Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование интернет систем»

Лабораторная работа №5

«Объектно-ориентированное моделирование. Структурные диаграммы UML»

Студент: Фамилия И. О.

ФИТ 4 курс 6 группа

Преподаватель: Якубенко К. Д.

Минск 2024

1. **Цель лабораторной работы**

Целью данной лабораторной работы является изучение основ объектно-ориентированного моделирования с использованием структурных диаграмм UML. Работа предполагает ознакомление с основными принципами и методами моделирования, которые применяются для создания структурных представлений системы, описания её архитектуры и взаимосвязей между различными компонентами. Структурные диаграммы UML служат для отображения статических аспектов системы, включая её классы, компоненты, пакеты и объекты, что позволяет лучше понять структуру программного решения, а также обеспечить его гибкость, модульность и масштабируемость. Использование UML для структурного моделирования упрощает процесс проектирования сложных систем, улучшает документирование и облегчает дальнейшее сопровождение проекта.

В рамках лабораторной работы рассматриваются различные виды структурных диаграмм UML, их особенности, назначение и области применения.

Диаграмма классов отображает структуру системы в терминах классов и их взаимосвязей. Она представляет основные компоненты системы, их атрибуты, методы и отношения, такие как ассоциации, зависимости, наследование и агрегация.

Диаграмма компонентов показывает физическую структуру системы, составляя её из программных модулей или библиотек, и полезна для понимания того, как отдельные модули взаимодействуют друг с другом.

Диаграмма пакетов организует классы и компоненты в логические группы (пакеты), упрощая структуру больших систем и предоставляя обзор на высоком уровне.

Работа предполагает использование CASE-средств (Computer-Aided Software Engineering), таких как Rational Rose, Visual Paradigm или Enterprise Architect, что способствует развитию навыков построения и редактирования диаграмм, анализа структуры системы и документирования. CASE-средства автоматизируют процесс создания UML-диаграмм, что помогает освоить построение диаграмм, их редактирование, анализ структуры системы и документирование. Лабораторная работа также развивает умение анализировать и структурировать архитектуру системы, выявляя её основные компоненты, их связи и взаимозависимости, что помогает создавать подробные диаграммы, служащие документированием и описанием системы, облегчая её дальнейшее сопровождение и развитие. Одновременно происходит закрепление знаний об основных принципах объектно-ориентированного подхода (инкапсуляция, наследование и полиморфизм) и изучение их использования при моделировании. Это помогает выстраивать логически связанную структуру системы и упрощает её реализацию и поддержку.

В результате работы достигается понимание назначения и области применения каждого типа диаграмм, а также навыки использования CASE-средств для проектирования и документирования программных систем, что в дальнейшем позволяет анализировать и улучшать архитектуру системы на этапе её моделирования.

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**

В рамках практического задания требуется построить диаграмму классов, которая описывает типы объектов системы и различные виды статических связей между ними.

Связи включают зависимость, возникающую, если изменения одного класса требуют изменения другого; ассоциацию, когда один класс использует функционал другого; обобщение, которое подразумевает наследование всех доступных полей, свойств и методов; агрегацию, когда один класс включается в другой в качестве составной части, но при этом может существовать независимо; и композицию, когда составной класс является неотъемлемой частью основного и не может существовать отдельно от него.

Диаграмма классов для сервиса «Платформа для обучения и контроля знаний по математике» представлена на рисунке 4.1

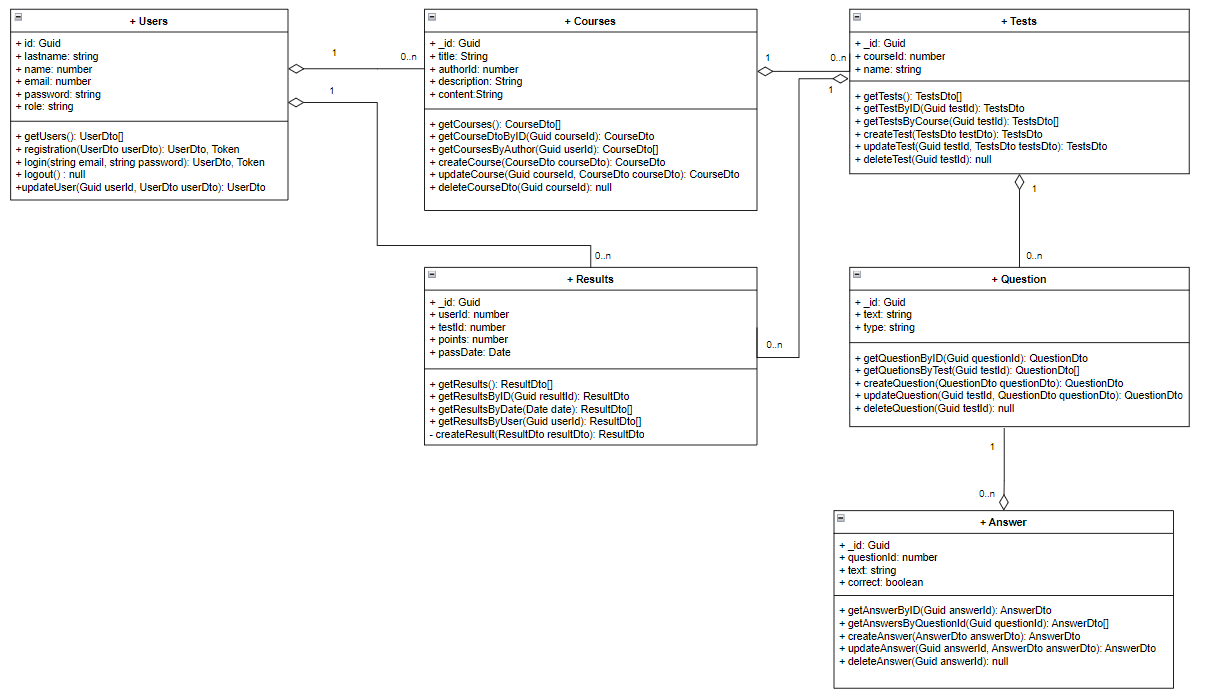


Рисунок 4.1 – Диаграмма классов

Для диаграммы классов приложения можно выделить основные классы и их методы, а также связи между ними.

Класс: Users (Пользователи):

* Атрибуты: id, lastname, name, email, phone, password, role.
* Операции: getUsers(), registerUser(), login(), updateUser().
* Связи: Связан с классом Courses агрегационной связью, так как один пользователь может создать 0 или более курсов. Также связан с классом Result агрегационной связью, так как один пользователь может иметь 0 или более результатов.

Класс: Courses (Курсы):

* Атрибуты: id, authorId, title, description, content.
* Операции: getCourses (), getCourseByID (), getCoursesByAuthor (), createCourse (), updateCourse(), deleteCourse().
* Связи: Связан с классом Tests агрегационной связью, так как один курс может содержать 0 или несколько тестов.

Класс: Results (Результаты):

* Атрибуты: id, userId, testId, points, passDate.
* Операции: getResults(), getResultsByID(), getResultsByDate(), getResultsByUser (), createResult().
* Связи: Связан с классами Users и Test композицией, так как результат не может существовать пользователя и теста.

Класс: Tests (Тесты):

* Атрибуты: id, courseId, name.

Операции: getTests(), getTestByID(), getTestsByCourse(), createTest(), updateTest().

* Связи: Связан с классами Courses, Questions и Results агрегационными связями, так как тесты зависят от пользователя, ответов и результатов.

Класс: Questions (Вопросы):

* Атрибуты: id, text, type.
* Операции: getQuestionByID(), getQuetionsByTest(), createQuestion(), updateQuestion(), deleteQuestion().
* Связи: Связан с классами Answers и Test агрегационными связями, так как вопросы определяются тестом и используются в ответах.

Класс: Answers (Ответы):

* Атрибуты: id, questionId, text, correct.
* Операции: getAnswerByID(), getAnswersByQuestionId(), createAnswer(), updateAnswer(), deleteAnswer ().
* Связи: Связан с классом Question связью композиции, так как ответы предоставляются в рамках конкретного вопроса.

В данном проекте используется многослойная архитектура, которая структурирует систему на несколько независимых слоев, каждый из которых выполняет конкретные задачи и функции. Такой подход способствует более организованному и гибкому проектированию, упрощает развитие и поддержку приложения, а также обеспечивает четкое разделение ответственности между слоями. Это разделение позволяет каждому слою быть изолированным от изменений в других частях системы, что упрощает тестирование, внесение изменений и добавление новых функций.

Диаграмма пакетов в UML наглядно отображает организацию многослойной архитектуры. Она помогает визуально структурировать приложение, разделив его на логические группы или пакеты, которые содержат связанные классы и компоненты. Каждый пакет представляет собой группу классов, связанных по смыслу и функционалу.

Диаграмма пакетов с описанием используемых слоев представлена на рисунке 4.2.

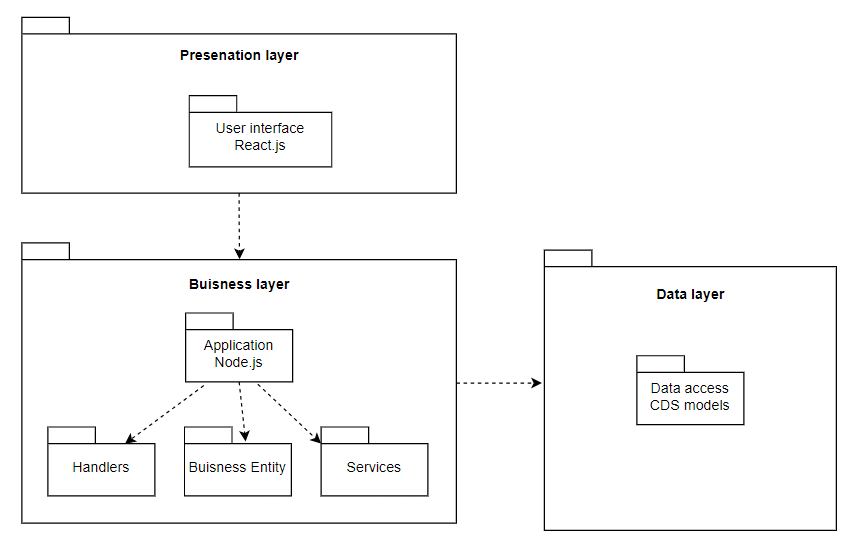


Рисунок 4.2 – Диаграмма пакетов

Диаграмма пакетов для платформы представляет систему с многослойной архитектурой, где каждый слой отвечает за выполнение определённых задач.

Presentation Layer (Слой представления). User Interface (Пользовательский интерфейс) реализован с использованием React.js. Он отвечает за взаимодействие с пользователями, отображение данных и передачу пользовательских действий в бизнес-логику.

Business Layer (Бизнес-логика). Пакет Handlers (Обработчики) отвечает за обработку запросов от слоя представления. Обработчики принимают запросы, вызывают необходимые сервисы и возвращают данные для отображения. Business Entity (Бизнес-сущности) описывает основные объекты системы, такие как пользователи, отели, номера и бронирования. Эти сущности играют центральную роль в логике работы приложения.

В пакете Services (Сервисы) находятся классы, которые реализуют основную бизнес-логику, такую как создание тестов, управление пользователями и курсами. Сервисы взаимодействуют с обработчиками и обращаются к DTO слою для передачи данных между уровнями.

Data Layer (Слой данных) содержит Пакет Data Access (Доступ к данным) использует SAP CDS для взаимодействия с базой данных. Этот пакет отвечает за все операции с базой данных, такие как чтение, запись, обновление и удаление данных. Здесь реализована вся логика для работы с коллекциями базы данных PostgreSQL.

1. **Ответы на вопросы**

1. Перечислите структурные диаграммы, которые входят в UML 2.0.

В UML 2.0 существует несколько типов структурных диаграмм, каждая из которых описывает статические аспекты системы, такие как классы, объекты, компоненты и их взаимосвязи. Диаграмма классов (Class Diagram) используется для отображения классов и их атрибутов, методов, а также связей между ними, что помогает понять основные структуры системы и связи между объектами. Диаграмма пакетов (Package Diagram) группирует классы и компоненты в пакеты, позволяя увидеть общие зависимости между различными частями системы.

2. Укажите назначение структурных диаграмм.

Структурные диаграммы UML предназначены для отображения статических аспектов системы, что помогает визуально представить её архитектуру и структуру. Основное назначение структурных диаграмм заключается в описании объектов, их типов и взаимосвязей, существующих между ними. Это позволяет разработчикам, архитекторам и аналитикам лучше понять устройство системы, разрабатывать её более гибко и масштабируемо, а также облегчить последующее сопровождение и модернизацию.

3. Опишите нотации, которые используются для построения Classes диаграмм.

В UML-диаграммах классов используется несколько нотаций для отображения структуры системы: классы изображаются как прямоугольники с секциями для атрибутов и методов, а связи между ними включают ассоциации, зависимости, обобщения, реализации, агрегации и композиции. Эти нотации показывают постоянные или временные связи, наследование, а также степень зависимости между классами, что помогает визуально описать архитектуру системы.

4. Для чего применяются расширения диаграмм UML?

Расширения диаграмм UML применяются для добавления специфических деталей и адаптации стандартных UML-диаграмм под требования конкретного проекта. Они позволяют уточнить поведение элементов и связей, задать дополнительные свойства и визуально подчеркнуть особенности системы, которые важны для её понимания и разработки.

5. Что означают понятия «стереотип» и «тегированное значение» в контексте

расширенных диаграмм?

Стереотип – это специальная метка, которая описывает назначение элемента или его роль, добавляя контекст к стандартному UML-элементу. Стереотипы позволяют расширять модель, сохраняя её читаемость.

Тегированное значение – это пара «ключ-значение», которая задаёт дополнительные атрибуты элементам. Тегированные значения уточняют свойства объектов или связей, добавляя к ним специфические данные, которые не включены в стандартную спецификацию UML.