Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа по предмету «Проектирование информационных систем» на тему «Объектно-ориентированное модулирование. Физические 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 развертывания и компонентов 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

Для полного физического представления программной системы необходимо иметь информацию о платформе и вычислительных средствах, на которых она функционирует. Отражение этой информации позволяет использовать диаграмму развертывания. Данная диаграмма позволяет лучше понять, как система организована и как взаимодействуют ее различные компоненты. Она является важным инструментом при проектировании и разработке программного обеспечения.

Сервер приложения играет ключевую роль в обработке данных, их формировании и передаче по удаленным каналам. Взаимодействие между клиентом и сервером осуществляется с использованием *HTTP* протокола, где клиент отправляет запросы на сервер. Ответы и данные обрабатываются сервером и возвращаются обратно клиенту.

Для хранения данных в программном средстве отведена роль *MySQL*, которая отвечает за все процессы, связанные с хранением и управлением данными. Благодаря использованию *MySQL*, приложение имеет надежное хранилище для своих данных.

Помимо этого, на диаграмме представлена более подробная архитектура программного средства, которая описывает использование различных технологий и библиотек в различных модулях системы. Это позволяет более детально изучить и понять взаимодействие компонентов и технологий в приложении.

Вся диаграмма развертывания представлена на рисунке 3.1.

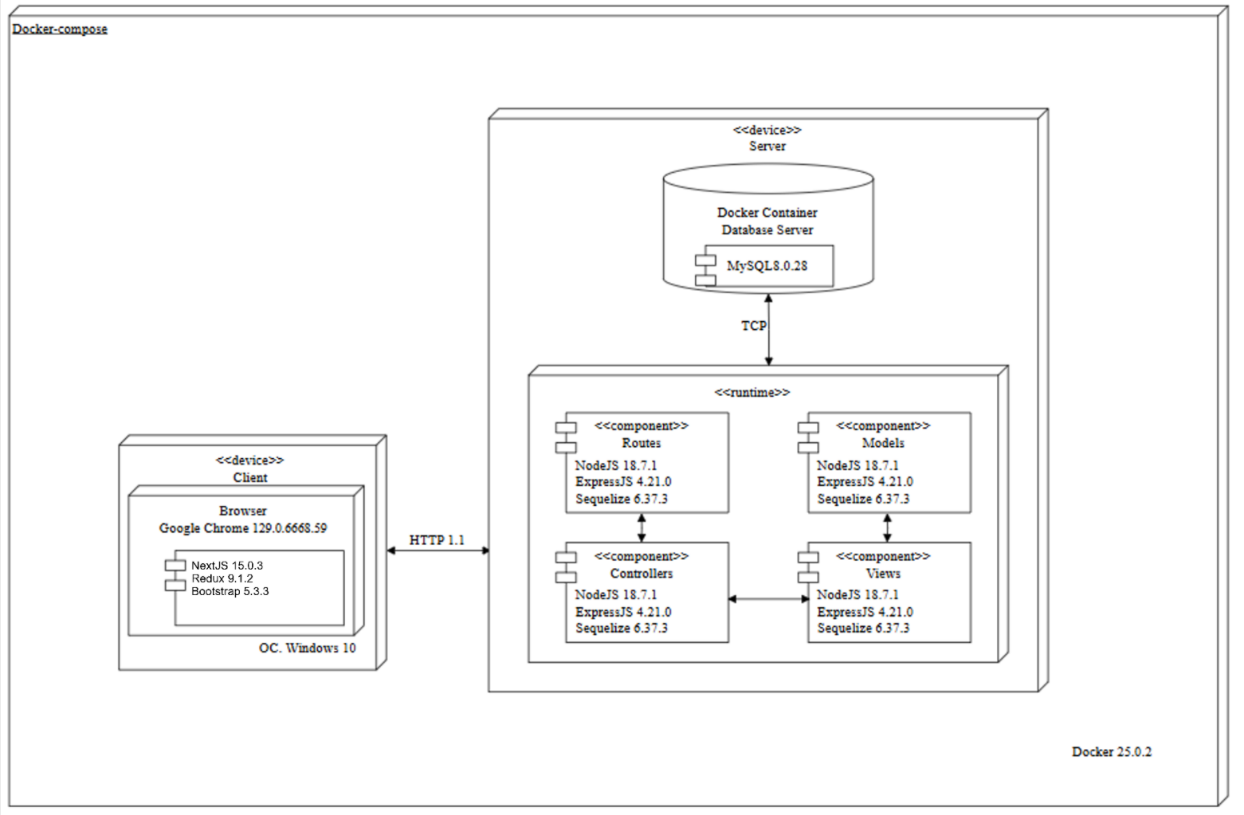


Рисунок 3.1 – Диаграмма развертывания

На диаграмме видны все описанные компоненты приложения.

# 4 Ответы на вопросы

1. Укажите назначение физических диаграмм: компонентов и развертывания.

Диаграмма компонентов позволяет определить состав программных компонентов в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма развёртывания же позволяет определить распределение компонентов системы по её физическим узлам; показать физические связи между всеми узлами реализации системы на этапе ее выполнения; выявить узкие места системы и реконфигурировать её топологию для достижения требуемой производительности.

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

Артефакт – некая физическая сущность, программный компонент, который используется или создаётся во время работы программного обеспечения.

Узел – то, что может содержать программное обеспечение (некоторый физически существующий элемент системы, обладающий вычислительным ресурсом).

Интерфейс – элемент, который служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры.

3. Опишите нотации, которые используются для представления компонентов.

Для представления компонентов используются две нотации – рисунок 2, отличаются они визуально.

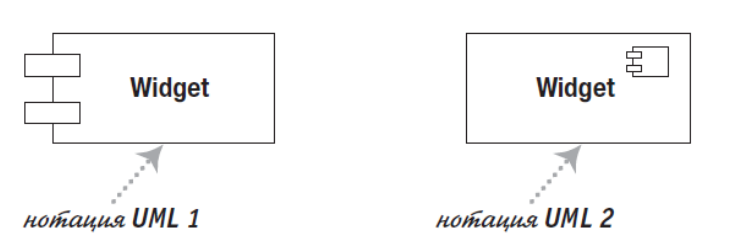


Рисунок 4.1 – Нотации для представления компонентов

4. Опишите основные нотации, которые используются для представления архитектуры системы в виде диаграммы развертывания.

Узлы (Nodes): Представляют физические устройства или вычислительные ресурсы, такие как серверы, ПК, мобильные устройства и т.д. Узлы изображаются в виде прямоугольников с именем устройства.

Артефакты (Artifacts): Представляют программные компоненты или данные, которые размещены на узлах. Они изображаются в виде прямоугольников или эллипсов внутри узлов.

Связи (Connections): Показывают взаимодействие между узлами. Они могут быть направленными или двунаправленными линиями, обозначающими поток данных, коммуникации или зависимостей между узлами.

5. Укажите основные виды связей между компонентами и между узлами

Сетевые связи (Network Links): Показывают физические сетевые соединения между узлами. Это могут быть Ethernet, Wi-Fi, VPN и другие типы сетевых соединений.

Коммуникационные связи (Communication Links): Представляют логические связи между компонентами системы. Это могут быть протоколы обмена данными, такие как HTTP, TCP/IP, MQTT и другие, которые описывают способы обмена информацией между компонентами.

Зависимости (Dependencies): Показывают, какие компоненты или узлы зависят от других. Например, если один компонент зависит от базы данных, это может быть представлено стрелкой или связью, указывающей на эту зависимость.

Репликация и балансировка нагрузки (Replication and Load Balancing): Если компоненты или данные реплицируются на несколько узлов или происходит балансировка нагрузки между узлами для повышения производительности и надежности, это может быть показано на диаграмме развертывания.

Прокси и шлюзы (Proxies and Gateways): Представляют промежуточные узлы, используемые для обеспечения безопасности, переадресации или трансляции данных между компонентами.