Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа по предмету «Проектирование информационных систем» на тему «Объектно-ориентированное модулирование. Структурные диаграммы UML»

Студент: Коршун Н. И.

ФИТ 4 курс 5 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# 1 Постановка задач

Темой данной лабораторной работы является построение UML-диаграмм пакетов и классов веб-приложения каталог автомобилей «AUTOSTOP», описание которого было представлено в лабораторной работе №1.

Web-приложение должно быть выполнено с использованием асинхронного программирования, взаимодействовать с базой данных, реализовано под разными платформами. Web-приложение должно представлять собой web-приложение с асинхронным UI с использованием фреймворка NextJS. Отображение, бизнес-логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Диаграмму вариантов использования разработать на основе UML, также необходимо разработать логическую схему базы данных и структурную схему приложения. Язык разработки проекта JavaScript, платформа «NodeJS». Развертывание конечного приложения для последующего использования должно осуществляться с использованием Docker.

Функционально web-приложение должно:

* обеспечивать возможность просмотра информации о различных моделях автомобилей (характеристики, фотографии);
* предоставлять функционал просмотра видеообзоров на автомобили;
* позволять пользователю сравнивать характеристики нескольких автомобилей;
* обеспечивать регистрацию и авторизацию пользователей с возможностью восстановления пароля через электронную почту
* поддерживать роли гостя, пользователя, администратора;
* для гостей доступен просмотр автомобилей и видеообзоров;
* для зарегистрированных пользователей доступен функционал сохранения избранных автомобилей и добавление автомобилей в список для сравнения;
* поддерживать функционал торговой площадки для продажи и покупки автомобилей с возможностью размещения объявлений, загрузки фотографий и контактов;
* администратор может добавлять и редактировать информацию о автомобилях, видеообзорах и управлять пользователями;
* поддерживать фильтрацию и поиск по различным параметрам автомобилей (марка, год выпуска, тип двигателя и т.д.);

Основные задачи приложения:

* разработать удобный интерфейс для поиска информации о автомобилях;
* предоставить функционал для просмотра видеообзоров на автомобили;
* создать систему сравнения характеристик автомобилей для более детального анализа.

# 2 Описание программных средств

Draw.io является мощным инструментом для создания диаграмм и визуального моделирования, который использовался для разработки и документирования архитектуры системы в процессе выполнения лабораторной работы. Приложение предоставило интуитивно понятный графический интерфейс для создания структурных схем, которые помогли визуализировать ключевые компоненты системы, а также их взаимосвязи и процессы взаимодействия. Использование draw.io стало важным шагом для проектирования системы, так как оно упростило процесс создания диаграмм, описывающих функциональные блоки приложения и их взаимодействие. Приложение также поддерживает интеграцию с различными облачными сервисами, что облегчило совместную работу над схемами и предоставило гибкость в управлении проектами.

Инструмент draw.io был выбран за его следующие особенности:

* Доступность: кроссплатформенная, поддержка работы в браузере, а также на Windows, macOS и Linux.
* Поддержка стандартов: возможность создания UML–диаграмм, блок–схем, диаграмм ER, IDEF0 и IDEF3, что позволило эффективно визуализировать бизнес–процессы и архитектуру системы.
* Легкость в использовании: интуитивно понятный интерфейс, который позволяет создавать диаграммы без необходимости владения сложными графическими навыками.
* Интеграция с облачными хранилищами: поддержка Google Drive, OneDrive, GitHub, что упрощает доступ к схемам и их совместное редактирование.

Основное назначение draw.io в этом проекте заключалось в создании визуальных моделей бизнес–процессов и архитектурных решений, которые легли в основу разработки системы. Эти диаграммы помогли в структурировании процессов разработки и визуализации потоков данных, что обеспечило более четкое понимание системы в целом.

Данные о технологии draw.io:

* Разработчик: JGraph Ltd.
* Адрес загрузки: https://app.diagrams.net.
* Использовался для: создания диаграмм и визуальных моделей бизнес–процессов и архитектурных решений системы.
* Доступность: кроссплатформенная, поддержка веб–версии и настольных приложений.

# 3 Описание практического задания

При разработке системы взаимодействуют несколько ключевых объектов, которые обеспечивают выполнение функциональных требований. Эти объекты включают в себя такие сущности, как пользователи, роли, сообщения, чаты, файлы и другие, каждая из которых определена специфическими атрибутами. Эти сущности связаны между собой, формируя структуру и поведение системы.

Диаграмма классов позволяет описать данные сущности. Она представлена на рисунке 3.1.

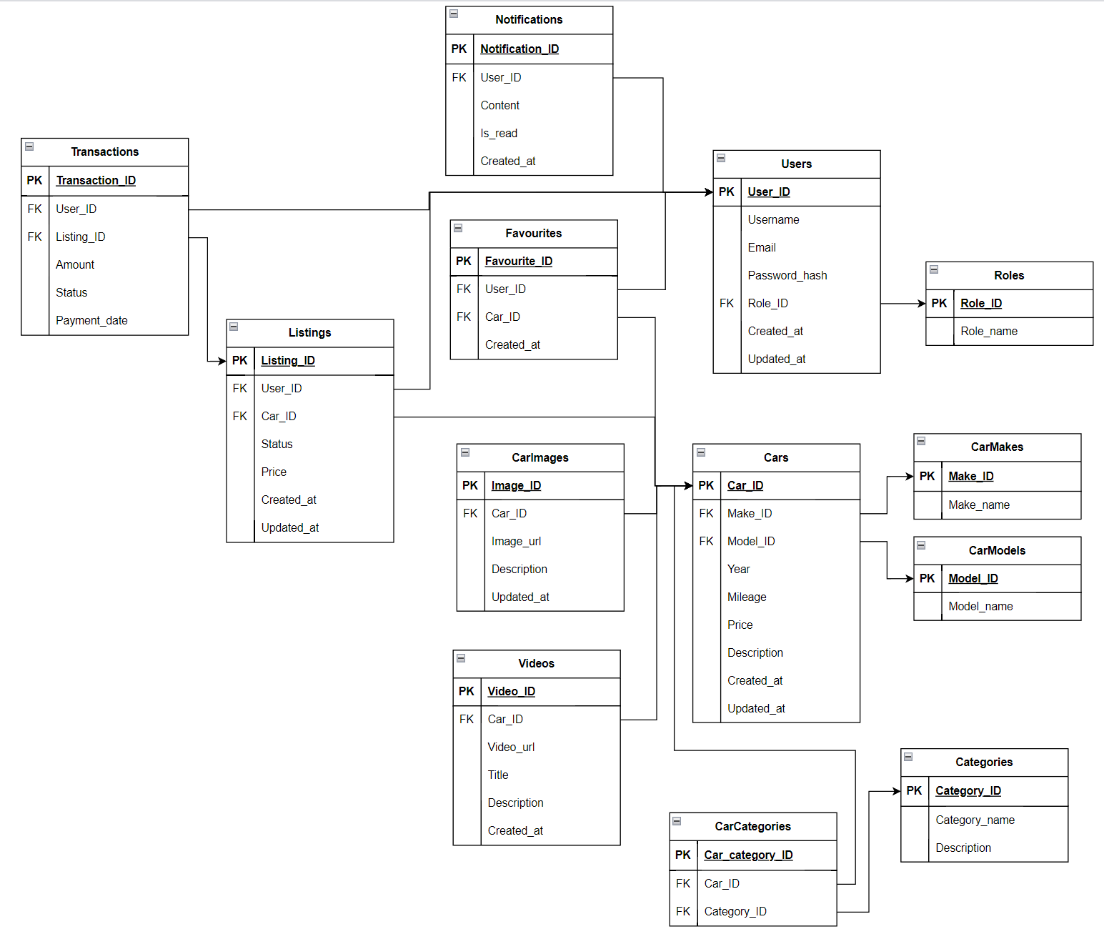


Рисунок 3.1 – Диаграмма классов

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.
* Articles – публичная переменная, содержащая доступ к таблице articles базы данных.
* Reception – публичная переменная, содержащая доступ к таблице receptions базы данных.

Также этот класс имеет следующие методы:

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

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.
  + Pass – пароль пользователя.
  + Avatar – фото пользователя.
  + Points – баллы 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:

* Username – ник 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.

ArticleController осуществляет обработку запросов, предназначенных статье. А именно: получение всех статей, поиск статьи по id, добавление и удаление. Содержит только публичные методы:

* + GetArticles – обработка запроса на получение указанного количества статей.
  + RemoveArticle – обработка запроса на удаление одной статьи с указанным id.
  + AddArticle – обработка запроса на добавление статьи в бд.

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

* + Id – id статьи.
  + Author– id пользователя, которому принадлежит данная статья.
  + Name – название статьи.
  + TextArticle – текст статьи.
  + DatetimePublish – дата и время публикации статьи в web-приложение.

Далее указаны публичные методы класса Articles:

* AddArticle – выполняет добавление статьи в бд.
* RemoveArticle – осуществляет удаление статьи из бд.
* CheckEmpty – выполняет проверку заполненности всех полей при отправке запроса.
* FindArticleById – осуществляет поиска статьи по указанному id.
* GetArticles – выполняет получение указанного количества статей из бд.

ArticleDto – класс, который используется для получения данных, необходимых при добавлении статьи в бд. Содержит только публичные свойства:

* IdUser – id пользователя, который публикует данную статью.
* Name – название статьи.
* Text – текст статьи.
* PubDate – дата и время публикации статьи.

ReceptionsController - осуществляет обработку запросов, предназначенных для назначения баллов за сданные отходы. Содержит только публичные методы:

* + Scoring – обработка запроса на назначение баллов.

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

* Id – id назнкачения.
* IdUser – id юзера которому назначают баллы.
* Waste – вид отхода который сдавали.
* Points –назначенные баллы.
* RecPoint – пункт приема.

Далее указаны все публичные методы данного класса:

* + Scoring – обработка запроса на назначение баллов.

ReceptionDto – класс, который используется для получения данных, необходимых при добавлении баллов в бд. Содержит только публичные свойства:

* IdUser – id пользователя, который оставляет комментарий.
* Points –назначенные баллы.
* RecPoint – пункт приема.

Далее представлены пакеты, в которые были сгруппированы классы. Диаграмма пакетов показана на рисунке 3.2.

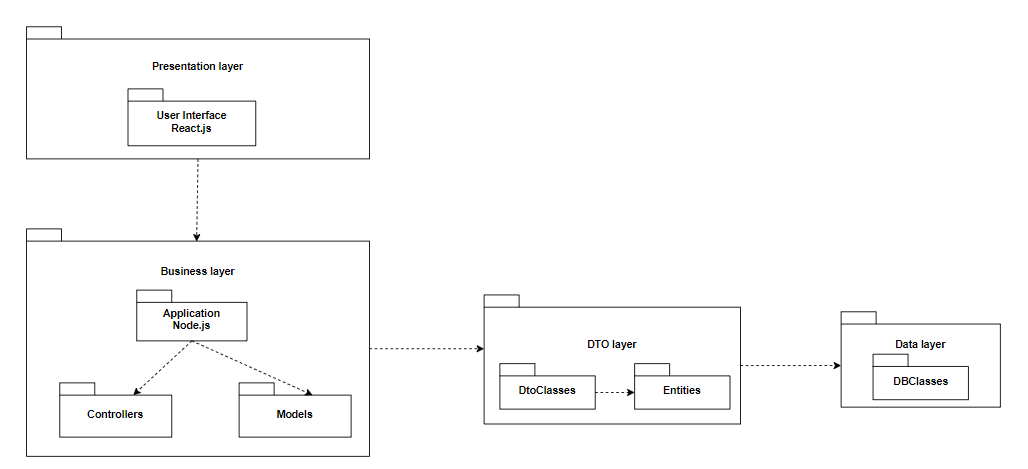


Рисунок 3.2 – Диаграмма пакетов

На диаграмме пакетов видно уровни и категории используемых данных. Controllers содержит классы, отвечающие за обработку запросов на сервер. К ним относятся: AuthController, ArticleController, RecepController.

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

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

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

# 4 Ответы на вопросы

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

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).

2. 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.

3. 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.

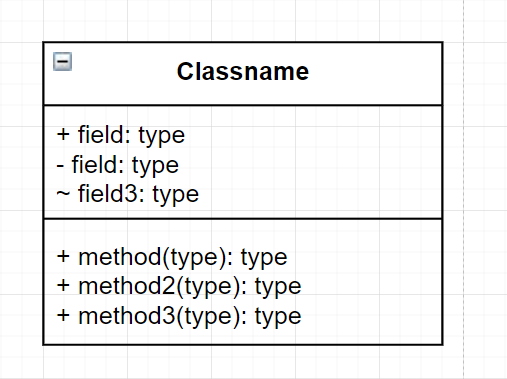


Рисунок 4.1 – Изображение класса

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.

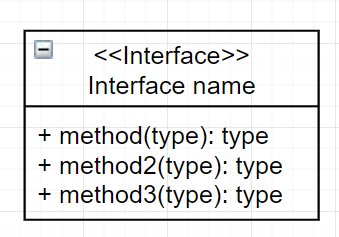


Рисунок 4.2 – Изображение интерфейса

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

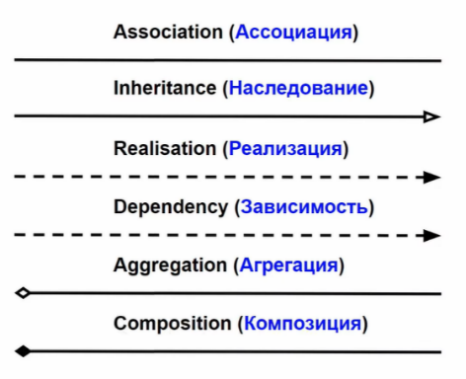


Рисунок 4.4 – Отношения в диаграмме классов

4. 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.

5. 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).