МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Чувашский государственный университет имени И. Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Пояснительная записка к курсовому проекту

По дисциплине: «Технологии разработки программного обеспечения»

На тему: Применение объектно-ориентированного подхода к анализу и

проектированию программного обеспечения

|  |
| --- |
| **Выполнил**:  студент группы ИВТ-41-15  Быков Н.Ю.  **Проверил**: доцент  Ржавин В.В. |

г. Чебоксары, 2019 г.

# Описание предметной области. Постановка задачи.

**Цель.**

Получить первые необходимые навыки профессионального подхода к процессу объектно-ориентированного анализа и проектирования программного обеспечения с использованием CASE-средств.

**Задание.**

Предприятие организовывает показ кино и мультфильмов населению. Большой объём информации о выполняемой работе предприятия, сведения о фильмах и предоставляемых услугах требуют своевременной обработки и предоставления отчётов для принятия решений. Для обеспечения своей деятельности оно нуждается в программной системе обработки заказов.

Клиенты приобретают билеты, направляя в компанию перечень товаров с информацией об оплате. Предприятие обрабатывает информацию и выда1т товары клиентам.

# Составление глоссария проекта.

Менеджер – человек, работающий с системой предприятия, сотрудник фирмы.

Клиент – человек или организация, желающий(ая) оформить заказ .

Товар – фильмы и услуги доступные клиенту для приобретений.

Заказ – совокупность сведений о фильме, клиенте, зале, оплате, хранящейся в БД системы и отражающая операцию по приобретению билета.

Зал – место, в котором будет осуществляться показ фильма.

# Описание дополнительных спецификаций.

Назначение дополнительных спецификаций – определить требования к системе начальной установки кодов доступа и контроля, которые не отражены в модели вариантов использования. Вместе они образуют полный набор требований к системе.

Дополнительные спецификации определяют нефункциональные требования к системе, такие, как надежность, удобство использования, производительность, сопровождаемость, а также ряд функциональных требований, являющихся общими для нескольких вариантов использования.

* **Функциональные возможности**

Система должна обеспечивать многопользовательский режим работы.

* **Удобство использования**

Пользовательский интерфейс должен быть интуитивно понятен.

* **Надежность**

Система должна быть в работоспособном состоянии 24 часа в день 7 дней в неделю, время простоя – не более 10%.

* **Производительность**

Система должна поддерживать запросы нескольких пользователей.

* **Безопасность**

Система должна разрешать доступ к базе данных пользователей, которые знают пароль.

# Создание модели вариантов использования.

Создание действующих лиц.

Действующим лицом (Actor) является любая сущность, взаимодействующая с системой извне. Им может быть человек, оборудование, другая система, т. е. с его помощью мы определяем, что взаимодействует с системой.

Действующие лица:

* Менеджер – человек, работающий с системой предприятия, занимающейся грузоперевозками, сотрудник фирмы.
* Клиент – человек или организация, желающий(ая) оформить заказ на ряд продукции по прайсу на перевозку груза.

Создание вариантов использования.

Вариант использования описывает, что система предоставляет актеру, т. е. определяет некоторый набор транзакций (групповых действий), совершаемый действующим лицом при взаимодействии с системой, при этом ничего не говориться о том, каким образом будет реализовано взаимодействие. Выделяются следующие варианты использования:

1. Авторизация.
2. Регистрация клиента.
3. Оформление заказа.
4. Оплата.
5. Отправка сведений об оплате.
6. Проведение заказа.

Реализация в Rational Rose изображена на рисунке 1.

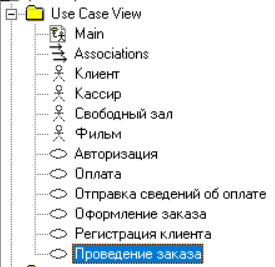


Рисунок 1 - Создание действующих лиц и вариантов использования.

**Построение диаграммы вариантов использования.**

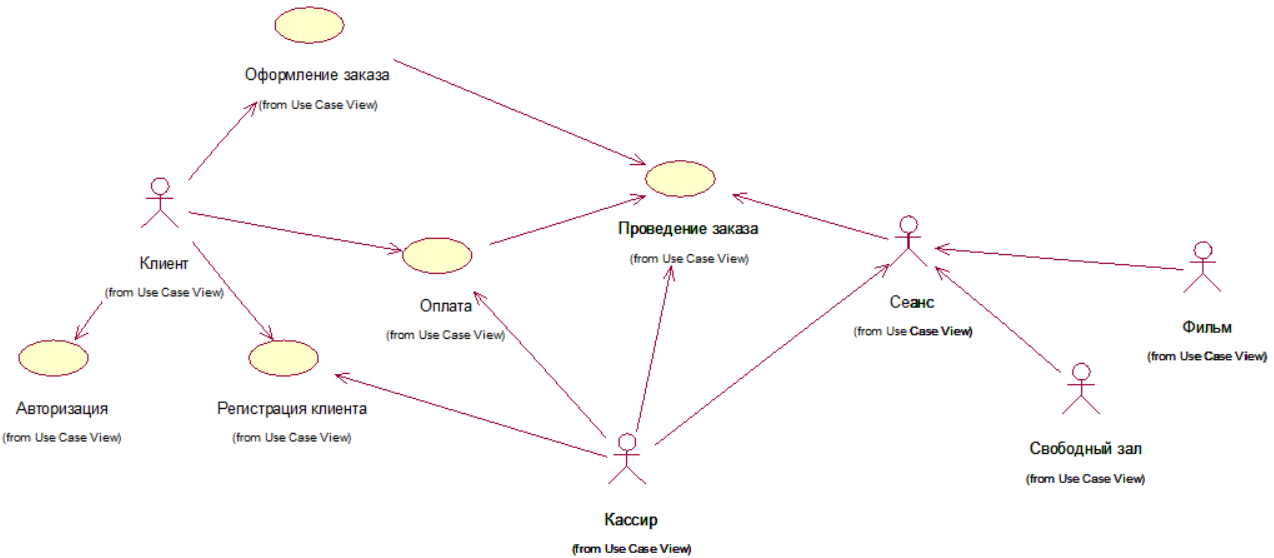


Рисунок 2 – Диаграмма вариантов использования

**Добавление описаний к вариантам использования**

|  |
| --- |
| Вариант использования: «Оформление заказа» |
| **ID:** |
| **Краткое описание:**  Вариант использования позволяет клиенту оформить заказ в соответствии с прейскурантом и сохранить сведения о нем в БД системы. |
| **Главное действующее лицо:**  Клиент |
| **Второстепенные действующие лица:**  Менеджер |
| **Предусловия:**  Перед выполнением этого варианта использования необходимо чтобы были известны исходные данные о сеансе (фильм,время) и клиенте (имя, телефон, адрес). |
| **Основной поток:**  1. Вариант использования начинается, когда клиент нажимает кнопку «Заказать билет» и это действие фиксируется в БД.  2. Клиент вводит пожелания о билете в поля формы.  3. Система автоматически присваивает идентификационный номер заказу.  4. По запросу кассира система сохраняет в БД информацию о клиенте. |
| **Постусловия:**  Если вариант использования выполнен успешно, то сведения о новом билете будут сохранены в БД.  В противном случае состояние системы не изменится. |

|  |
| --- |
| **Альтернативные потоки:**  При неправильном вводе информации система выдает предупреждающее сообщение пользователю.  В случае исправления ошибок ввода пользователем, вариант использования завершается. |

|  |
| --- |
| Вариант использования: «Регистрация клиента» |
| **ID:** |
| **Краткое описание:**  Вариант использования позволяет сотруднику предприятия зарегистрировать нового клиента и сохранить сведения о нем в БД системы или же предоставляется с помощью приложения новому пользователю самостоятельно зарегистрироваться в системе, как клиенту. |
| **Главное действующее лицо:**  Менеджер |
| **Второстепенные действующие лица:**  нет |
| **Предусловия:**  Перед выполнением этого варианта использования необходимо чтобы была известна информация о клиенте (ФИО, телефон). |
| **Основной поток:**  1. Вариант использования начинается, когда новый клиент оформляет заказ  2. Клиент вводит информацию (ФИО, номер телефона).  3. Система автоматически присваивает идентификационный номер клиенту.  4. Система по запросу пользователя осуществляет сохранение сведений о новом клиенте в БД. |
| **Постусловия:**  Если вариант использования выполнен успешно, то сведения о новом клиенте будут сохранены в БД.  В противном случае состояние системы не изменится. |
| **Альтернативные потоки:**  При неправильном вводе информации система выдает предупреждающее сообщение клиенту.  В случае исправления ошибок ввода клиентом, вариант использования завершается. |

|  |
| --- |
| Вариант использования: «Проведение заказа» |
| **ID:** |
| **Краткое описание:**  Вариант использования позволяет сотруднику предприятия оформить билет для клиента |
| **Главное действующее лицо:**  Менеджер |
| **Второстепенные действующие лица:**  нет |
| **Предусловия:**  Перед выполнением этого варианта использования необходимо чтобы была сформирована БД со сведениями о фильмах, клиентах, сеансах и была проведена оплата за полную стоимость заказа. |
| **Основной поток:**  1. Менеджер принимает пожелания клиента.  2. Менеджер выдаёт возможные варианты.  5. Заказ переводится на оформления.  6. После оплаты клиент получает свой билет. |
| **Постусловия:**  Если вариант использования выполнен успешно, то информация по запросу будет обновлена в БД.  В противном случае состояние системы не изменится. |
| **Альтернативные потоки:**  нет |

# Анализ системы.

**Архитектурный анализ выполняется архитектором системы и включает в себя:**

* утверждение общих стандартов (соглашений) моделирования и
* документирования системы;
* предварительное выявление архитектурных механизмов (механизмов
* анализа);
* формирование набора основных абстракций предметной области
* (классов анализа);
* формирование начального представления архитектурных уровней.

**Соглашения моделирования** определяют:

* используемые диаграммы и элементы модели;
* правила их применения;
* соглашения по именованию элементов модели;
* организацию модели (пакеты).

**Набора соглашений моделирования:**

* Имена вариантов использования должны быть короткими, глагольными фразами.
* Имена классов должны быть существительными, соответствующими, по возможности, понятиям предметной области.
* Имена классов должны начинаться с заглавной буквы.
* Имена атрибутов и операций должны начинаться со строчной буквы.
* Составные имена должны быть сплошными, без подчеркиваний, каждое отдельное слово должно начинаться с заглавной буквы.
* Все классы и диаграммы, описывающие предварительный системный проект, помещаются в пакет с именем AnalysisModel.
* Диаграммы классов, реализующих вариант использования, и диаграммы взаимодействия, отражающие взаимодействие объектов в процессе реализации сценариев варианта использования, помещаются в кооперацию с именем данного варианта использования и стереотипом «use-caserealization». Все кооперации помещаются в пакет с именем UseCaseRealizations. Связь между вариантом использования и его реализацией изображается на специальной диаграмме трассировки.

Идентификация основных абстракций заключается в предварительном определении набора классов системы (классов анализа) на основе описания предметной области и спецификации требований к системе (в частности, глоссария).

Так, для системы регистрации идентифицировано два класса анализа:

* Customer (Клиент)
* Session (Сеанс)

**Создание структуры модели по требованиям архитектурного анализа.**

****

Рисунок 3 - Диаграмма трассировки Traceabilities

**Создание классов анализа по требованиям архитектурного анализа**

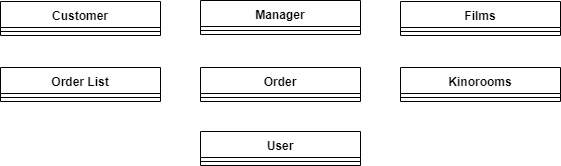
****

Рисунок 4 - Классы анализа KeyAbstractions

На рис. 4 представлены предварительно определенные классы анализа системы, т.е. идентифицированы ключевые абстракции предметной области.

**Анализ вариантов использования**

**Идентификация классов, участвующих в реализации потоков событий варианта использования.**

В потоках событий варианта использования выявляются классы трех типов:

1. Граничные классы (Boundary) – служат посредниками при взаимодействии внешних объектов с системой. Как правило, для каждой пары «действующее лицо – вариант использования» определяется один граничный класс. Типы граничных классов: пользовательский интерфейс (обмен информацией с пользователем, без деталей интерфейса - кнопок, списков, окон), системный интерфейс и аппаратный интерфейс(используемые протоколы, без деталей их реализации).

2. Классы-сущности (Entity) – представляют собой ключевые абстракции (понятия)разрабатываемой системы. Источники выявления классов-сущностей: ключевые абстракции, созданные в процессе архитектурного анализа, глоссарий, описание потоков событий для вариантов использования.

3. Управляющие классы (Control) – обеспечивают координацию поведения объектов в системе. Могут отсутствовать в некоторых вариантах использования, ограничивающихся простыми манипуляциями с хранимыми данными. Как правило, для каждого варианта использования определяется один управляющий класс.

Пример набора классов, участвующих в реализации варианта использования «Регистрация клиента», приведен на рис. 5.

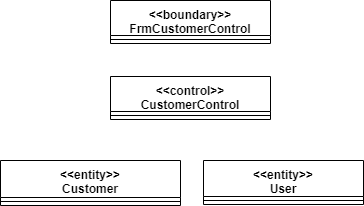


Рисунок 5 - Классы, участвующие в реализации варианта использования «Регистрация клиента»

**Распределение поведения, реализуемого вариантом использования, между классами.**

Реализуется с помощью диаграмм взаимодействия (диаграмм последовательности и диаграмм кооперации). В первую очередь строится диаграмма (одна или более), описывающая основной поток событий и его подчиненные потоки. Для каждого альтернативного потока событий строится отдельная диаграмма. Примеры:

* обработка ошибок,
* контроль времени выполнения,
* обработка неправильных вводимых данных.

На рис. 6-11 представлены диаграммы последовательности и кооперативные диаграммы, участвующие во всех ранее определенных вариантах использования.

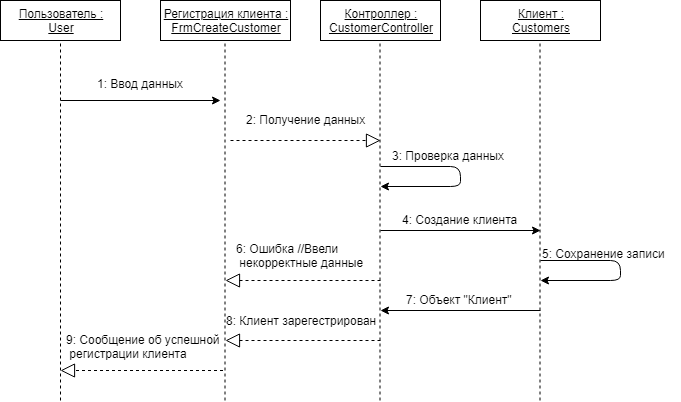


Рисунок 6 - Диаграмма последовательности «Зарегистрировать клиента".

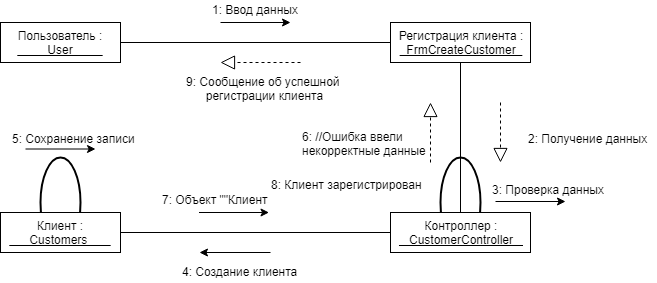


Рисунок 7 - Кооперативная диаграмма «Зарегистрировать клиента".

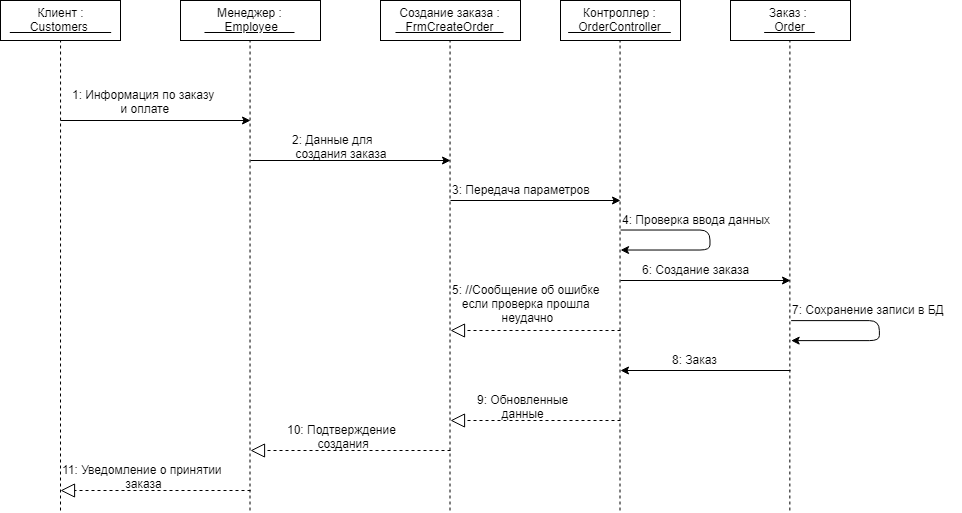


Рисунок 8 - Диаграмма последовательности «Оформление заказа".

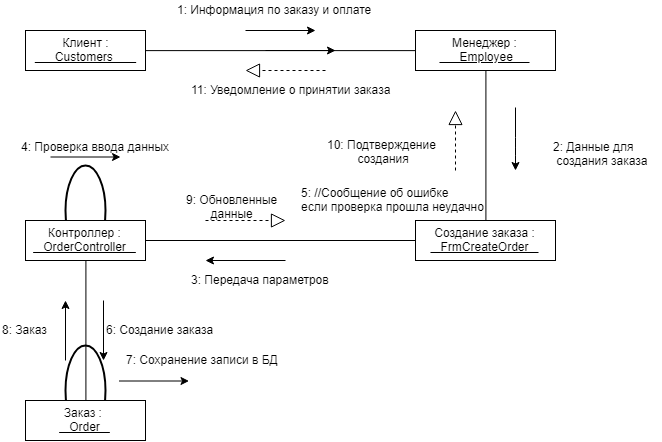


Рисунок 9 - Кооперативная диаграмма «Оформление заказа".

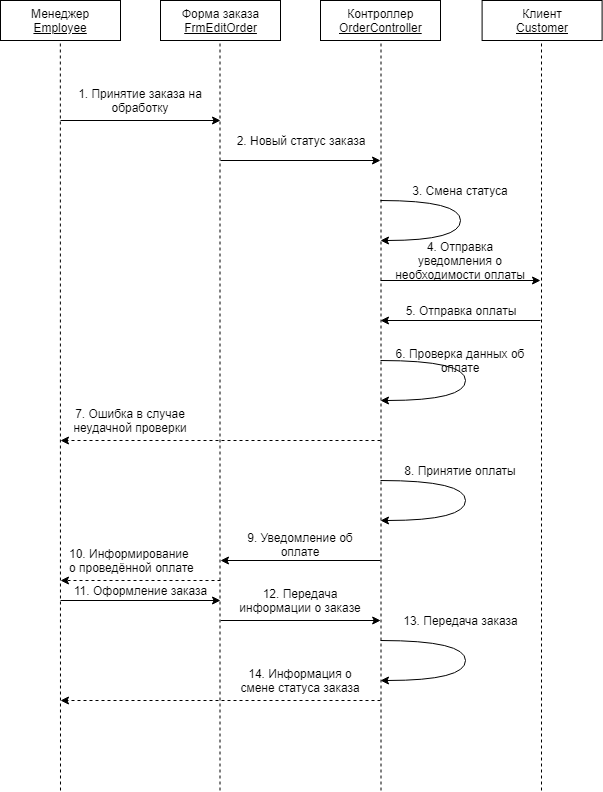


Рисунок 10 - Диаграмма последовательности «Проведение заказа".

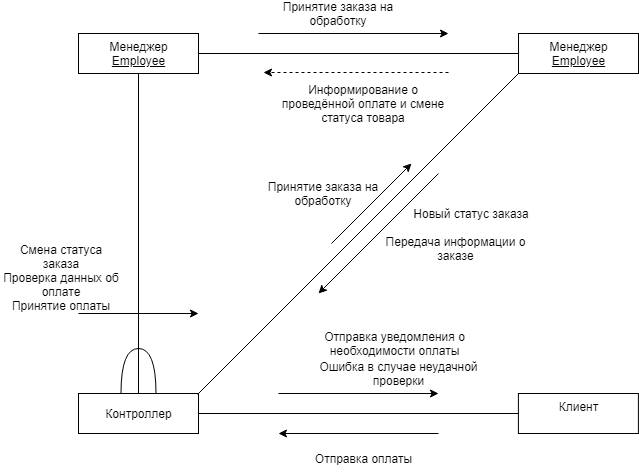


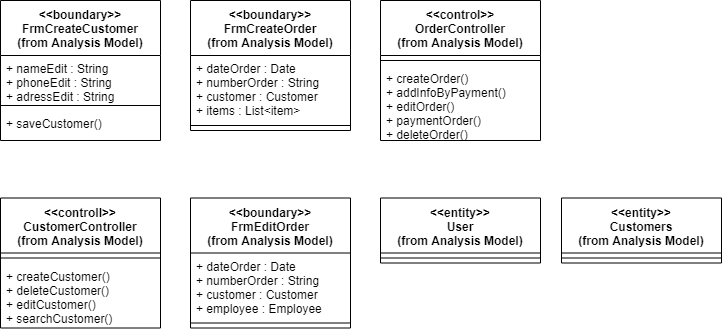
Рисунок 11 - Кооперативная диаграмма «Проведение заказа".

**Определение обязанностей, атрибутов и ассоциаций классов**

*Добавление атрибутов к классам*

Обязанность– действие, которое объект обязан выполнять по запросу других объектов. Обязанность преобразуется в одну или более операций класса на шаге проектирования. Обязанности определяются, исходя из сообщений на диаграммах взаимодействия, и документируются в классах в виде операций «анализа», которые появляются там автоматически в процессе построения диаграмм взаимодействия (соотнесения сообщений с операциями).

Атрибуты классов анализа определяются, исходя из знаний о предметной области, требований к системе и глоссария.



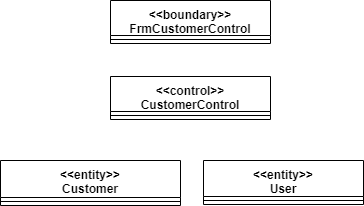


Рисунок 12 - Классы с операциями “анализа” и атрибутами

Связи (ассоциации) между классами определяются в два этапа:

1. Начальный набор связей определяется на основе анализа кооперативных диаграмм. Если два объекта взаимодействуют (обмениваются сообщениями), между ними на кооперативной диаграмме должна существовать связь (путь взаимодействия), которая преобразуется в двунаправленную ассоциацию между соответствующими классами. Если сообщения между некоторой парой объектов передаются только в одном направлении, то для соответствующей ассоциации вводится направление навигации.

2. Анализируются и уточняются ассоциации между классами- сущностями. Задаются мощности ассоциаций, могут использоваться множественные ассоциации, агрегации, обобщения и ассоциации- классы.

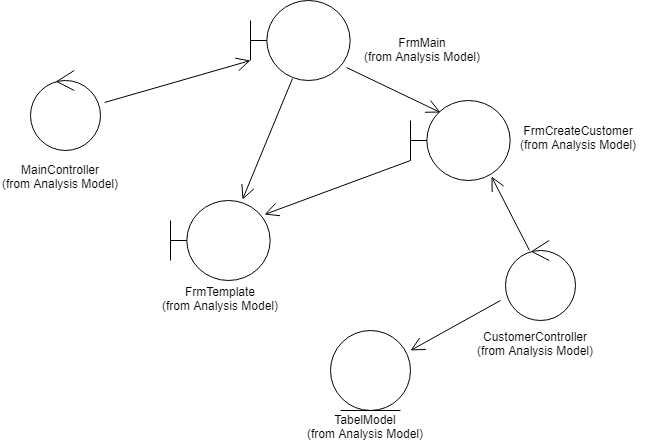


Рисунок 13 - Полная диаграмма классов для варианта использования «Регистрация клиента».

# Проектирование системы.

Целью проектирования является адаптация предварительного системного проекта (набора классов «анализа) к среде реализации с учетом всех нефункциональных требований.

Объектно-ориентированное проектирование включает два вида деятельности:

1. проектирование архитектуры системы;

2. проектирование элементов системы.

**Проектирование архитектуры системы**

Проектирование архитектуры системы выполняется архитектором системы и включает в себя:

* идентификацию архитектурных решений и механизмов, необходимыхдля проектирования системы;
* анализ взаимодействий между классами анализа, выявление подсистеми интерфейсов;
* формирование архитектурных уровней;
* проектирование структуры потоков управления;
* проектирование конфигурации системы.

**Моделирование распределенной конфигурации системы**

Распределенная конфигурация системы моделируется с помощью диаграммы размещения (Рисунок 14). Ее основные элементы:

* узел - вычислительный ресурс (процессор или другое устройство,дисковая память, контроллеры устройств и т.д.). Для узла можнозадать выполняющиеся на нем процессы;
* соединение - канал взаимодействия узлов (сеть).

Распределение процессов по узлам сети производится с учетом следующих факторов:

* используемые образцы распределения (терминал-сервер, файл-сервер,клиент-сервер, равноправные узлы и т.д.);
* время отклика;
* минимизация сетевого трафика;
* мощность узла;
* надежность оборудования и коммуникаций.

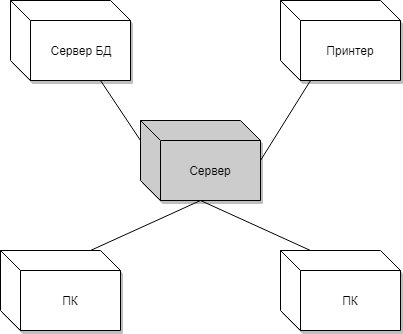
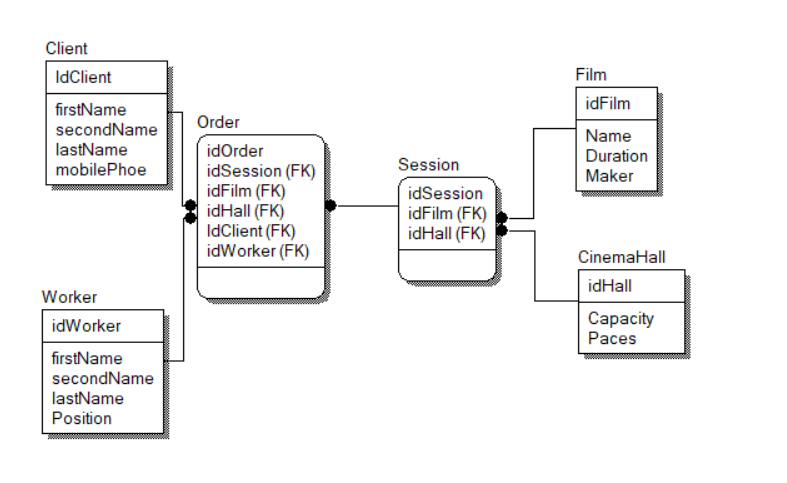


Рисунок 14 - Конфигурация системы.

**Описание классов.**

|  |  |
| --- | --- |
| **Название** | **Описание** |
| formMain | Основное окно программы. |
| MainController | Для запуска и управления приложением. |
| TableModel | Для взаимодействия программного приложения и таблицы БД. |
| FormCreateClient | Подклассы formMain, позволяют вводить данные по всем ключевым объектам системы |
| FormCreateOrder |
| FormEditOrder |

**Проектирование баз данных.**



# Реализация системы.

**Создание компонентов**

В RationalRose диаграммы компонентов создаются в представлении компонентов системы. Отдельные компоненты можно создавать непосредственно на диаграмме, или перетаскивать их туда из браузера.

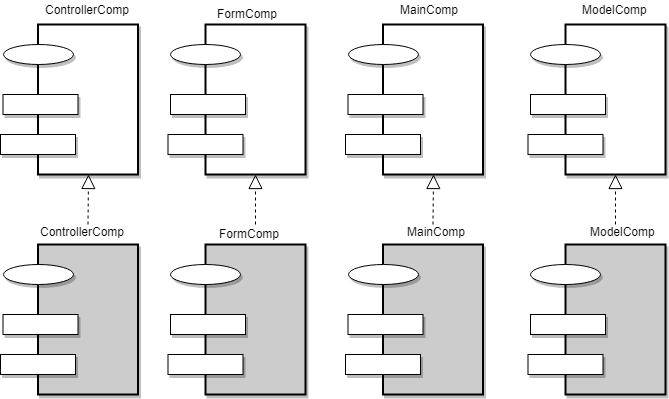


Рисунок 15 - Диаграмма компонентов системы