Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование интернет систем»

Лабораторная работа №6

«Объектно-ориентированное моделирование. Физические диаграммы UML»

Студент: Фамилия И. О.

ФИТ 4 курс 6 группа

Преподаватель: Якубенко К. Д.

Минск 2024

1. **Цель лабораторной работы**

Цель лабораторной работы по теме "Объектно-ориентированное моделирование. UML-физические диаграммы" заключается в углубленном изучении методологии объектно-ориентированного моделирования (ООМ) с использованием UML (Unified Modeling Language) для создания и анализа физических диаграмм, которые отражают архитектуру и развертывание программных систем. Объектно-ориентированное моделирование представляет собой ключевой подход в проектировании программного обеспечения, направленный на описание и построение сложных систем, что позволяет создавать устойчивую, гибкую и масштабируемую архитектуру программных решений. Работа с UML-физическими диаграммами позволяет лучше понять инфраструктурные требования к системе и то, как различные компоненты взаимодействуют на уровне аппаратного и программного обеспечения.

Одной из задач лабораторной работы является ознакомление с основными принципами объектно-ориентированного проектирования, такими как инкапсуляция, наследование и полиморфизм, которые лежат в основе объектно-ориентированного подхода. Эти принципы позволяют более эффективно управлять сложностью разработки и развертывания программных систем, способствуя созданию структурированных и удобных для сопровождения архитектур. В ходе работы изучаются методы представления системы в виде взаимосвязанных компонентов, что имеет решающее значение для обеспечения совместимости и надежности при развертывании в различных средах.

В лабораторной работе особое внимание уделяется моделированию физического расположения компонентов системы и их взаимодействия в развернутой среде. Рассматривается, как программные компоненты и узлы, такие как серверы и базы данных, взаимодействуют друг с другом, обмениваются данными и поддерживают связь, что позволяет более точно отразить архитектурные требования и планировать будущие изменения. Описание взаимодействий между физическими компонентами помогает глубже понять распределение нагрузки и схемы развертывания, что способствует оптимизации производительности системы.

Еще одной целью работы является овладение навыками анализа и корректировки UML-моделей, чтобы они соответствовали реальным требованиям развертывания информационных систем. UML позволяет анализировать как логическую, так и физическую структуру системы, что помогает выявить потенциальные ошибки на этапе проектирования. В процессе работы формируется понимание того, как обнаруживать узкие места в архитектуре, проверять корректность взаимодействия компонентов и предложить возможные улучшения и оптимизации инфраструктуры.

Кроме того, выполнение лабораторной работы направлено на развитие практических навыков использования UML как универсального инструмента для документирования и анализа информационных систем.

1. **Описание функциональных требований**

Функциональные требования к проекту «Платформа для обучения и контроля знаний по математике» можно разделить по ролям: администратор, репетитор и ученик.

Каждая роль имеет свои специфические задачи и доступ к различным функциям системы.

Функционально web-приложение должно:

* обеспечивать возможность авторизации и аутентификации;
* поддерживать роли администратора, репетитора и ученика;
* предоставлять администратору возможность управления пользователями;
* предоставлять администратору возможность управления контентом, включая учебные материалы, курсы, тесты;
* предоставлять репетитору возможность регистрации учеников;
* предоставлять репетитору возможность изменения информации об учениках;
* предоставлять возможность интеграции с Google календарем;
* предоставлять репетитору возможность управления учебными курсами и тестами;
* предоставлять репетитору возможность создания интерактивных математических моделей, включая 2D и 3D модели;
* предоставлять репетитору возможность отслеживания результатов учеников;
* предоставлять возможность ученикам просматривать учебные материалы;
* предоставлять возможность ученикам проходить учебные тесты;
* предоставлять возможность ученикам отслеживать прогресс в личном кабинете.

Функциональные требования платформы для обучения и контроля знаний по математике отражают ключевые аспекты работы системы, направленные на удовлетворение потребностей всех пользователей.

Разделение на роли – преподаватель, ученик и администратор – позволяет гибко управлять процессом обучения, предоставляя каждому пользователю необходимый набор инструментов для выполнения их задач.

Преподаватели получают мощные средства для создания курсов и тестов, ученики могут взаимодействовать с материалами и отслеживать свой прогресс, а администратор обеспечивает контроль за системой и её поддержкой.

Такой подход обеспечивает целостность и функциональность платформы, делая её удобной и эффективной в использовании.

1. **Описание программных средств**

Draw.io (или diagrams.net) — это мощный и удобный инструмент для создания разнообразных диаграмм и схем, который завоевал популярность благодаря своей функциональности и бесплатной основе. Одной из главных особенностей платформы является простота в использовании, что делает её доступной как для новичков, так и для профессионалов. Благодаря интуитивно понятному интерфейсу, пользователи могут быстро создавать различные типы диаграмм, такие как блок-схемы, диаграммы UML, сетевые диаграммы, карты процессов, модели IDEF0 и организационные структуры.

Инструмент также предлагает обширную библиотеку готовых шаблонов и элементов для создания визуализаций. Пользователи могут выбирать из множества стандартных фигур и иконок, что значительно ускоряет процесс построения диаграмм.

Платформа поддерживает множество форматов для экспорта, включая PNG, JPG, SVG и PDF, что делает ее универсальной для различных задач, таких как представление проектов, печать или размещение в электронных документах. Более того, для будущего редактирования диаграммы можно сохранить в собственном формате XML, что позволяет пользователям возвращаться к проектам в любое время для их доработки. Draw.io также интегрируется с популярными облачными сервисами, такими как Google Drive, Dropbox и OneDrive, что упрощает процесс хранения, редактирования и обмена диаграммами, а также обеспечивает доступ к проектам с любого устройства и совместную работу в реальном времени.

Для командной работы draw.io предлагает инструменты для совместного редактирования диаграмм, позволяя нескольким пользователям одновременно вносить изменения в схемы и делиться комментариями. Дополнительным плюсом является возможность работы в оффлайн-режиме через настольное приложение, что позволяет создавать и редактировать схемы даже без доступа к интернету.

Платформа также предлагает интеграцию с различными корпоративными системами и популярными платформами для управления проектами, такими как Atlassian Confluence и Jira. Это позволяет использовать draw.io для визуализации данных и процессов в рамках более широких решений для управления проектами. Для пользователей, которые уже работают с другими инструментами для диаграмм, draw.io предлагает импорт и экспорт данных, что делает его совместимым с такими сервисами, как Lucidchart и Microsoft Visio, позволяя легко переходить между различными инструментами.

Таким образом, draw.io сочетает в себе простоту использования, гибкость работы с различными форматами и возможность интеграции с облачными сервисами, что делает его одним из самых популярных инструментов для создания диаграмм и визуализации данных. Платформа является идеальным решением как для индивидуальных пользователей, так и для команд, работающих над совместными проектами, предлагая все необходимые инструменты для эффективной работы.

1. **Описание практического задания**

На основании требований разработана диаграмма развертывания и компонентов, представляющая архитектуру системы, её физическое развертывание и взаимодействие между основными компонентами.

На рисунке 4.1 представлена диаграмма развертывания.

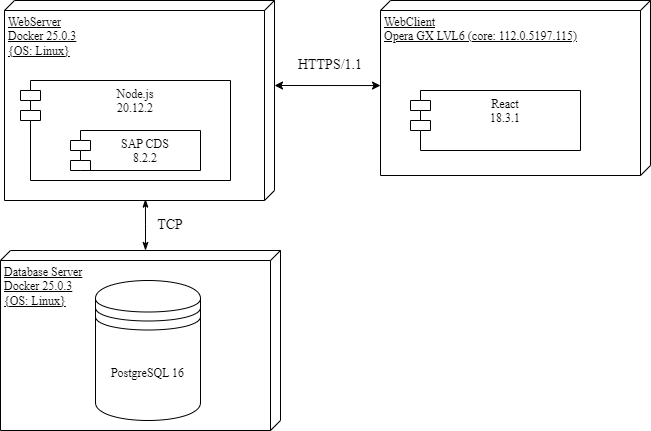


Рисунок 4.1 – Диаграмма развертывания

Описание компонентов на диаграмме:

* WebServer: контейнеризация сервера с помощью Docker с версией 25.0.3 на базе ОС Linux, обеспечивающая гибкость в развертывании. Среда выполнения для JavaScript Node.js 20.12.2, обеспечивающая работу серверного кода. Компонент, реализующий модель данных и логику приложения на стороне сервера – SAP CDS 8.2.2;
* Database Server: контейнеризация базы данных с той же версией Docker и ОС Linux. Сервер базы данных PostgreSQL 16 для хранения и управления данными, доступ к которому осуществляется по протоколу TCP;
* WebClient: веб-браузер Opera GX LVL6 (core: 112.0.5197.115), обеспечивающий интерфейс для взаимодействия пользователя с системой через HTTP/1.1. JavaScript-библиотека React 18.3.1 для создания пользовательского интерфейса, работающая на стороне клиента.

Компоненты взаимодействуют между собой через протоколы HTTPS/1.1 (между клиентом и сервером) и TCP (между сервером базы данных и сервером приложений).

1. **Ответы на вопросы**

1. Укажите назначение физических диаграмм: компонентов и развертывания.

Диаграмма компонентов служит для отображения структуры системы на уровне компонентов, показывая, как программные модули и файлы взаимодействуют между собой. Она позволяет анализировать зависимости между частями системы и планировать их изменения. Диаграмма развертывания описывает физическое развертывание системы на оборудовании, показывая, как программные компоненты размещаются на аппаратных узлах (например, серверах). Эти диаграммы полезны для анализа производительности, совместимости и масштабируемости системы.

2. Дайте определение понятиям: узел, артефакт, интерфейс.

Узел – это физический элемент, представляющий аппаратное или программное окружение, на котором развернуты компоненты, например сервер, рабочая станция или виртуальная машина. Артефакт – это конкретный результат разработки, программный модуль или файл, который может быть размещен на узле. Интерфейс – это точка взаимодействия, определяющая способы общения между компонентами и другими элементами системы, отражающая публичные методы, доступные для использования другими частями системы.

3. Опишите нотации, которые используются для представления компонентов (их

вариации).

Компоненты обычно изображаются в виде прямоугольников с иконкой "вкладки" в верхней части, что указывает на их идентичность. Вариации включают изображение компонента с ассоциированными интерфейсами и зависимостями, что может показывать, какие компоненты требуют или предоставляют определенные услуги, или связанный артефакт, для которого компонент является реализацией.

4. Опишите основные нотации, которые используются для представления

архитектуры системы в виде диаграммы развертывания.

Узлы на диаграммах развертывания изображаются в виде объемных прямоугольников, символизирующих физические или виртуальные машины. Компоненты, размещенные на узле, могут быть представлены как вложенные прямоугольники или обозначены линиями связи. Между узлами могут быть показаны связи, отражающие сетевые или другие физические подключения. Компоненты внутри узлов могут также иметь интерфейсы, показывающие взаимодействия между ними.

5. Укажите основные виды связей между компонентами и между узлами.

Связи между компонентами включают зависимости (когда один компонент зависит от другого для своей работы) и интерфейсы (точки доступа для других компонентов). Между узлами используются ассоциации, обозначающие физические соединения, например сетевые подключения, а также связи размещения, которые показывают, какие компоненты развернуты на каких узлах.