### Undestanding C# and .NET

**C# (произносится как C-шарп):**

**Назначение: C# - это современный, общего назначения язык программирования, разработанный Microsoft. Он предназначен для создания широкого спектра приложений, начиная от программ для настольных компьютеров до веб-приложений и мобильных приложений.**

**Синтаксис: Синтаксис C# похож на другие языки, основанные на языке C, такие как C++ и Java, что облегчает разработчикам, знакомым с этими языками, переход на C#.**

**Объектно-ориентированный: C# - это язык программирования, ориентированный на объекты, что означает поддержку принципов инкапсуляции, наследования и полиморфизма.**

**.NET (произносится как дот-нет):**

**Назначение: .NET - это фреймворк, разработанный Microsoft, который предоставляет платформу для создания и запуска программных приложений. Он поддерживает несколько языков программирования, включая C#.**

**Компоненты .NET:**

**Общая языковая среда выполнения (CLR): Это среда выполнения, которая управляет выполнением программ .NET. Она обеспечивает управление памятью, безопасность и выполнение кода.**

**Базовая библиотека классов (BCL): Сборник предварительно созданных классов и функций, предоставляющих общую функциональность. Эта библиотека упрощает задачи, такие как ввод-вывод файлов, сетевые операции и другие.**

**.NET Core против .NET Framework:**

**.NET Framework: Традиционный фреймворк, в основном используемый для приложений под управлением Windows.**

**.NET Core: Переносимый, открытый фреймворк, который более модульный и поддерживает разработку под Windows, macOS и Linux.**

**Понимание взаимосвязи:**

**C# - это язык программирования, предназначенный для написания кода.**

**.NET - это фреймворк, который предоставляет среду выполнения и библиотеки для выполнения и поддержки этого кода.**

**Разработчики используют C# для написания приложений, которые выполняются на фреймворке .NET.**

**Основные концепции:**

**Компиляция: Обычно код на C# компилируется в промежуточный язык (IL), который затем выполняется средой выполнения общего языка (CLR) во время выполнения.**

**Сборки: Скомпилированный код C# обычно упаковывается в сборки, которые можно обменивать и использовать в различных проектах.**

**Пространство имен: Пространства имен в C# помогают организовывать код, предотвращая конфликты имен и улучшая читаемость кода.**

**Visual Studio: Интегрированная среда разработки Microsoft, обычно используемая для разработки на C# и .NET.**

**В итоге C# - это язык программирования, используемый для написания кода, и .NET - это фреймворк, который предоставляет среду выполнения и библиотеки для выполнения этого кода. Вместе они позволяют разработчикам создавать различные приложения для разных платформ.**

### NET SDK vs .NET Runtime

**.NET SDK (Software Development Kit) и .NET Runtime - это два ключевых компонента, используемых в экосистеме .NET. Давайте разберем их различия:**

**.NET SDK (Software Development Kit):**

**Назначение: SDK предназначен для разработки приложений с использованием .NET.**

**Содержание SDK:**

**Компилятор C#: Инструмент, преобразующий исходный код на C# в исполняемый код.**

**Библиотеки и инструменты: Набор библиотек и инструментов, необходимых для разработки, тестирования и управления проектами.**

**dotnet CLI (Command Line Interface): Командная строка, позволяющая выполнять различные задачи, такие как компиляция, тестирование, создание проектов и др.**

**NuGet: Пакетный менеджер для управления зависимостями и пакетами в проекте.**

**.NET Runtime:**

**Назначение: Runtime предоставляет среду выполнения для .NET-приложений, запуская их на целевой машине.**

**Содержание Runtime:**

**Common Language Runtime (CLR): Основной компонент, обеспечивающий выполнение программы, управление памятью, безопасность и другие аспекты времени выполнения.**

**Just-In-Time (JIT) компилятор: Преобразует промежуточный язык (IL), который генерируется при компиляции, в машинный код во время выполнения.**

**Base Class Library (BCL): Набор предварительно созданных классов и функций для общих задач, таких как работа с файлами, сетевое взаимодействие и многое другое.**

**Важные моменты:**

**Если вы разрабатываете приложение, вам нужен .NET SDK для создания и управления проектами, написания кода и выполнения различных задач разработки.**

**Если вы запускаете готовое .NET-приложение на целевой машине, вам нужен .NET Runtime, который обеспечивает выполнение приложения.**

**Часто .NET SDK включает в себя .NET Runtime, поэтому при разработке приложения обычно вы используете .NET SDK, а при запуске приложения на целевой машине — .NET Runtime.**

### . NET CLI

**.NET CLI (Command-Line Interface) - это интерфейс командной строки, предоставляемый инструментами и командами для разработки приложений с использованием платформы .NET.**

**Назначение:**

**.NET CLI предоставляет команды, которые позволяют вам управлять проектами, компилировать их, запускать, тестировать и выполнять различные задачи в рамках разработки приложений.**

**Основные команды .NET CLI:**

**dotnet new: Создание нового проекта.**

**dotnet build: Компиляция проекта.**

**dotnet run: Запуск приложения.**

**dotnet test: Запуск тестов.**

**dotnet publish: Публикация проекта для развертывания.**

**dotnet restore: Восстановление зависимостей проекта.**

1.1.3 Ma'lumot toifalari va o'zgaruvchilar

2. Variables

### Naming convention

**Соглашения по именованию - это набор правил и рекомендаций, используемых для определения имен переменных, функций, классов и других элементов в вашем коде. Соблюдение согласованных соглашений по именованию делает ваш код более читаемым и поддерживаемым. Вот некоторые общепринятые практики, используемые в C#:**

**Pascal Case (ВерблюжийРегистр):**

**Первая буква каждого слова в имени пишется с заглавной буквы.**

**Используется для имен классов, методов и свойств.**

**Пример: МойКласс, РассчитатьОбщуюСумму(), ИмяПользователя.**

**Camel Case (ПрописнаяБукваВНачалеСлова):**

**Первая буква каждого слова, кроме первого, пишется с заглавной буквы.**

**Используется для имен переменных, параметров и локальных переменных.**

**Пример: мояПеременная, рассчитатьОбщуюСумму(), имяПользователя.**

**ВерхнийРегистр (КОНСТАНТЫ):**

**Используйте все заглавные буквы для имен констант.**

**Разделяйте слова подчеркиванием.**

**Пример: МАКС\_ЗНАЧЕНИЕ, ПИ, ВРЕМЯ\_ОЖИДАНИЯ\_ПО\_УМОЛЧАНИЮ.**

**Префикс подчеркивания для частных переменных:**

**Используйте префикс подчеркивания для обозначения частных переменных экземпляра.**

**Пример: \_мояЧастнаяПеременная, \_имяКлиента.**

**Аббревиатуры:**

**Избегайте использования аббревиатур в именах, если они не являются широкоизвестными и широко принятыми.**

**Используйте полные, описательные имена для большей ясности.**

**Пример: количествоЭлементов вместо колЭлементов, имяКлиента вместо имяКл.**

**Именование пространств имен:**

**Используйте осмысленные и четкие имена для пространств имен.**

**Следуйте ВерблюжьемуРегистру для имен пространств имен.**

**Пример: МояКомпания.МойПродукт.Утилиты.**

**Именование интерфейсов:**

**Начинайте имя интерфейса с буквы 'I'.**

**Используйте ВерблюжийРегистр для остальной части имени интерфейса.**

**Пример: ILogger, IServicе.**

**Суффикс для логических переменных:**

**Используйте суффиксы типа Is, Has или Can для логических переменных и свойств.**

**Пример: включен, имеетДанные, можноРедактировать.**

**Формирование множественного числа:**

**Используйте форму множественного числа для коллекций и массивов.**

**Пример: клиенты, сотрудники, элементы.**

**Описательные имена:**

**Выбирайте имена, передающие назначение или смысл переменной или функции.**

**Избегайте однобуквенных имен, если они не используются в качестве** счетчиков циклов.

4. Primitive Data types

### **Dynamic vs object types**

**dynamic:**

**Динамическая типизация: Переменные типа dynamic предоставляют динамическую типизацию, что означает, что их тип определяется во время выполнения программы.**

**Отложенное связывание: Выполнение вызовов методов и доступ к членам объекта типа dynamic определяется во время выполнения, что позволяет откладывать некоторые решения до этого момента.**

**object:**

**Статическая типизация: Переменные типа object являются статически типизированными, и их тип определяется во время компиляции программы.**

**Низкоуровневая общность: Тип object является базовым для всех типов в .NET и может хранить значения любого типа, так как все типы в .NET наследуют от object.**

**Различия:**

**Производительность: Тип dynamic может вносить некоторое снижение производительности из-за отложенного связывания, в то время как использование object может быть более производительным, но менее гибким.**

**Безопасность типов: Использование dynamic может привести к ошибкам во время выполнения, если не предприняты соответствующие меры предосторожности, в то время как object может потребовать явного приведения типов.**

**Гибкость: dynamic предоставляет большую гибкость, так как тип определяется во время выполнения, что полезно при работе с динамическими данными, например, при работе с данными из JSON.**

**Безопасность компиляции: Использование object обеспечивает безопасность на уровне компиляции, что может помочь предотвратить ошибки связанные с типами.**

### Getting and Setting default values

**Использование значений по умолчанию в программировании имеет несколько причин и польз:**

**Избегание ошибок:**

**Установка значений по умолчанию может предотвратить ошибки, связанные с попыткой использовать переменные или объекты, которые не были инициализированы.**

**Безопасность кода:**

**Использование значений по умолчанию способствует безопасности кода, так как предотвращает использование неинициализированных переменных или объектов, что может привести к неопределенному поведению.**

**Обеспечение работы кода:**

**Устанавливая значения по умолчанию, вы можете гарантировать, что ваш код будет работать, даже если не все параметры были явно заданы.**

**Удобство использования:**

**Использование значений по умолчанию делает код более удобным для использования, так как часто нет необходимости явно указывать значения, если они могут быть автоматически определены.**

**Предоставление альтернативных вариантов:**

**Значения по умолчанию могут предоставлять альтернативные варианты для тех случаев, когда конкретное значение не задано. Это особенно полезно при проектировании API и библиотек.**

7. Input/Output

### **Casting va Converting**

**"Casting" (приведение типов) и "Converting" (конвертация) - это два термина, которые относятся к изменению типов данных в программировании. Давайте разберемся с их значениями:**

**Casting (Приведение типов):**

**Неявное приведение (Implicit Casting):**

**Происходит автоматически компилятором.**

**Приведение от менее точного типа к более точному.**

**Нет потери данных.**

**Пример:**

**csharpCopy code**

**int intValue = 10; double doubleValue = intValue; // Неявное приведение**

**Явное приведение (Explicit Casting):**

**Происходит вручную с использованием операторов явного приведения.**

**Приведение от более точного типа к менее точному.**

**Возможна потеря данных.**

**Пример:**

**csharpCopy code**

**double doubleValue = 10.5; int intValue = (int)doubleValue; // Явное приведение**

**Converting (Конвертация):**

**Convert класс:**

**Используется класс Convert для преобразования одного типа данных в другой.**

**Предоставляет методы для конвертации различных типов данных.**

**Пример:**

**csharpCopy code**

**string strValue = "123"; int intValue = Convert.ToInt32(strValue); // Конвертация строки в целое число**

**Parse метод:**

**Многие типы данных в C# имеют метод Parse, который позволяет преобразовать строку в соответствующий тип.**

**Пример:**

**csharpCopy code**

**string strValue = "123"; int intValue = int.Parse(strValue); // Преобразование строки в целое число**

**TryParse метод:**

**Предоставляет безопасный способ преобразования строки в число с обработкой ошибок.**

**Возвращает булево значение, показывающее успешность преобразования.**

**Пример:**

**csharpCopy code**

**string strValue = "123"; bool success = int.TryParse(strValue, out int intValue);**

**Различия:**

**Приведение типов (Casting) чаще используется для преобразования между схожими типами.**

**Конвертация (Converting) обычно включает преобразование между различными типами данных, включая строки.**

**Оба эти подхода важны в программировании и выбор между ними зависит от конкретной ситуации и требований задачи. Важно также учитывать возможные потери данных при приведении или конвертации.**

1. Implicit and explicit casting

2. System.Convert method

3. ToString()

4. Parse and TryParse

4

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

1

1.1

### Все операторы

В языке программирования C# существует несколько категорий операторов, которые выполняют различные операции над данными. Давайте рассмотрим эти категории на русском языке:

1. Бинарные операторы (Binary Operators):

Бинарные операторы применяются к двум операндам и выполняют операции, такие как сложение, вычитание, умножение и деление.

Примеры бинарных операторов:

int sum = a + b; // Сложение int difference = a - b; // Вычитание int product = a \* b; // Умножение int quotient = a / b; // Деление

2. Унарные операторы (Unary Operators):

Унарные операторы применяются к одному операнду и выполняют операции, такие как инкремент, декремент, отрицание и другие.

Примеры унарных операторов:

int x = 5; x++; // Инкремент x--; // Декремент bool isTrue = !true; // Логическое отрицание

3. Операторы присваивания (Assignment Operators):

Операторы присваивания используются для присвоения значения переменной.

Примеры операторов присваивания:

int a = 10; int b = 5; a += b; // Эквивалентно a = a + b;

4. Логические операторы (Logical Operators):

Логические операторы используются для выполнения операций над логическими значениями (true/false).

Примеры логических операторов:

bool result = (a > b) && (c < d); // Логическое И (AND) bool anotherResult = (x == y) || (z != 0); // Логическое ИЛИ (OR)

5. Условные логические операторы (Conditional Logical Operators):

Условные логические операторы используются для выполнения операций, основанных на условии.

Пример условного логического оператора:

int maxValue = (a > b) ? a : b; // Тернарный оператор ?:

6. Битовые и операторы сдвига (Bitwise and Binary Shift Operators):

Битовые и операторы сдвига применяются к битам в двоичном представлении чисел.

Примеры битовых операторов:

int bitwiseResult = a & b; // Побитовое И (AND) int shiftRight = a >> 2; // Бинарный сдвиг вправо на 2 бита

7. Разнообразные операторы (Miscellaneous Operators):

В эту категорию входят операторы **nameof** и **sizeof**, которые используются для получения имени переменной и размера типа соответственно.

Примеры операторов **nameof** и **sizeof**:

string variableName = nameof(myVariable); // Получение имени переменной int sizeOfInt = sizeof(int); // Получение размера типа

Каждый из этих операторов выполняет определенные задачи и имеет свои спецификации. Выбор оператора зависит от требований конкретной задачи.

8

1.1.6 Shart va tanlash operatorlari

1. If

2. If else

3. nested if

4. Ternary operator

5.switch

8

1.1.7 Takrorlanish operatorlari

1. for

2. while

3. do while

4. foreach

5. continue, break

8

1.1.8 Funksiyalar bilan ishlash

1. Funksiya nima?

2. Funksiyaning kiruvchi parametrlari(ref and out)

3. Qiymat qaytaradigan va qaytarmaydigan

funksiyalar

4. Rekursiv funksiyalar

5. Debugging

8

1.1.9 Matnlar bilan ishlash

1. String classi metodlari

2. String formatting

3. String interpolation

4. Verbatim string

12

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

1

1.1

1.1.10 Massivlar bilan ishlash

1. One demensional

2. Multi demensional

3. Array class metodlari

12

1.1.11 Amaliyot va oraliq nazorat Amaliyot va oraliq nazorat 4

1.2 C# intermediate 80

1.2.1 OOPga kirish

1. OOP nima?

2. Class, field, method, constructor, destructor

3. Object instance hosil qilish, this keyword

4. Access modifiers

8

1.2.2 OOP Concepts, Encapsulation

1. OOP pillars

2. Encapsulation

3. getter and setter

8

1.2.3 Inheritence, base, static, sealed

keyoword

1. Inheritence

2. base keyword

3. sealed class and methods

4. static class, static constructors

8

1.2.4 Polymorphism, Aggregation

1. Polymorphism

2. function and operator overloading

3. Overriding

4. Aggregation

8

1.2.5 Abstraction, Interface

1. Abstraction, abstract class

2. Interface

3. nested classes

4. partial class

8

1.2.6 Enum, struct va generic toifalari

1. Enum

2. Generics

3. struct

4

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

1

1.2

1.2.7 Collections(1-qism)

1. Systems.Collections

2. Array vs ArrayList

3. List, LinkedList, SortedList

4. Stack, Queue

5. Dictionary vs HashTable

8

1.2.8 Collections(2-qism)

1. IEnumerable, ICollection, IList, IDictionary

2. Indexer

3. Iteration, yield operator

8

1.2.9 Tuple va Nullable types

1. Tuple

2. ValueTuple

3. Nullable types

4. Null-coalescing operator

4

1.2.10 Event va Delegate types

1. Delegate

2. Event

3. Anonymous methods

4. Lambda expression

12

1.2.11 Amaliyot va oraliq nazorat Amaliyot va oraliq nazorat 4

1.3 C# advanced 80

1.3.1 Exception handling

1. Exceptions

2. try, catch, final

3. throw

4

### Working with text

**Работа с текстом** в программировании включает в себя использование различных инструментов и подходов для манипулирования строками, сравнения текстовых данных и поиска шаблонов. Давайте рассмотрим три ключевых аспекта работы с текстом на русском языке:

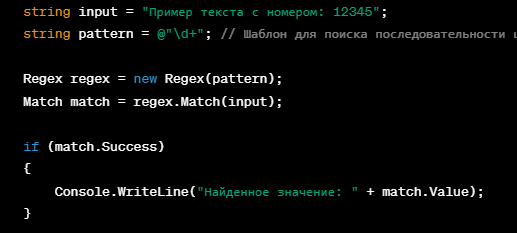
StringBuilder (СтроительСтрок):

**StringBuilder** - это класс в языке программирования C#, который предоставляет эффективные средства для изменения и работы со строками, особенно при выполнении множества операций конкатенации строк.

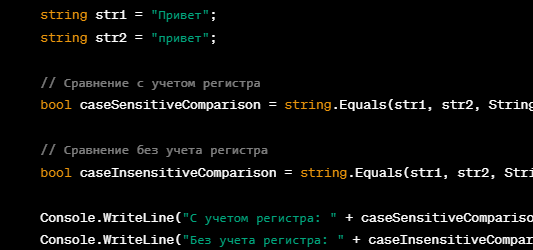


Regex (Регулярные выражения):

Регулярные выражения - это мощный инструмент для работы с текстовой информацией. Они позволяют осуществлять поиск, сопоставление и обработку строк на основе шаблонов.



Сравнение строк (String Comparison):

Сравнение строк включает в себя сравнение содержимого строк, игнорируя регистр, и другие параметры. 

1.3.3 Vaqt bilan ishlash

**Работа с временем** в программировании часто включает использование различных классов и структур для представления и манипуляций с временными данными. Давайте рассмотрим три ключевых аспекта работы с временем на русском языке:

### 1. Класс DateTime:

**DateTime** - это класс в языке программирования C#, который предоставляет функциональность для работы с датами и временем. Он позволяет создавать, представлять и выполнять различные операции с датой и временем.

Пример использования класса **DateTime**:

1.3.4 File I/O

1. File, FileInfo

2. StreamWriter, StreamReader

3. MemoryStream, FileStream

4. Directory, DirectoryInfo

5. DriveInfo

6. Path

### Object serialization, JSON

**Сериализация объектов - это процесс преобразования объектов в формат, который можно сохранить или передать по сети, и в последующем восстановить из этого формата обратно в объекты. В языке программирования C# для сериализации объектов широко используются различные средства. Давайте рассмотрим три из них на русском языке:**

**1. JsonSerializer (Сериализатор JSON):**

**JsonSerializer - это класс в пространстве имен System.Text.Json, предоставляющий функциональность для сериализации и десериализации объектов в формат JSON.**

**XmlSerializer (Сериализатор XML):**

**XmlSerializer - это класс в пространстве имен System.Xml.Serialization, предназначенный для сериализации и десериализации объектов в формат XML.**

**BinaryFormatter (Бинарный форматтер):**

**BinaryFormatter - это класс в пространстве имен System.Runtime.Serialization.Formatters.Binary, который позволяет сериализовать и десериализовать объекты в бинарный формат.**

1.3.6 Workin with network(API)

1. System.Net, WebClient

2. API connections, URI

2. HttpClient, HttpMethod

8

1.3.7 MultiThreading, Parallel and

Asynchronous programming

1. MultiThreading

2. Asynchronous Programming

3. Paraller Programming

12

1.3.8 TelegramBotApi

1. Polling vs Webhook

2. SendMessage, SendSticker, SendPhoto

3. ReplyKeyboardMarkup and

InlineKeyboardMarkup

20

1.3.9 Amaliyot va oraliq nazorat Amaliyot va oraliq nazorat 4

2 DATA BASE 80

2.1 PostgreSQL 80

2.1.1

PostgreSQL MOBT

1. Database

2. Database Management Systems (DBMS)

(Postgresql, Oracle)

3. Database types(relational, nosql)

4. PostgreSQL and PgAdmin install

5. Tablespace, Schema, Table, User

4

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

2

2.1

2.1.2

Data types and Constraints

1. Connect To a PostgreSQL Database

2. Server

3. Data types

4. Constraints => NOT NULL, UNIQUE, PK, FK,

CHECK

8

2.1.3 SQL so’rovlar tili, DDL, DML

buyruqlari

1. DDL(CREATE, ALTER, TRUNCATE, DROP)

2. DML(INSERT, SELECT, UPDATE, DELETE) 8

2.1.4

DSL, TCL, Build Structure 1. DCL (Grant, Revoke)

2. TCL(Commit, Rollback, Savepoint)

3. Build Structure(ONE to ONE, ONE to MANY,

MANY to ONE, MANY to MANY)

8

2.1.5

Opertorlar, PostgreSQL fundamentals 1. Operators(Arithmetic, Comparison, Logic)

2. PostgreSQL fundamentals (select distinct, order

by, where, limit, fetch, in, between, like, is null,

table and column aliases)

8

2.1.6

Joining Multiple Tables

1. Joining Multiple Tables (join, cross join, natural

join)

2. AGGREGATE FUNCTIONS (count, avg, max,

min, sum)

3. Grouping Data (group by, having)

8

2.1.7

Extra fundamentals for table 1. Subquery, ANY, ALL, EXISTS, Set

2. Operations (UNION, Intersect, Except)

3. CONDITIONAL EXPRESSIONS &

OPERATORS (case, coalesce, nullif, cast)

4

2.1.8 Viewlar bilan ishlash 1. Views(Simple, Complex, Inline, Materialized)

2. Managing PostgreSQL Views(CREATE, DROP,

UPDATABLE)

4

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

2

2.1

2.1.9 Funksiyalar bilan ishlash 1. Functions(aggregate, math, String, Date)

2. SQL functions

3. PL/PgSql functions

8

2.1.10

Triggers, Lock and Privileges 1. Triggers

2. Crypto

3. Lock

4. Privileges

8

2.1.11

Postgresql Advanced Features

1. JSON, JSONB

2. Indexing

3. Postgres Cron

4. CREATING TABLE PARTITIONS

5. Vertical table

8

2.1.12 Kurs ishi va Oraliq nazorat Kurs ishi va Oraliq nazorat 4

3 .NET Backend 240

3.1 ASP.NET Core Web API Introduction 80

3.1.1 Introduction to ASP.NET Core Web

API

1. Introduction to REST

2. What is ASP.NET Web API?

3. First application

4. Startup va Program filelari

8

3.1.2

Controller, Action, Routing

1. Controller nima?

2. Action bilan ishlash

3. IActionResult, ActionResult

4. Routing nima?

5. Fallback routes

8

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

3

3.1

3.1.3 Middleware, Request pipeline, DI

Container

1. Middleware nima?

2. Request pipeline

3. Dependency Injection

4. DI Container

5. Registering services

8

3.1.4 Global error handling and Logging

1. ILogger interfeysi

2. Serilog configuration

3. Error handling

8

3.1.5 Model binding and validation

1. Model binding

2. Model Validation

3. Handling validation errors

8

3.1.6 CRUD operations

1. Get metodi bilan ishlash

2. Post metodi bilan ishlash

3. Put va Patch metodlari

4. Delete metodi

12

3.1.7 RESTful API with Repository Pattern

1. What is RESTful api?

2. Repository Pattern haqida

3. Implement RESTful api with RP

12

3.1.8 Filters, Sorting, Searching and

Pagination

1. Filtering

2. Sorting

3. Searching

4. Pagination

8

3.1.9 Object mapping

1. DTO nima?

2. AutoMapper

3. Mapster

4

3.1.10 Amaliyot va Oraliq nazorat 4

3.2 ASP.NET Core Web API, WebMVC additional 80

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

3

3.2

3.2.1 OpenApi and Swagger. API Versioning

1. Configure Swagger

2. URL versioning

3. Http Header versioning

4. Deprecating Versions

8

3.2.2 Caching, Rate limiting

1. Cache types

2. Cache-store, Expiration model, Validation model

3. Etag

4. Rate limiting

8

3.2.3 CORS

1. What is CORS?

2. Enabling CORS

3. Setting CORS origins and HTTP methods

4. Creating a custom CORS policy

8

3.2.4 ASP.NET Core MVC

1. MVC haqida

2. Views, Strongly typed views

3. Layout, ViewStart, ViewImport

4. Tag helpers

5. Statik fayllar bilan ishlash

6. Razor view

12

3.2.5 Authentication with JWT

1. Authentication

2. JWT(JSON Web token)

3. Authentication with JWT

12

3.2.6 ASP.NET Identity

1. Create IdentityDbContext

2. Register, Login, Logout

3. Identity services

8

3.2.7 Cookies

1. What is Cookies?

2. Setting and getting cookies 8

3.2.8 Authorization server with

IdentityServer4

1. IdentityServer options

2. Identity resource

3. API scope, API resource, API client

4. Authorization

12

3.2.9 Amaliyot va Oraliq nazorat Amaliyot va Oraliq nazorat 4

3.3 ASP.NET Advanced 80

3.3.1 Coding Standards 1. Clean Code

2. Refactoring 8

3.3.2 Clean Design

1. SOLID

2. DRY

3. GoF

12

3.3.3 Software Testing

1. Unit Testing

2. AAA pattern

3. FIRST pattern

4. TDD

5. Test Controllers

6. Mock tests

12

3.3.4 Deploy to IIS Server

1. IIS Server overview

2. IIS Modules

3. Publish to IIS

8

3.3.5 Azure 1. Azure services

2. Publish and Deploy the app 8

3.3.6 Dockerizing ASP.NET Core app

1. Create dockerfile

2. Docker container

3. Docker image

4. Docker hub

12

3.3.7 Devops 1. What is CI/CD

2. Github actions 8

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

3

3.3

3.3.8 Agile, Jira, Trello

1. Agile

2. Jira

3. Trello

8

3.3.9 Amaliyot va Oraliq nazorat Amaliyot va Oraliq nazorat 4

4 Data Structures and Algorithms 80

4.1 Algorithms 32

4.1.1

Number Algorithms 1. Five Websites for DSA,

2. LeetCode Platform, LeetCode Problems

3. Power of Two

4. Valid Perfect Square

8

4.1.2 String Algorithms 1. Valid Anagram

2. Valid Parentheses 8

4.1.3 Array and Matrix Algorithms 1. Single Number

2. Move Zeros 8

4.1.4

Dynamic Programming

1. Fibonacci Number

2. Iterative Method

3. Recursive Method

4. Recursive-Make it work

5. DP-Make it efficient

6. Memoization-Top Down

7. Tabulation-Top Down

8. Coin Change 2

8

4.2 Data Structures 48

4.2.1 Asymptotic Analysis Asymptotic Analysis 4

4.2.2 ArrayList and LinkedList ArrayList 8

4.2.3 Stack and Queue LinkedList 4

4.2.4 HashTable HashTable 4

PDP ACADEMY PDP ACADEMY PDP ACADEMY

PDP ACADEMY PDP ACADEMY PDP ACADEMY

4

4.2

4.2.5 Tree Tree 8

4.2.6 Trie Trie 4

4.2.7 Graph (Directed) Graph (Directed) 8

4.2.8 Graph (Undirected) Graph (Undirected) 8

5 Amaliyot 80

Tuzuvchi: X. Abduxoshimo

Introduction to ASP.NET Core Web API

1.Introduction to ASP.NET Core

2.ASP.NET Core features.First Application.Startup and Program.

3.ASP.NET Core hosting

4.ASP.NET Core launchSettings.json file

2 ASP.NET Core project structure

1.Configurations

2.Dependency injection and DI container

3.Middleware and Request pipeline

3 Client-Server Relationship. Server side request Handling

1.HTTP Request. Client-Server relationship

2.Controllers and Endpoints

3.Routing 4 REST API structure.Request-Response types

1.API Architectural Styles.Introduction to REST

2.ActionResult & IActionResult. Http status code

3.Model binding 5 Entity Framework Core in ASP.NET Core. Practice with REST API

1.Using Entity framework core in ASP.NET core

2.Http Get and Post 3.Http Put, Patch and Delete 6 Project architectures. Repository pattern 1.Clean architecture 2.Repository pattern and Implementation 7 Logging in ASP.NET CORE 1.ILogger 2.Serilog basics 3.Serilog advanced 8 Data validation in ASP.NET CORE 1.Model Validation 2.Custom validation attribute 3.Fluent Validation 9 Mapping, Global Exception handling. IoC container service lifetime 1. Data Transfer Objects (DTO) 2. AutoMapper 3.Registering services and Service lifetime 4.Global exception handling 10 Sample Project with ASP.NET CORE Web API 1.Build architecture of the project 2.Create services and configure DI container 3.CRUD operations with REST Api Modul 7. ASP.NET Core Web API 1 Caching in ASP.NET CORE 1.Cache Introduction 2.In-Memory Caching 3.Distributed Caching 2 Entity tag and Rate limiters. 1.Entity tag 2.Rate limiting 3 Introduction Json Web Token (JWT) 1.JWT Authentification Introduction 2.Create Access token 4 Validate JWT Token. Refresh tokens 1.Validate JWT Access Token and Authentificate User 2.Refresh Token 5 Role-Permission based access control(RBAC) with JWT 1.What is Role-Permission based Access Control (RBAC) 2.Using RBAC with JWT tokens 3.Role Configuration in JWT 6 Practice with JWT 1.User Configuration in JWT 2.Test RBAC Authorization with JWT 7 Introduction Filters. Filter types in ASP.NET CORE 1.Introduction Filters and Authorization Filters 2.Resource filters 3.Action filters 8 Filters and searching in ASP.NET Core 1.Exception filters 2.Result filters 3.Searching 9 Cross origin resources sharing(CORS) in ASP.NET Core 1.What is CORS? 3.Setting CORS origins, HTTP methods and Headers 4.Creating a custom CORS policy 10 MediatR and CQRS 1.Using MediatR and CQRS 2.Using MediatR Notification Modul 8. ASP.NET CORE Advanced 1 Introduction to ASP.NET Core Web API 1.Introduction MVC 2.Layout, ViewStart, ViewImport, Partial Views 3.Views, Weakly typed vs Strongly typed views 2 Working with files and Tag helpers 1. Tag helpers 2. UseStaticFiles 3. File Upload in MVC 3 CRUD operation and Validation 1. CRUD in MVC ? 2. Validation in MVC 4 Filtering and Pagination 3. Sorting in MVC 3. Data Filtering in MVC 4. Pagination in MVC 5 Working with Cookies 1. What is Cookies? 2. Cookies Settings 6 Real Time application in Blazor. SignalR 1.Introduction to Blazor App 1.SignalR Introduction 2.Build Real Time application in Blazor with SignalR 7 Unit Testing in ASP.NET . XUnit 1.Introduction software testing. What is XUnit? 2.XUnit practice. 8 Microsoft IdentityUser 1.Introduction Microsoft IdentityUser 2.UserIdentity customization. 9 Reflections 1. .Net compiler. What is metadata and assembly? 2.C#-Reflection. 10 Project Release. introduction CI/CD, Docker and Microservice architecture 1.Create custom nuget package 2.What is hosting and domain? Release Project 1.Cloud, CI/CD and Docker. 3.Complete Deploying Project. Using project as Platform 3.Introduction Microservice Architecture. Microservice vs Monolit Identity Server4 1. IdentityServer4 options 2. Identity resource User Identity library and Single Sign on (SSO) 1.Microsoft Identity User Library 2.Google SSO with User Identit