SWE 203 - WEB PROGRAMMING

DR. DENIZ BALTA

ASP. NET CORE MVC INTRODUCTION

WEEK 1

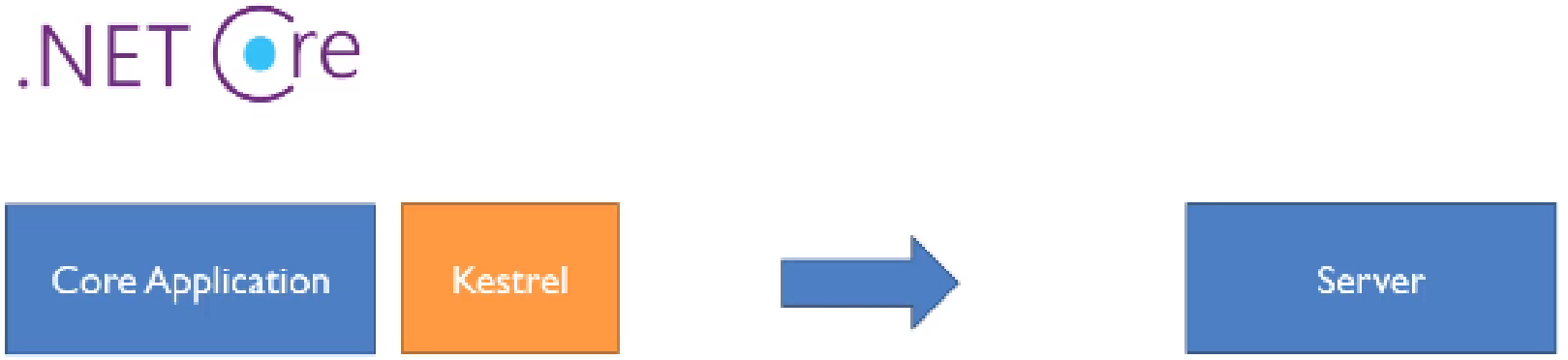
EVALUATION

- **Success rate within the term: %60**
- Midterm : %50
- Quiz 1 : %10
- Quiz 2 : %10
- Homework/Project : %30
- **Success rate end of the term (Final) : %40**

CONTENTS

- .NET CORE
- CORE MVC Fundamentals
- Preparing the development environment

.NET CORE



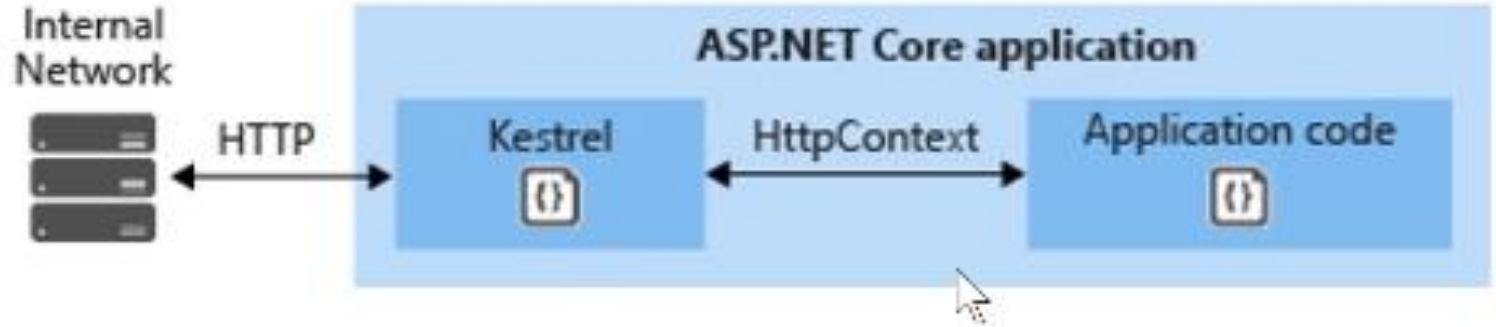
Visual Studio Code

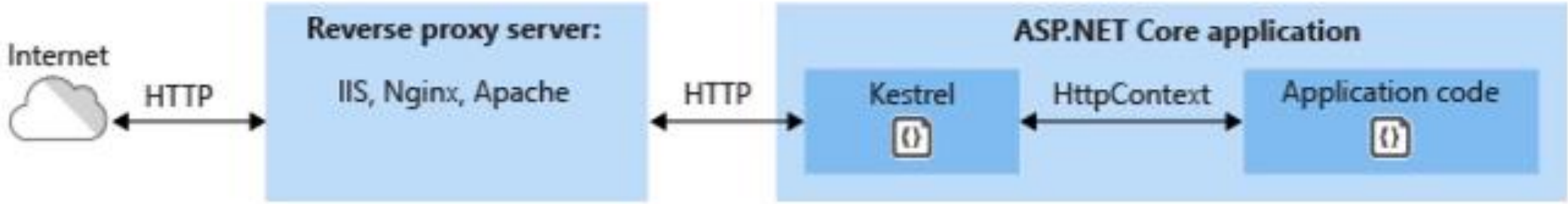
Visual Studio

Microsoft IIS

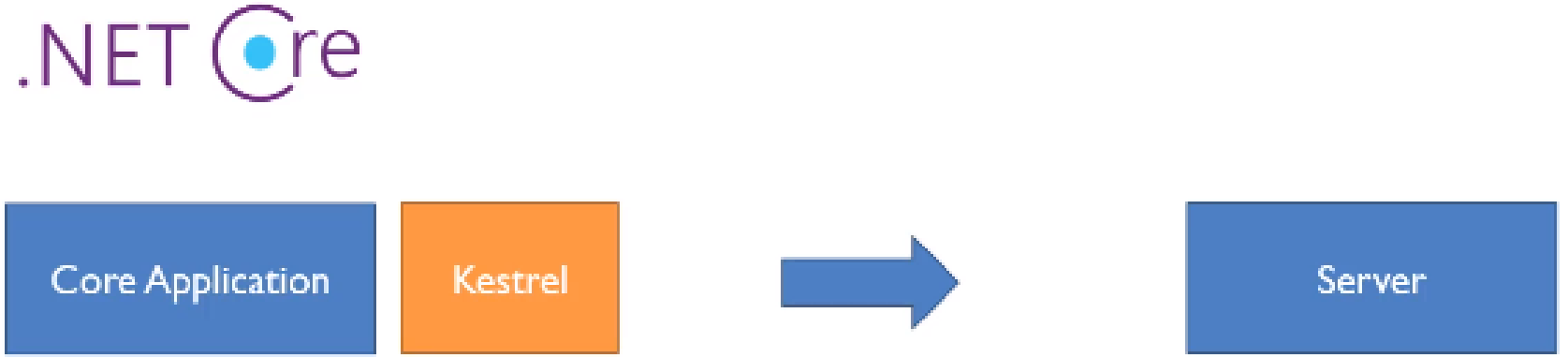
APACHE

.NET CORE


Internal
Network
HTTP
ASP.NET Core application
Kestrel
HttpContext
Application code


Internet
HTTP
Reverse proxy server:
IIS, Nginx, Apache
HTTP
ASP.NET Core application
Kestrel
HttpContext
Application code

.NET CORE



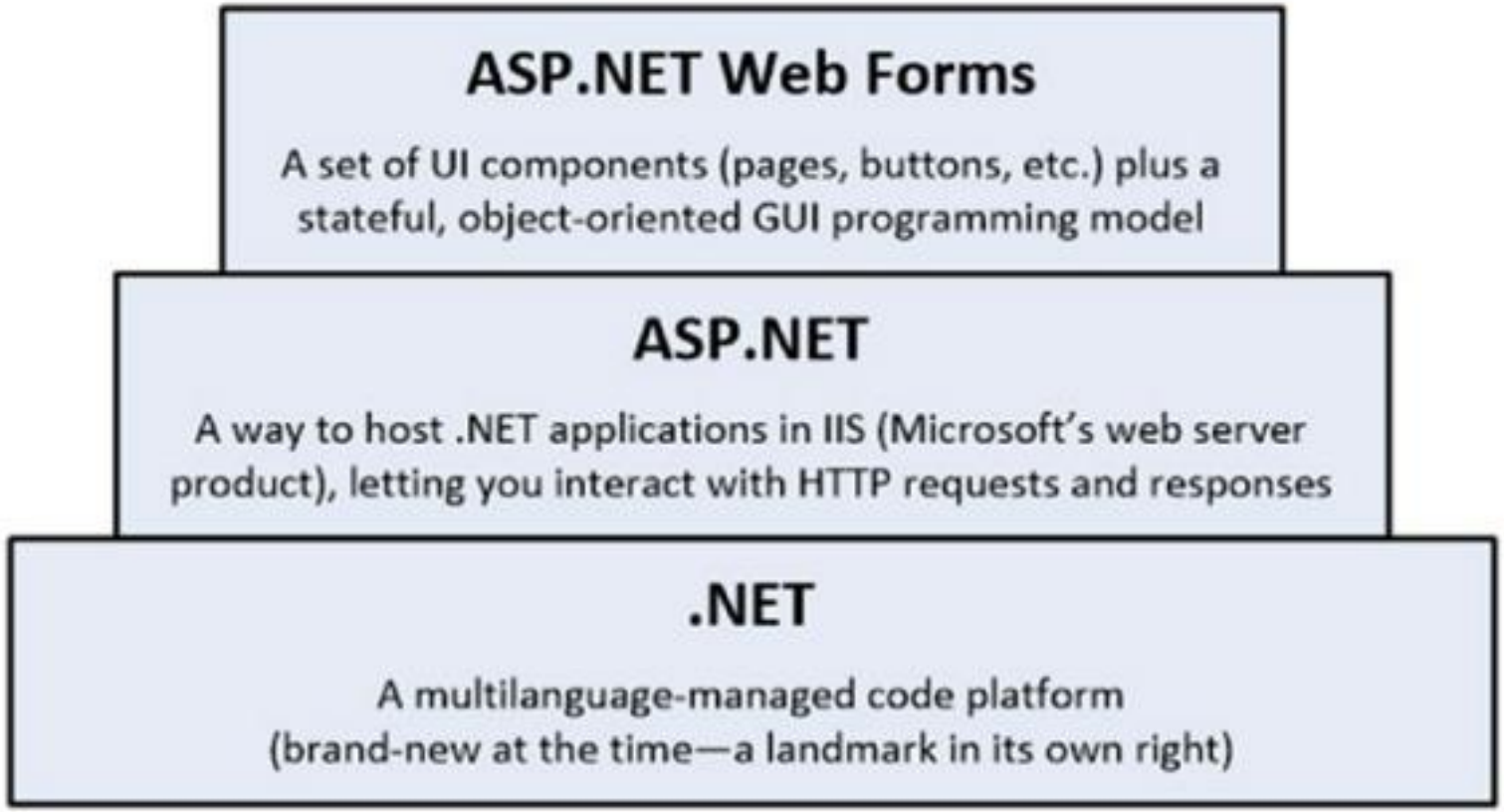
Visual Studio Code

Visual Studio

Microsoft IIS

APACHE

UNDERSTANDING THE HISTORY OF ASP.NET CORE MVC



ASP.NET WEB FORMS

Traditional ASP.NET Web Forms development was good in principle, but reality proved more complicated.

- **View State weight** → to slower response times and increasing the bandwidth demands
- **Page life cycle** →connecting client-side events with server-side event handler code could be complicated and delicate
- **False sense of separation of concerns** → Separate logic and presentation, but developers want to mix presentation code with their application logic in these same code-behind classes.
- **Limited control over HTML** → server controls generated unpredictable and complex ID attributes. These problems have improved in Web Forms, but it can still be tricky to get the HTML you expect.
- **Leaky abstraction** → For implementing custom behaviors, you have to fell out of the abstraction, which forced you to reverse-engineer the postback event mechanism or perform to make it generate the desired HTML.
- **Low testability** → The tightly coupled architecture was unsuitable for unit testing. Integration testing could be a challenge.

THE ORIGINAL MVC FRAMEWORK

- In October 2007, Microsoft announced a new development platform, built on the existing ASP.NET platform.
- The new platform was called the ASP.NET MVC Framework and reflected the emerging trends in web application development, such as
  - HTML and CSS standardization,
  - RESTful web services,
  - effective unit testing,
  - the idea that developers should embrace the stateless nature of HTTP.

WHAT WAS WRONG WITH THE ORIGINAL MVC FRAMEWORK?

- At the time the MVC Framework was created, it made sense for Microsoft to create it on top of the existing ASP.NET platform.
- As the MVC Framework grew in popularity, Microsoft started to take some of the core features and add them to Web Forms.
- The new frameworks added their own configuration and development conventions, each of which had its own benefits and oddities, and the overall result was a fragmented mess.

UNDERSTANDING ASP.NET CORE

- In 2015, Microsoft announced a new direction for ASP.NET and the MVC Framework, which would eventually produce ASP.NET Core MVC.
- ASP.NET Core is built on .NET Core, which is a cross-platform version of the .NET Framework without the Windows-specific application programming interfaces (APIs).
- It is simpler, it is easier to work with, and it is free of the legacy that comes from Web Forms. And, since it is based on .NET Core, it supports the development of web applications on a range of platforms and containers.

KEY BENEFITS OF ASP.NET CORE MVC

- ***MVC Architecture***

ASP.NET Core MVC follows a pattern called model-view-controller, which guides the shape of an ASP.NET web application and the interactions between the components it contains.

- ✓ User interaction with an application that adheres to the MVC pattern follows a natural cycle
- ✓ Combining several technologies

KEY BENEFITS OF ASP.NET CORE MVC

- ***Extensibility***

  - ✓ ASP.NET Core and ASP.NET Core MVC are built as a series of independent components that have well-defined characteristics, satisfy a .NET interface, or are built on an abstract base class. You can easily replace key components with ones of your own implementation.

- ***Tight Control over HTML and HTTP***

  - ✓ Instead of generating out swathes of HTML over which you have little control, ASP.NET Core MVC encourages you to craft simple, elegant markup styled with CSS.
  - ✓ ASP.NET Core MVC works in tune with HTTP. You have control over the requests passing between the browser and server, so you can fine-tune your user experience as much as you like.

KEY BENEFITS OF ASP.NET CORE MVC

- ***Testability***
  - ✓ The ASP.NET Core MVC architecture gives you a great start in making your application maintainable and testable because you naturally separate different application concerns into independent pieces.
  - ✓ In addition, each piece of the ASP.NET Core platform and the ASP.NET Core MVC framework can be isolated and replaced for unit testing, which can be performed using any popular open source testing framework.

Powerful Routing System

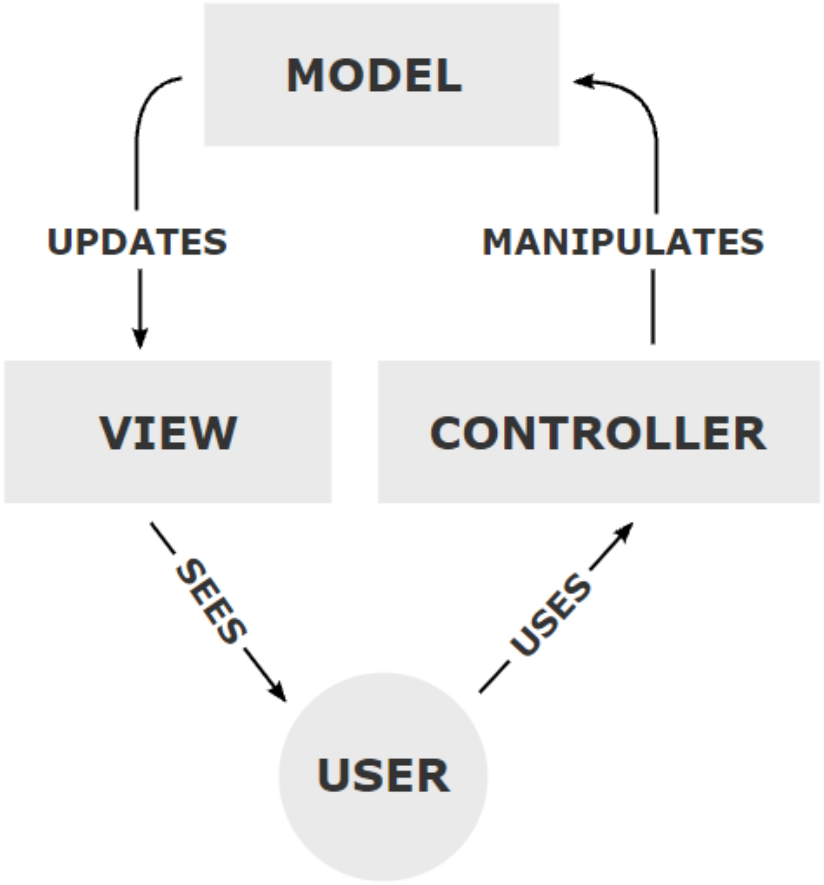
- ✓ Clean URLs were hard to implement in earlier frameworks, but ASP.NET Core MVC uses a feature known as URL routing to provide clean URLs by default.
- ✓ This gives you control over your URL schema and its relationship to your application, offering you the freedom to create a pattern of URLs that is meaningful and useful to your users, without the need to conform to a predefined pattern.

KEY BENEFITS OF ASP.NET CORE MVC

- ***Modern API***
  - ✓ Its API can take full advantage of language and runtime innovations familiar to C# programmers, including the await keyword, extension methods, lambda expressions, anonymous and dynamic types, and LINQ.

- ***Cross-Platform***
  - ✓ Cross-platform support makes it easier to deploy ASP.NET Core MVC applications, and there is good support for working with application container platforms, such as Docker.
  - ✓ Microsoft has also created a cross-platform development tool called Visual Studio Code, which means that ASP.NET Core MVC development is no longer restricted to Windows.

MVC PATTERN



It is an architecture that came out in 1970.

Node.js, Ruby on rails

It separates internal representations of information from the ways information is presented to and accepted from the user.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications.

Popular programming languages have MVC frameworks that facilitate implementation of the pattern.

MVC PATTERN

- ***MODEL***

- The central component of the pattern. It is the application's dynamic data structure, independent of the user interface.
- It directly manages the data, logic and rules of the application.
- Model represents an object or JAVA carrying data.
- It can also have logic to update controller if its data changes.

MVC PATTERN

- ***VIEW***

- Defines how the application's UI will be displayed. It is a pure HTML which decides how the UI is going to look like.
- Any representation of information such as a chart, diagram or table.
- Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

MVC PATTERN

- ***CONTROLLER***

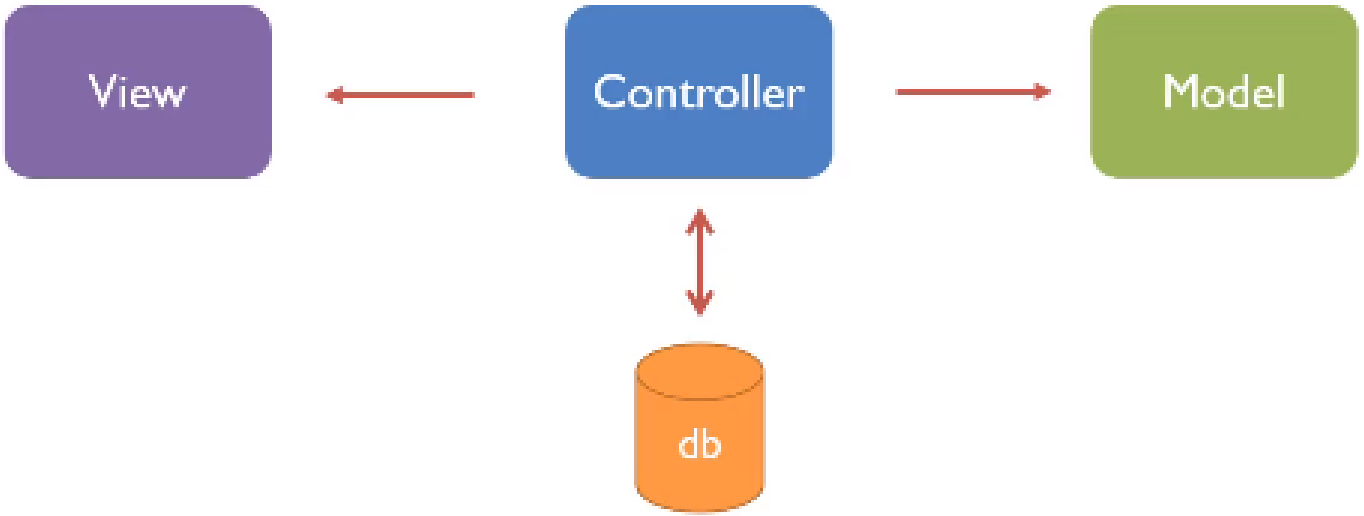
- Accepts input and converts it to commands for the model or view.
- Controller acts on both model and view.
- It controls the data flow into model object and updates the view whenever data changes.
- It keeps view and model separate.
- A set of classes that handles communication from the user, overall application flow, and application-specific logic.

MVC PATTERN

- The idea is that you'll have a component called the view which is solely responsible for rendering this user interface whether it should be HTML or whether it actually should be a UI widget on a desktop application.
- The view talks to a model, and that model contains all the data that the view needs to display.
- In a web application, the view might not have any code associated with it at all.
- It might just have HTML and then some expressions of where to take the pieces of data from the model and plug them into the correct places inside the HTML template that you've built in the view.
- The controller organizes everything. When an HTTP request arrives for an MVC application, the request gets routed to a controller, and then it's up to the controller to talk to either the database, the file system, or a model.

MVC PATTERN

https://seng.sakarya.edu.tr/tr/icerik



FOR AN HTTP REQUEST…

REQUIREMENTS

- Visual Studio 2022
- Visual Studio Code
- PostgreSQL Server (You can download it using Docker Technology)
- DBeaver for using PostgreSQL
- Entity Framework (EF Core Tools)
- Node.js + Angular CLI (from internet download) (cmd)

SETUP COMMANDS

Docker Commands:

- PostgreSQL Docker : (docker)

```
docker run --name PostgreSQL -p 5432:5432 -e POSTGRES_PASSWORD=123456 -d postgres
```

- EF Core Tools Install : (cmd)

```
dotnet tool install --global dotnet-ef
```

CREATE EMPTY PROJECT …