МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование информационных систем**

**Лабораторная работа №5**

**Тема:** «Объектно-ориентированное моделирование. Структурные UML-диаграммы»

**Цель:** 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.

Выполнил:

студент 4 курса 4 группы

Белашков Василий Александрович

Минск 2023

**Теоретические вопросы:**

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

Новая версия UML 2.0, разработанная ассоциацией OMG в 2005 г., отличается от UML 1.0 большей степенью формализации в связи с необходимостью автоматизации перевода описания на UML в машинный код.

Все расширения языка основаны на получении новых модулей путем вложения базовых модулей более низкого уровня в создаваемые модули более высоких уровней.

В UML 2.0 введено 13 типов диаграмм, разделенных на два подмножества — структурных и поведенческих диаграмм.

Структурные модели, известные также как статические модели, описывают структуру сущностей или компонентов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения. К структурным относятся следующие диаграммы:

* диаграммы пакетов или контейнеров (Package diagrams);
* диаграммы классов (Class or Structural diagrams);
* диаграммы объектов (Object diagrams);
* композитные диаграммы (Composite Structure);
* диаграммы компонентов (Component diagrams), показывающие образование структур из классов и интерфейсы между структурами;
* диаграммы развертывания (Deployment diagrams).

1. **Укажите назначение структурных диаграммы.**

Основное назначение структурных диаграмм заключается в графическом представлении состава статистических совокупностей, характеризующихся как соотношение различных частей каждой из совокупностей. Состав статистической совокупности графически может быть представлен с помощью как абсолютных, так и относительных показателей.

1. **Опишите нотации, которые используются для построения Classes диаграмм.**

Существует несколько обозначений диаграмм классов, которые используются при рисовании диаграмм классов UML. Мы перечислили ниже наиболее распространенные нотации диаграммы классов.

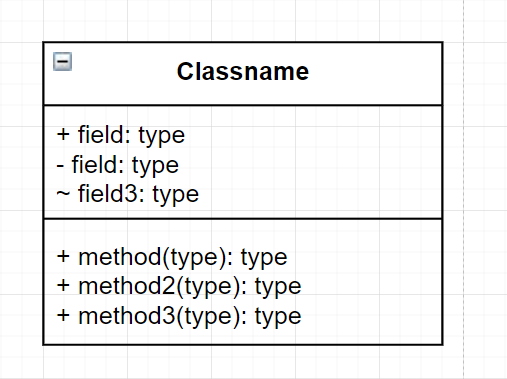


Рисунок 1 – Изображение класса

Классы представляют собой центральные объекты в системе. Он представлен прямоугольником с 3 отсеками.

Первый показывает имя класса, а средний – атрибуты класса, которые являются характеристиками объектов. В нижнем списке перечислены операции класса, которые представляют собой поведение класса.

Последние два отсека являются необязательными. Нотация класса без последних двух отделений называется простым классом и содержит только имя класса.

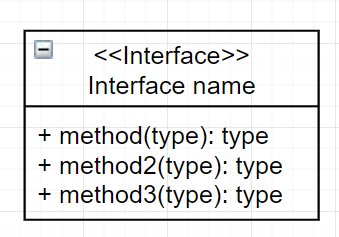


Рисунок 2 – Изображение интерфейса

Символ интерфейса на диаграммах классов обозначает набор операций, которые детализируют ответственность класса.

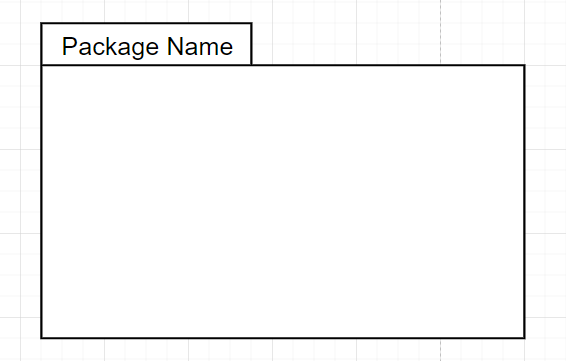


Рисунок 3 – Изображение пакета

Символ пакета используется для группировки классов или интерфейсов, которые либо похожи по своей природе, либо связаны. Группировка этих элементов дизайна с использованием символов упаковки улучшает читабельность диаграммы

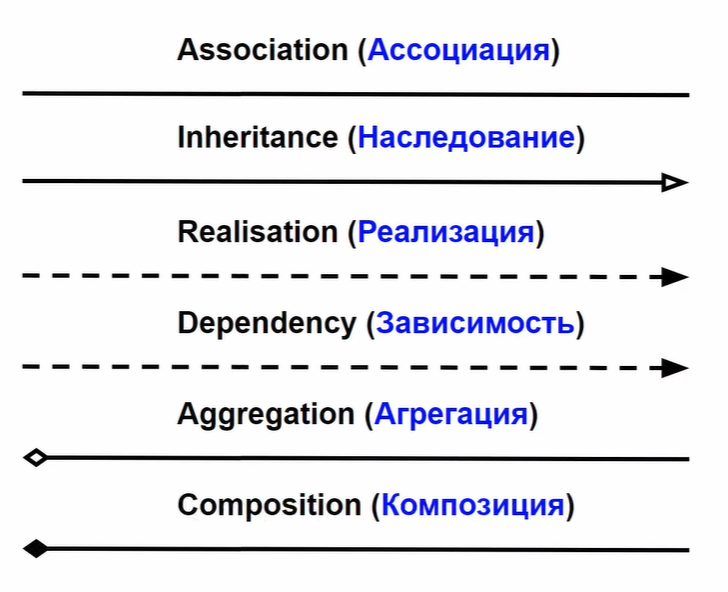


Рисунок 4 – Отношения в диаграмме классов

1. **Для чего применяются расширения диаграмм UML?**

Механизмы расширения UML включают: стереотипы (stereotype) - расширяют словарь UML, позволяя на основе существующих элементов языка создавать новые, ориентированные для решения конкретной проблемы; помеченные значения (tagged value) - расширяют свойства основных конструкций UML, позволяя включать дополнительную информацию в спецификацию элемента.

1. **Что означают понятия «стереотип»**

Стереотип- класс, связанный с "метамоделью", что означает стандартные элементы **UML**, определенные в стандарте **UML**. Таким образом, это класс классов (или класс ассоциаций, или любого другого элемента **UML**).

# Постановка задачи:

Задачей данной работы выступает описание структуры взаимодействия классов в web-приложении, описанной в лабораторной работе №1.

Главными сущностями приложения выступают:

* AppContext – класс с помощью которого осуществляется доступ к базе данных данного web-приложения.
* user.controller.ts – осуществляет обработку запросов, предназначенных пользователю. А именно: регистрация, авторизация, выход из аккаунта.
* dictionary.controller.ts – осуществляет обработку запросов, предназначенных словарем. А именно: получение всех словарей, добавление, обновление, удаление словарей.
* word.controller.ts – осуществляет обработку запросов, связанных со словами. А именно: создание слова, изменение слова удаление слова из словаря.

**Описание программных средств:**

Построение моделей выполнялось в программной среде Draw.io.

Draw.io — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

Адрес ресурса: *https://app.diagrams.net/.*

# Описание практического задания:

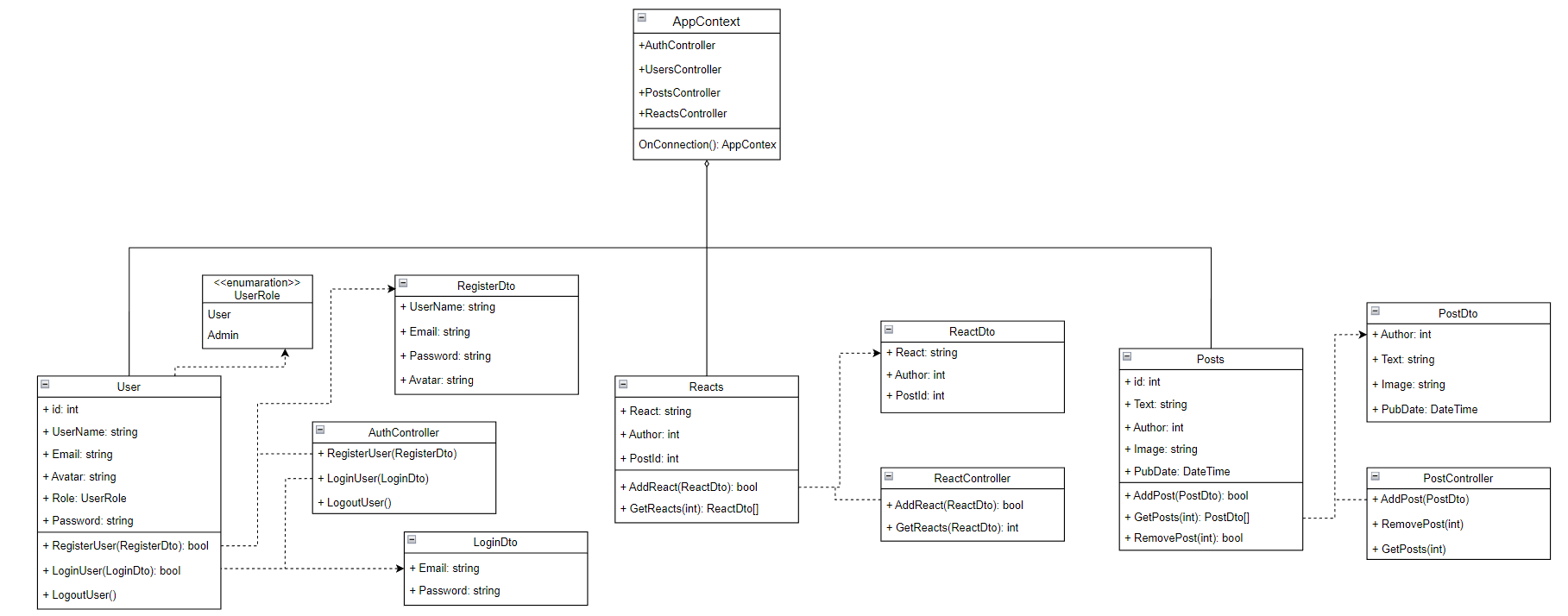


Рисунок 5 ­– Диаграмма классов

Класс AppContext реализует связь под названием “агрегация”, переменные, хранящиеся в данном классе описаны ниже:

* dbContext – приватная переменная, которая содержит экземпляр класса AppDbContext, является следствием реализации паттерна Singleton;
* Users – публичная переменная, содержащая доступ к таблице users базы данных;
* Posts– публичная переменная, содержащая доступ к таблице posts базы данных;
* Reacts – публичная переменная, содержащая доступ к таблице reacts базы данных;

Также этот класс имеет следующие методы:

* + OnConnection – защищенный метод, который выполняется при первом получении доступа к бд.

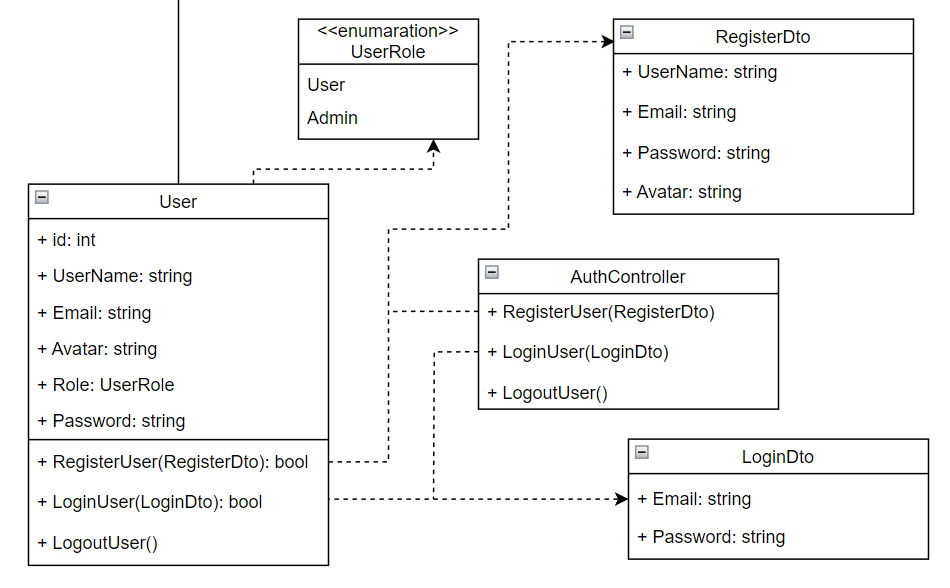


Рисунок 6 – Описание модуля «User»

**UserController** – осуществляет обработку запросов, предназначенных пользователю. А именно: регистрация, авторизация, выход из аккаунта, блокировка, разблокировка пользователя. Он содержит только публичные методы:

* + LoginUser – обработка запроса на авторизацию;
  + RegisterUser – обработка запроса на регистрацию;
  + LogoutUser – обработка запроса на выход из аккаунта;

**Users** – класс, содержащий свойства и методы для обработки запросов, которые приходят из контроллера UsersController. Далее перечислены все публичные свойства:

* + Id – id пользователя;
  + Username – ник пользователя;
  + Email – почта пользователя;
  + Password – пароль пользователя;
  + Avatar – фото пользователя;
  + Role – роль пользователя, на основе которой определяются допустимые права пользователя.

Далее указаны публичные методы класса **Users**:

* RegisterUser – выполняет добавление данных пользователя в бд;
* LoginUser – выполняет проверку, зарегистрирован для данный пользователь;
* Logout – выполняет выход из аккаунта;

**LoginDto** – класс, который используется для получение данных, необходимых при авторизации. Содержит только публичные свойства:

* + Email – почта пользователя;
* Password – пароль пользователя.

**RegisterDto** – класс, который используется для получение данных, необходимых при регистрации. Содержит только публичные свойства:

* Username – ник пользователя;
* Email – почта пользователя;
* Avatar – фото пользователя;
* Password – пароль пользователя.

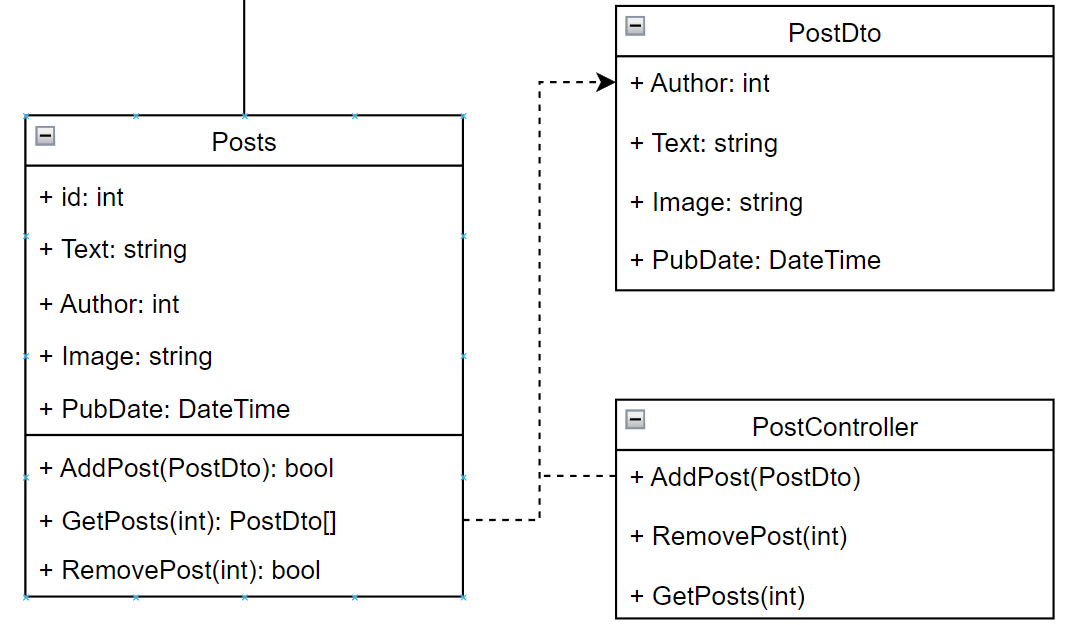


Рисунок 7 - Описание модуля «Posts»

**PostController** осуществляет обработку запросов, предназначенных посту. А именно: получение всех постов, поиск поста по id, добавление и удаление. Содержит только публичные методы:

* + GetPosts – обработка запроса на получение указанного количества постов;
  + RemovePost – обработка запроса на удаление одного поста с указанным id;
  + AddPost – обработка запроса на добавление поста в бд;

**Posts** – класс, содержащий свойства и метод для обработки запросов, которые приходят из контроллера PostController. Далее перечислены все публичные свойства:

* + Id – id поста;
  + Author– id пользователя, которому принадлежит данный пост;
  + Text – текст поста;
  + PubDate – дата и время публикации поста в web-приложении.

Далее указаны публичные методы класса **Posts**:

* AddPost – выполняет добавление поста в бд;
* RemovePost – осуществляет удаление поста из бд;
* CheckEmpty – выполняет проверку заполненности всех полей при отправке запроса;
* FindPostById – осуществляет поиска поста по указанному id;
* GetPosts – выполняет получение указанного количества постов из бд.

**PostDto** – класс, который используется для получения данных, необходимых при добавлении поста в бд. Содержит только публичные свойства:

* Author – id пользователя, который публикует данную запись;
* Text – текст поста;
* Image – картинка, прикреплённая к посту;
* PubDate – дата и время публикации поста.

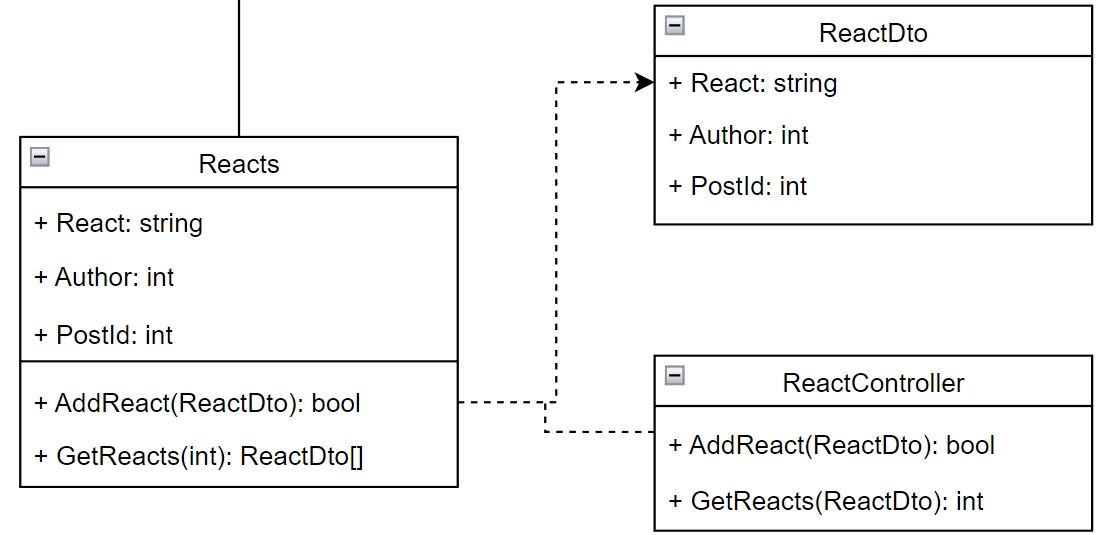


Рисунок 8 – Описание модуля «Reacts»

**ReactController** - осуществляет обработку запросов, предназначенных для добавления эмодзи к постам. Содержит только публичные методы:

* + AddReact – обработка запроса на добавление эмодзи;
  + GetReacts – получение всех реакций к заданному посту;

**Reacts** – класс, содержащий свойства и метод для обработки запросов, которые приходят из контроллера **ReactController**. Далее перечислены все публичные свойства:

* React – список доступных эмодзи;
* Author – id пользователя, которому принадлежит пост;
* PostId – id поста, к которому прикреплены реакции.

Далее указаны все публичные методы данного класса:

* + AddReact – обработка запроса на добавление эмодзи;
  + GetReacts – получение всех реакций к заданному посту;

**ReactDto** – класс, который используется для получения данных, необходимых при добавлении реакций в бд. Содержит только публичные свойства:

* React – доступные реакции;
* Author – id пользователя, который оставляет реакцию;
* PostId – id поста, к которому прикреплён эмодзи.

Далее представлены пакеты в которые были сгруппированы классы.

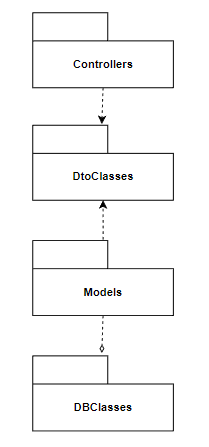


Рисунок 9 – Пакеты классов

Controllers содержит классы, отвечающие за обработку запросов на сервер. К ним относятся: AuthController, PostController, ReactController.

DtoClasses содержит классы, отвечающие за передаваемые в контроллерах данные. К этим классам относятся: LoginDto, RegisterDto, PostDto, ReactDto.

Models содержит классы, отвечающие за взаимодействие с базой данных. К этим классам относятся: Users, Posts, Reacts.

DBClasses содержит классы, которые позволяют организовать соединение с бд. К данным классам относится AppContext.

Вывод: в данной лабораторной работе были изучены методологии объектно-ориентированного моделирования средствами UML, основные принципы объектно-ориентированного проектирования программного обеспечения, получили навыки проектирования структуры информационной системы с применением UML.