Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №6

«Объектно-ориентированное моделирование физической диаграммы UML»

Студент: Беласин Д.А.

ФИТ 3 курс 1 группа

Преподаватель: Якубенко К.Д.

Минск 2024

1. **Постановка задачи**

Задачей данной работы является детальное описание структуры взаимодействия компонентов, используемых в web-приложении, на основе данных, описанных в лабораторной работе №1.

Это предполагает построение точной и понятной модели, которая поможет глубже понять, как компоненты приложения взаимодействуют друг с другом, какие функции они выполняют, какие интерфейсы предоставляют и требуют, а также как они развёртываются в целевой среде.

Диаграмма компонентов — это важнейший элемент UML, позволяющий проектировщикам, разработчикам и другим участникам процесса разработки наглядно представлять физическую архитектуру системы. Диаграмма компонентов играет ключевую роль в разработке приложений, так как она демонстрирует:

* Физическую структуру приложения: как организованы его компоненты (библиотеки, исполняемые файлы, файлы конфигурации и т.д.).
* Взаимодействие компонентов: какие компоненты связаны между собой и каким образом (через интерфейсы, зависимости и т.д.).
* Размещение компонентов: где физически располагаются компоненты (на серверах, в облаке и т.д.).
* Зависимости между компонентами: какие компоненты зависят от других компонентов и какие интерфейсы используются для взаимодействия.

Диаграмма развёртывания — это важнейший элемент UML, который показывает физическое размещение компонентов приложения в среде выполнения. Она отображает:

* Узлы: физические или виртуальные машины, на которых развернуты компоненты.
* Компоненты размещенные на узлах компоненты приложения.
* Связи между узлами сети, по которым взаимодействуют узлы.
* Размещение компонентов на узлах, какие компоненты расположены на каких узлах.

Задачей данной работы является построение диаграммы компонентов и диаграммы развёртывания для интерфейса аренды парковочных мест, что включает в себя определение компонентов, их интерфейсов, зависимостей и физического размещения на узлах. Разработка таких диаграмм помогает визуализировать физическую архитектуру приложения, обеспечить правильное взаимодействие компонентов и подготовить основу для эффективного развёртывания и масштабирования.

# 2. Описание программных средств

Описание программного средства Draw.io:

* Название: Draw.io.
* Версия: Актуальная версия (по состоянию на 2024 год). Программа обновляется регулярно.
* Разработчик: JGraph Ltd.
* Адрес загрузки: https://app.diagrams.net/. Draw.io доступен как онлайн-приложение, так и в виде настольного клиента для различных операционных систем.
* Режим использования: Draw.io предоставляет бесплатный доступ ко всем своим функциям без ограничений. Это полностью бесплатный инструмент с открытым исходным кодом, доступный как для личного, так и для корпоративного использования.
* Платформы: Draw.io доступен как веб-приложение, которое работает в любом современном браузере. Также доступны настольные версии для Windows, macOS и Linux.
* Типы моделей, с которыми работает Draw.io: Draw.io используется для создания диаграмм, схем, моделей процессов и прочих визуальных структур. Программа поддерживает работу с блок-схемами, диаграммами UML, IDEF0, сетевыми диаграммами, архитектурными схемами, организационными диаграммами и многими другими визуальными представлениями данных.
* Основные функции: Создание и редактирование диаграмм различных типов (например, UML, блок-схемы, BPMN, ERD). Поддержка импорта и экспорта файлов в различных форматах, включая XML, PNG, SVG и PDF. Возможность совместной работы с другими пользователями через облачные сервисы (Google Drive, OneDrive, Dropbox и др.). Функции автосохранения и интеграции с популярными системами управления версиями. Поддержка версии оффлайн, которая позволяет работать без подключения к интернету. Гибкая система настройки шаблонов и библиотек элементов для ускорения работы.

Draw.io — это удобный, бесплатный инструмент для создания диаграмм и схем, который поддерживает совместную работу и интеграцию с облачными сервисами, обеспечивая гибкость и доступность на различных платформах.

**3.** **Практическое задание**

Диаграмма развертывания для нашего проекта будет выглядеть следующим образом – Рисунок 3.1.

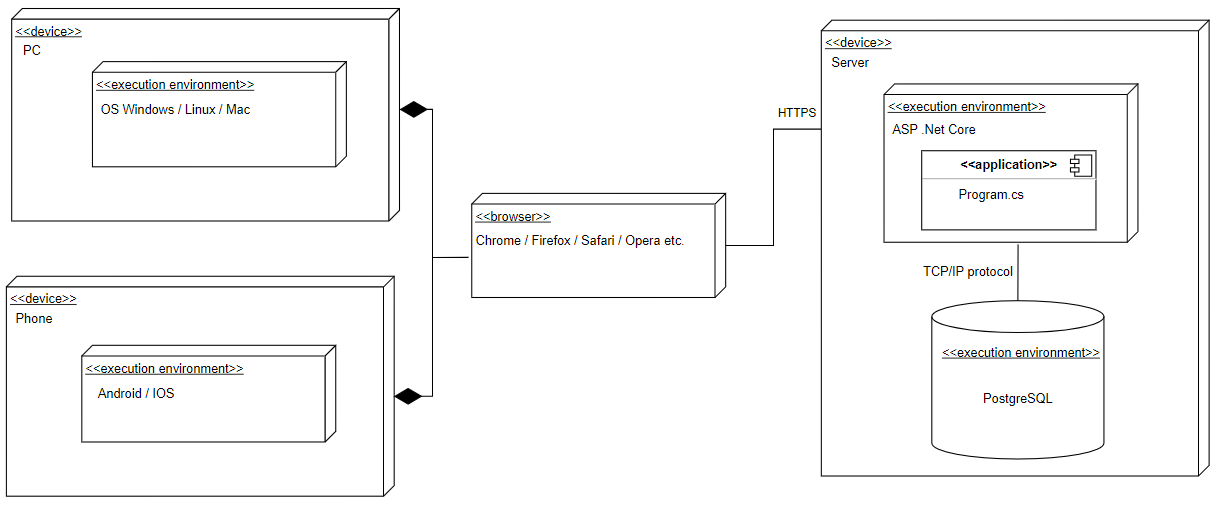


Рисунок 3.1 – Диаграмма развертывания

Разберем элементы диаграммы развертывания:

* Device PC.
* Device Phone.
* Browser.
* Server.
* Databse.
* Application.

Диаграмма развертывания иллюстрирует физическое размещение компонентов. Клиентские устройства (PC и Phone) через браузеры взаимодействуют с сервером, который, в свою очередь, обращается к базе данных.

Все это можно представить как приложение (Application), состоящее из взаимосвязанных компонентов, развернутых на различных устройствах и сервере.

Использование React на стороне клиента и ASP.NET Core на стороне сервера, а также PostgreSQL в качестве СУБД, обеспечивает современный и масштабируемый подход к разработке web-приложений.

Применение Entity Framework упрощает взаимодействие с базой данных, повышая производительность разработки.

Далее рассмотрим диаграмму компонентов на рисунке 3.2:

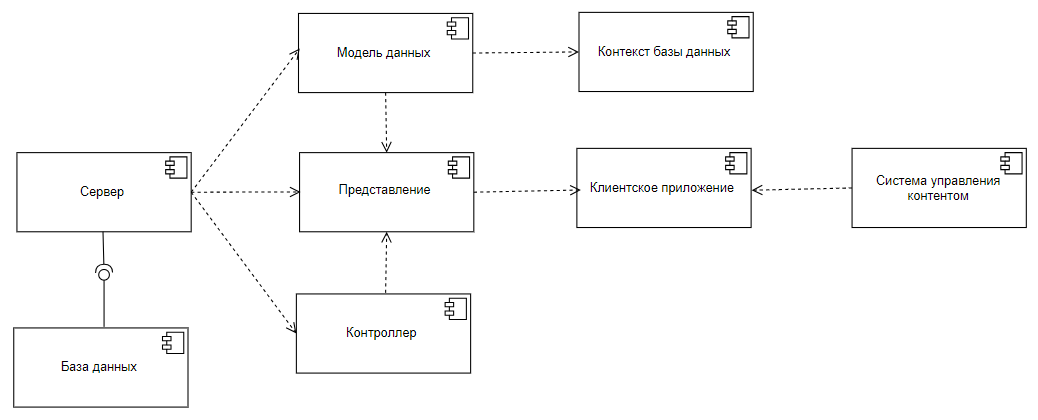


Рисунок 3.2 – Диаграмма компонентов

Опишем используемые компоненты и технологии их реализации в нашей архитектуре:

1. База данных: используется СУБД PostgreSQL.
2. Контекст базы данных: используется EntityFramework.
3. Сервер: используется ASP .NET CORE.
4. Модель данных: используются классы C#.
5. Представление: используется Razor.
6. Контроллер: используются классы C#.
7. Клиентское приложение: используется React.
8. Система управления контентом: используется React CMS.

Связи между компонентами были описаны зависимостью и интерфейсом. Сервер взаимодействует с базой данных при помощи интерфейса.

В данной архитектуре web-приложения используется многоуровневый подход. Серверная часть, реализованная на ASP.NET Core, взаимодействует с базой данных PostgreSQL через контекст Entity Framework.

Модель данных представлена классами C#. Логика обработки запросов реализована в контроллерах (C#), а представление формируется с помощью Razor. Клиентская часть приложения построена на React, включая систему управления контентом React CMS.

# Теоретические вопросы подготовки к лабораторной работе

1. Назначение физических диаграмм компонентов и развертывания:

Диаграмма компонентов используется для отображения физических компонентов системы и связей между ними.

Она показывает, как программные компоненты (например, модули, библиотеки или исполняемые файлы) взаимодействуют и каким образом они связаны друг с другом.

Это важно для разработки архитектуры программного обеспечения, позволяя моделировать физическое распределение логических классов, повышая эффективность проектировки.

Диаграмма развертывания используется для представления физической инфраструктуры системы, показывая, как и где компоненты программного обеспечения будут развернуты.

Она включает в себя узлы (серверы, компьютеры и устройства) и описывает, как они соединяются для взаимодействия.

Диаграмма развертывания помогает определить, какие аппаратные ресурсы будут использоваться и как они должны быть настроены для работы системы.

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

Узел — это физическая или виртуальная вычислительная платформа (например, сервер, компьютер, устройство), на которой может выполняться программное обеспечение. Узлы на диаграмме развертывания представляют аппаратные устройства или системы.

Артефакт — это конкретный файл или компонент, являющийся результатом разработки программного обеспечения (например, исполняемый файл, библиотека, база данных). Артефакты могут быть развернуты на узлах, что повышает эффективность разработки.

Интерфейс — это контракт или соглашение, описывающее набор методов или функций, которые компонент или узел предоставляет или требует для взаимодействия с другими компонентами.

1. Нотации для представления компонентов:

* Стандартный компонент представлен прямоугольником с изображением маленькой "папки" в углу (или стереотипом «component»). Такой элемент показывает единицу программного кода или приложения, что помогает в удобстве разработки.
* Вариации компонента могут включать стереотипы для различия между разными типами компонентов.
* Также компоненты могут иметь интерфейсы, которые отображаются в виде полуэллипсов (для предоставляемых интерфейсов) или кругов с линиями (для требуемых интерфейсов).

1. Нотации для представления архитектуры системы на диаграмме развертывания:

* Узел отображается в виде трехмерного куба, представляющего физическое или виртуальное устройство.
* Связи между узлами (например, коммуникации по сети) обычно изображаются в виде линий между узлами.
* Артефакты, развернутые на узлах, могут быть изображены внутри узлов, показывая, где и как выполняются конкретно выбранные программные компоненты.
* Компоненты и артефакты могут быть связаны с узлами через линии ассоциации, что демонстрирует их развертывание на конкретно выбранном устройстве.

1. Основные виды связей между компонентами и между узлами:

* Зависимость (dependency) — показывает, что один компонент зависит от другого для своей работы.
* Ассоциация (association) — связь между двумя узлами или компонентами, означающая, что они взаимодействуют.
* Реализация (realization) — показывает, что компонент или узел реализует интерфейс, предоставленный другим элементом.
* Композиция (composition) — более сильная связь, показывающая, что один компонент является частью другого.
* Связи между узлами обычно отображают физические каналы связи, например, такие каналы связи как сеть или прямое подключение между устройствами.

Диаграммы компонентов и развертывания являются важными инструментами моделирования в UML, позволяющими визуализировать физическую архитектуру программного обеспечения и его развертывание в инфраструктуре.

Диаграмма компонентов фокусируется на взаимодействии физических компонентов (библиотеки, исполняемые файлы и т.д.), в то время как диаграмма развертывания показывает, как эти компоненты размещаются на физических или виртуальных узлах (серверах, компьютерах, устройствах).