: https://evergreens.com.ua/ru/articles/uml-diagrams.html - свободный.
6. Сайт IT-GOST.RU – электронная библиотека для технических писателей, занимающихся оформлением проектной документации. [Электронный ресурс] – Режим доступа: http://www.it-gost.ru/articles/view\_articles/94 - свободный.
7. Свободная онлайн энциклопедия WIKIPEDIA. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_прецедентов - свободный.