Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование программного обеспечения»

Лабораторная работа №6

«ООМ. Физические диаграммы UML»

Студент: Стрелковская В. А.

ФИТ 3 курс 2 группа

Преподаватель: Якубенко К. Д.

# 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

* 1. **Постановка задачи**

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

* 1. **Описание программных средств**

Для построения моделей использовалось программное средство Draw.io.

Draw.io – это инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, диаграмм связей сущностей (ERD), программных блоков, схем сетевых архитектур и других визуальных представлений данных. Сервис распространяется бесплатно, с возможностью использования в онлайн-режиме, а также предлагает десктопные приложения для различных операционных систем. Draw.io поддерживает интеграцию с популярными облачными хранилищами, такими как Google Drive, Dropbox, OneDrive, и GitHub, что делает его удобным для совместной работы и версионирования диаграмм. Пользователи могут экспортировать диаграммы в различные форматы, такие как PNG, PDF, SVG, и другие.

Особенности Draw.io:

* более 500 шаблонов элементов и фигур;
* облегчённый интерфейс, в котором за короткий промежуток времени можно создать готовый проект;
* поддержка горячих клавиш, задействованных в большинстве графических редакторов;
* экспорт в форматы: JPG, PNG, SVG, VDSX;
* возможность совместной работы;
* наличие различных фоновых тем;
* мультиязычный интерфейс.

Программное средство draw.io также поддерживает импорт файлов в различных форматах, включая .vsdx (формат Microsoft Visio), Gliffy™ и Lucidchart™. Это позволяет вам работать с существующими диаграммами, созданными в других инструментах, и продолжать их редактирование и доработку в draw.io.

1. **Практическое задание**

Диаграмма компонентов – показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код.

Результат построения представлен на рисунке 4.1.

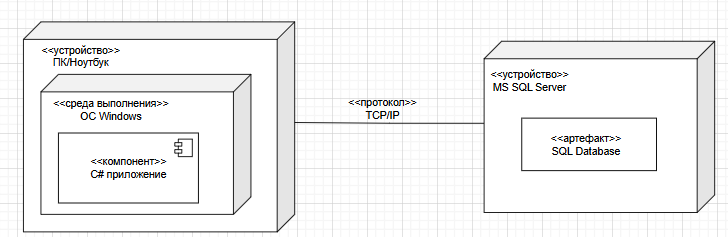
****

Рисунок 4.1 – Диаграмма развёртывания

Артефакт – некая физическая сущность, программный компонент, который используется или создаётся во время работы программного обеспечения. На данной диаграмме представлены следующие артефакты:

* "ПК/Ноутбук" – это физическое устройство, которое является артефактом, поскольку оно используется для запуска и выполнения программного обеспечения.
* "C# приложение" – это программный компонент, разработанный на языке C#, который также выступает в качестве артефакта.
* "ОС Windows" – это программная среда, в которой выполняется клиентское приложение, и также представляет собой артефакт.
* "MS SQL Server" – это физическое устройство, на котором размещается база данных и серверная часть приложения.
* "SQL Database" – это программная система управления базами данных, которая является артефактом, развернутым на сервере.
* "TCP/IP" – это сетевой протокол, используемый для связи между клиентским и серверным компонентами, и также может рассматриваться как артефакт.

Узел – то, что может содержать программное обеспечение.

Узлы бывают двух типов:

* устройство (device) – это физическое оборудование: компьютер или устройство, связанное с системой;
* среда выполнения (execution environment) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процесс-контейнер.

На основе технического задания были определены компоненты, узлы и взаимоотношения между ними.

1. Список компонентов подсистемы:
2. Компонент «ПК/Ноутбук»:

* технология реализации: общепринятые технологии для ПК/ноутбуков.

1. Компонент «C#-приложение»:

* технология реализации: WPF: фреймворк от Microsoft для разработки десктопных приложений под операционную систему Windows, использующий язык программирования C#.

1. Компонент «Система базы данных»:

* технология реализации: Microsoft SQL Server.

1. Список физических устройств:
2. Устройство «ПК/ноутбук»:

Основные характеристики: ПК или ноутбук, поддерживающий установку и выполнение приложений.

1. Устройство «Сервер базы данных»:

* основные характеристики: физический сервер, способный запускать и обслуживать базу данных Microsoft SQL Server.

1. Соединения – протоколы связи между устройствами:

Соединение между «Серверным приложением» и «Клиентским приложением»: протокол связи: TCP/IP.

1. Среды выполнения:

Среда выполнения разрабатываемого приложения предоставляет возможность запускать и выполнять код приложения. Она обеспечивает интерпретацию или компиляцию и выполнение инструкций, записанных на выбранном языке программирования.

1. Размещение компонентов на устройствах:
2. Клиентское устройство:

* компонент «среда выполнения»

1. Серверное устройство:

* компонент система базы данных Microsoft SQL Server.

Таким образом, данная архитектура обеспечивает четкое разделение ответственности между физическими устройствами и логическими компонентами системы, что способствует ее масштабируемости, надежности и гибкости. Использование стандартных технологий и протоколов связи позволяет интегрировать эту систему с другими приложениями и сервисами при необходимости.

Кроме того, выбор технологий реализации, таких как WPF и Microsoft SQL Server, демонстрирует стремление к использованию проверенных и надежных решений, что повышает вероятность успешного внедрения и эксплуатации системы.

Данные архитектурные и технологические решения повышают вероятность успешного внедрения и эксплуатации системы. Они демонстрируют стремление разработчиков к использованию проверенных, производительных и масштабируемых решений, что положительно скажется на общей эффективности и стабильности разрабатываемого приложения.

На рисунке 4.2 представлена диаграмма компонентов.

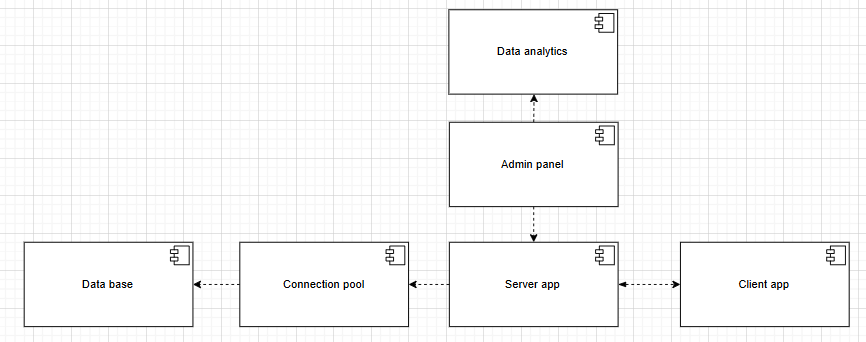


Рисунок 4.2 – Диаграмма компонентов

На данной диаграмме компонентов представлена архитектура программной системы, состоящей из нескольких ключевых элементов:

* Data base – компонент, отвечающий за хранение и управление данными в базе данных.
* Connection pool – компонент, реализующий пул соединений для эффективной работы с базой данных.
* Server app – компонент, представляющий серверную часть приложения, обрабатывающий запросы от клиентов.
* Client app – компонент, отвечающий за клиентское приложение, с которым взаимодействуют конечные пользователи.
* Admin panel – компонент, предоставляющий административный интерфейс для управления системой.
* Data analytics – компонент, отвечающий за аналитические возможности, позволяющие анализировать данные, собранные в процессе работы системы.

Эти компоненты распределены по различным узлам, что позволяет достичь масштабируемости, надежности и гибкости архитектуры. Использование пула соединений и разделение на клиентскую и серверную части обеспечивает эффективное использование ресурсов и высокую производительность системы.

Таким образом, данная диаграмма компонентов представляет собой архитектуру, ориентированную на масштабируемость, расширяемость и управляемость программной системы.

1. **Ответы на вопросы**
   * 1. Укажите назначение физических диаграмм: компонентов и развертывания.

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма развёртывания же позволяет определить распределение компонентов системы по её физическим узлам; показать физические связи между всеми узлами реализации системы на этапе ее выполнения; выявить узкие места системы и реконфигурировать её топологию для достижения требуемой производительности.

* + 1. Дайте определение понятиям: узел, артефакт, интерфейс.

Узел – то, что может содержать программное обеспечение (некоторый физически существующий элемент системы, обладающий вычислительным ресурсом).

Артефакт – некая физическая сущность, программный компонент, который используется или создаётся во время работы программного обеспечения.

Интерфейс – элемент, который служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры.

* + 1. Опишите нотации, которые используются для представления компонентов.

Для представления компонентов используются две нотации – рисунок 5.1, отличаются они визуально.

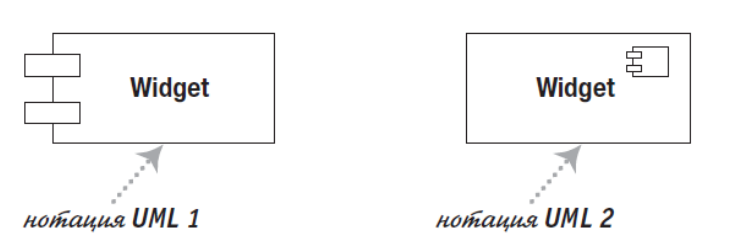


Рисунок 5.1 – Нотации для представления компонентов

* + 1. Опишите основные нотации, которые используются для представления архитектуры системы в виде диаграммы развертывания.

Узлы (Nodes): представляют физические устройства или вычислительные ресурсы, такие как серверы, ПК, мобильные устройства и т.д. Узлы изображаются в виде прямоугольников с именем устройства.

Артефакты (Artifacts): представляют программные компоненты или данные, которые размещены на узлах. Они изображаются в виде прямоугольников или эллипсов внутри узлов.

Связи (Connections): показывают взаимодействие между узлами. Они могут быть направленными или двунаправленными линиями, обозначающими поток данных, коммуникации или зависимостей между узлами.

* + 1. Укажите основные виды связей между компонентами и между узлами.

Сетевые связи (Network Links): показывают физические сетевые соединения между узлами. Это могут быть Ethernet, Wi-Fi, VPN и другие типы сетевых соединений.

Коммуникационные связи (Communication Links): представляют логические связи между компонентами системы. Это могут быть протоколы обмена данными, такие как HTTP, TCP/IP, MQTT и другие, которые описывают способы обмена информацией между компонентами.

Зависимости (Dependencies): показывают, какие компоненты или узлы зависят от других. Например, если один компонент зависит от базы данных, это может быть представлено стрелкой или связью, указывающей на эту зависимость.

Репликация и балансировка нагрузки (Replication and Load Balancing): если компоненты или данные реплицируются на несколько узлов или происходит балансировка нагрузки между узлами для повышения производительности и надежности, это может быть показано на диаграмме развертывания.

Прокси и шлюзы (Proxies and Gateways): представляют промежуточные узлы, используемые для обеспечения безопасности, переадресации или трансляции данных между компонентами.