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.

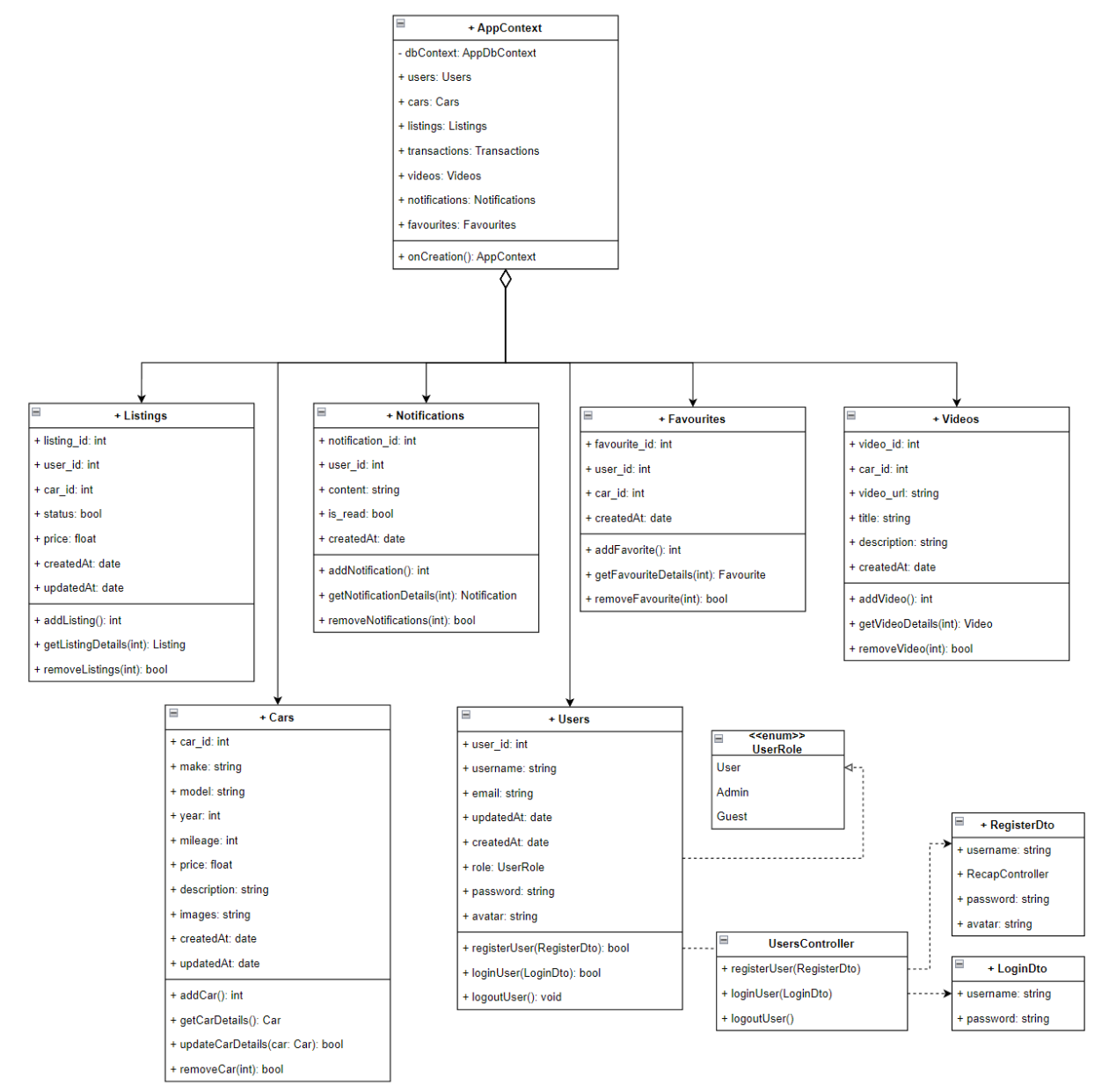


Рисунок 3.1 – Диаграмма классов

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

* dbContext — приватная переменная типа AppDbContext, содержащая экземпляр класса, управляющего доступом к базе данных. Она обеспечивает безопасное и структурированное взаимодействие с данными, хранящимися в базе, помогая управлять транзакциями и запросами;
* users — публичная переменная типа Users, предоставляющая доступ к таблице пользователей. Это позволяет управлять данными о пользователях, включая их регистрацию, авторизацию и профильные данные;
* cars — публичная переменная типа Cars, обеспечивающая доступ к таблице автомобилей. Она помогает организовывать и манипулировать данными об автомобилях, хранящихся в системе;
* listings — публичная переменная типа Listings, предоставляющая доступ к таблице объявлений. С ее помощью можно управлять списком объявлений, выполняя различные операции, связанные с ними;
* transactions — публичная переменная типа Transactions, позволяющая получить доступ к таблице транзакций и обрабатывать данные о финансовых операциях. Это помогает отслеживать и управлять платежными процессами;
* videos — публичная переменная типа Videos, предоставляющая доступ к таблице видео. Она дает возможность хранить и извлекать информацию о видеозаписях, связанных с автомобилями, улучшая пользовательский опыт;
* notifications — публичная переменная типа Notifications, предоставляющая доступ к таблице уведомлений. Она помогает управлять уведомлениями, обеспечивая возможность отправлять и получать уведомления для пользователей;
* favourites — публичная переменная типа Favourites, предоставляющая доступ к таблице избранного. Это позволяет пользователям добавлять объекты в избранное и управлять этим списком.

Класс также включает метод:

* onCreation — выполняется при создании экземпляра и инициализации базы данных, что обеспечивает готовность всех компонентов и подключений для дальнейшей работы.

Класс UsersController обрабатывает запросы, связанные с пользователями. Включает публичные методы:

* registerUser — обрабатывает запрос на регистрацию пользователя;
* loginUser — обрабатывает запрос на авторизацию;
* logoutUser — обрабатывает запрос на выход из аккаунта.

Класс Users содержит свойства и методы для обработки запросов от UsersController. Публичные свойства включают:

* user\_id — уникальный идентификатор пользователя;
* username — имя пользователя;
* email — электронная почта пользователя;
* createdAt — дата создания аккаунта;
* updatedAt — дата последнего обновления аккаунта;
* role — роль пользователя из перечисления UserRole (User, Admin, Guest);
* avatar — аватар пользователя.

Методы:

* registerUser — добавляет нового пользователя в базу данных;
* loginUser — проверяет учетные данные для входа пользователя;
* logoutUser — выполняет выход пользователя из аккаунта.

Перечисление UserRole определяет роли пользователя:

* User.
* Admin.
* Guest.

Класс Listings содержит данные и методы для обработки объявлений. Публичные свойства включают:

* listing\_id — уникальный идентификатор объявления;
* user\_id — идентификатор пользователя, создавшего объявление;
* status — статус объявления (активен или нет);
* price — цена товара или услуги;
* createdAt — дата создания объявления;
* updatedAt — дата последнего обновления объявления.

Методы:

* addListing — добавляет новое объявление;
* getListingDetails — получает детали объявления по его идентификатору;
* removeListing — удаляет объявление из базы данных.

Класс Cars описывает автомобиль и содержит следующие свойства:

* car\_id — уникальный идентификатор автомобиля;
* make — марка автомобиля;
* model — модель автомобиля;
* year — год выпуска;
* mileage — пробег автомобиля;
* price — цена автомобиля;
* description — описание автомобиля;
* images — изображения автомобиля;
* createdAt — дата создания записи о машине;
* updatedAt — дата последнего обновления записи.

Методы:

* addCar — добавляет информацию о новом автомобиле;
* getCarDetails — возвращает детали автомобиля по идентификатору;
* updateCarDetails — обновляет информацию о конкретном автомобиле;
* removeCar — удаляет запись автомобиля из базы данных.

Класс Notifications управляет уведомлениями, включает следующие свойства:

* notification\_id — идентификатор уведомления;
* user\_id — идентификатор пользователя, получившего уведомление;
* content — содержание уведомления;
* is\_read — статус уведомления (прочитано или нет);
* createdAt — дата создания уведомления.

Методы:

* addNotification — добавляет новое уведомление;
* getNotificationDetails — получает детали уведомления;
* removeNotifications — удаляет уведомление.

Класс Favourites хранит избранные объекты пользователя, свойства включают:

* favourite\_id — идентификатор избранного объекта;
* user\_id — идентификатор пользователя;
* car\_id — идентификатор избранного автомобиля;
* createdAt — дата добавления в избранное.

Методы:

* addFavourite — добавляет объект в избранное;
* getFavouriteDetails — получает информацию об избранном объекте;
* removeFavourite — удаляет объект из избранного.

Класс Videos описывает видео, включает следующие свойства:

* video\_id — уникальный идентификатор видео;
* car\_id — идентификатор автомобиля, связанного с видео;
* video\_url — URL-ссылка на видео;
* title — название видео;
* description — описание видео;
* createdAt — дата добавления видео.

Методы:

* addVideo — добавляет новое видео;
* getVideoDetails — получает детали видео;
* removeVideo — удаляет видео.

Класс RegisterDto используется для передачи данных при регистрации:

* username — имя пользователя;
* email — электронная почта;
* password — пароль;
* avatar — аватар пользователя.

Класс LoginDto используется для передачи данных при авторизации:

* username — имя пользователя;
* password — пароль пользователя.

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

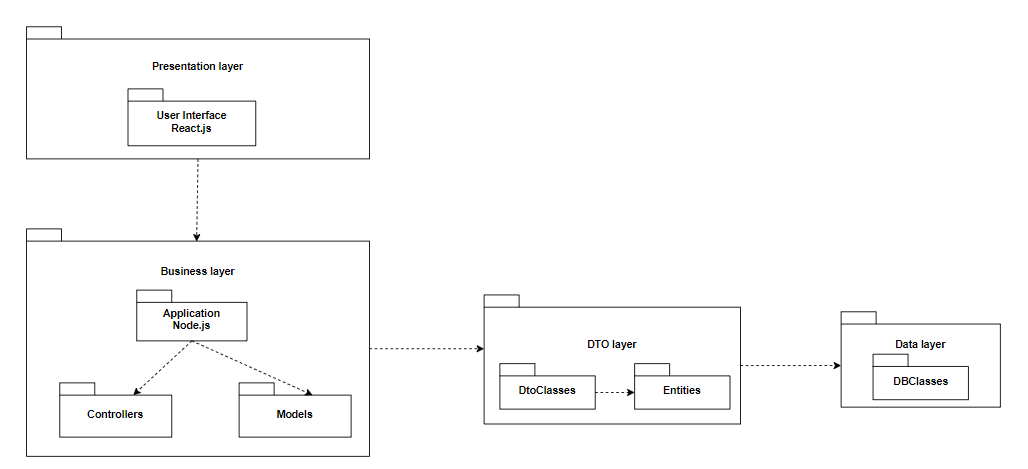


Рисунок 3.2 – Диаграмма пакетов

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

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

Models содержит классы, отвечающие за взаимодействие с базой данных. К этим классам относятся: Users, Cars, Listings, Notifications, Favourites и Videos.

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.

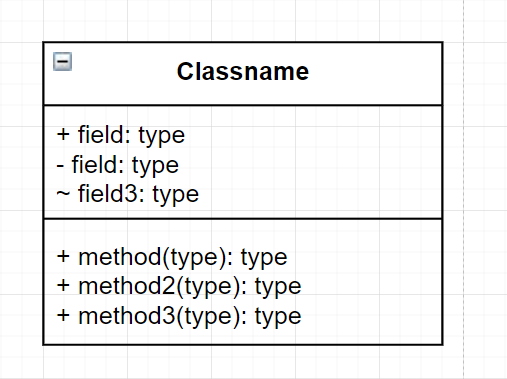


Рисунок 4.1 – Изображение класса

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.

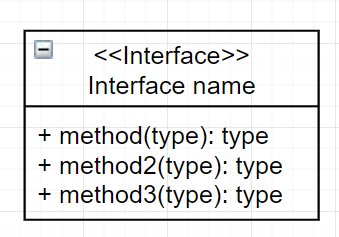


Рисунок 4.2 – Изображение интерфейса

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

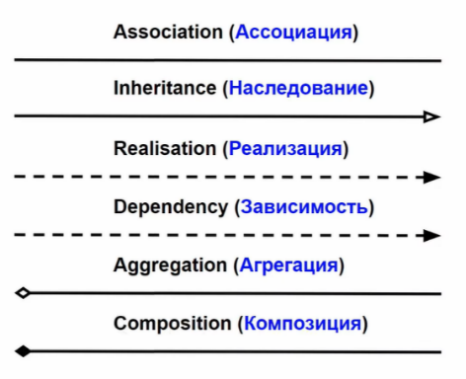


Рисунок 4.4 – Отношения в диаграмме классов

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).