C **web development stacks**

Front End

Back End

2

Logic Tire

1

Data Tire

Nextss

Clint side

rendering

Angler

Blazor

**Server side**

**Rendering**

MVC/blazor

**Database**

“SQL Server”

**Programming language**

“C#”

+

**Framework**

“.Net/ .Net Core”

API

**ORM**

3

Presentation tier

HTML/TCSS

“entity

Framework”

Direct

Access

Visual Studio+SSMS

VSCode

* Client side: also known as frontend, this part of the architecture is responsible for user interaction with the application.
* Database server: this part sends the client’s data to the server.
* Server side: also known as backend storage, this part is responsible for storing the data, processing requests from the database server and sending back the responses.
* how all of them interact with each other, step by step?

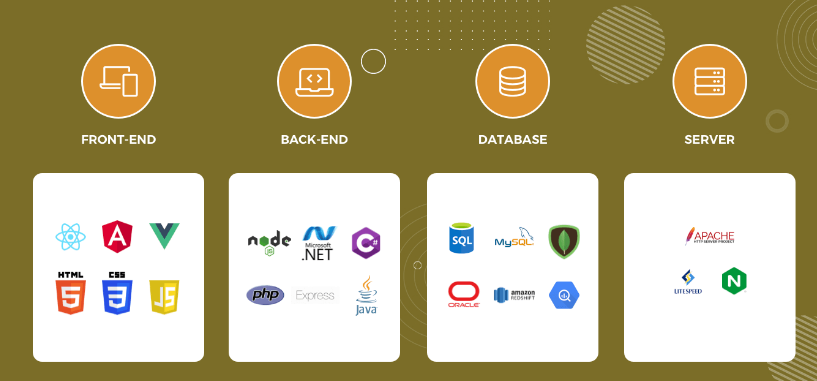
1. A user types in the URL in the browser field.
2. The browser sends the request to the DNS (Domain Name Server) to check whether the request is valid and the IP address is recognized.
3. If successful, the browser then sends the request to the server.
4. The server transfers the request to the data storage, so it locates the page and displays relevant data.
5. The user sees the requested data in the browser.

* **Web app servers**

Web app servers process a user’s’ request and send the docs (usually in JSON, XML, or XMK formats) back to a browser. To do so, web app servers communicate with the backend infrastructure (database, cache server, etc.).

## **Main web application components**

An average app normally features two main types of its components:

1. ****

**Where to use**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Front End | Back End | Database | Server |
| When to use | User interface and experience | Server logic, databases, APIs, and performance of application. | store all the information that the user needs to store | Improving privacy controls  Improving website or app performance  Improving data quality |
| Features | buttons, checkboxes, graphics, and text messages. | databases, authentication/authorization features, application programming interfaces | supports multiple views of data. | deliver dynamic content, like real-time updates, personalized information, and customer support |

1. .Net development tools:
   1. Visual Studio/Visual Studio Code
   2. LINQPad
   3. Stackify Prefix
   4. NDepend
   5. NET Reflector
2. Postman
3. Azure Data Studio
4. JetBrains Rider
5. Git
6. NuGet
7. ReSharper
8. Chocolatey
9. NCrunch
10. NUnit
11. SQL Complete