Hyperiondev

TASK

Back-end Web Development

Visit our website

Introduction

WELCOME TO THE BACK-END WEB DEVELOPMENT TASK!

In this task, we'll dive into back-end development. You will focus on server-side operations and the tools used to build robust web applications. You'll learn about frameworks, databases, and API development, as well as gain crucial knowledge to succeed as a back-end developer. A comparison between front-end and back-end development will also be covered, highlighting their distinct roles and synergies. By the end of this task, you will be well-equipped to describe the key concepts and tools for web application development.

WHAT IS BACK-END WEB DEVELOPMENT?

Back-end development refers to building and maintaining the server-side components of a web or software application. It involves handling the logic, data storage, and information processing that enables an application's functionality. It can be thought of as the 'powerhouse' of a website. It consists of a server, an application, and a database.

Key aspects of back-end development (in no specific order) include server-side programming, web frameworks, APIs (Application Programming Interfaces), databases, integration with third-party applications, security, testing and debugging, deployment of the website, and performance!

RESPONSIBILITIES OF A BACK-END DEVELOPER

A back-end developer is responsible for creating and managing the technology that enables the user interface of a website to function effectively.

Whenever you navigate to a website such as **https://www.amazon.com/**, the site's servers send information to your computer or mobile device, which becomes the page you see. This is the result of the work of a back-end developer. In addition, when you enrolled in your bootcamp, the storage of your personal information was also due to the work of a back-end developer.

Back-end developers use server-side languages, which run on web servers, cloud-based servers, or a hybrid combination, to build an application. Examples of server-side languages are Node.js, PHP, Ruby, Python, Java, and .Net. Back-end developers also use tools like MongoDB, MySQL, Oracle, and SQL Server to find, save, or change data. The data is then served back to the user in the form of front-end code with which the user interacts. Anything you see on a website or an

application is made possible by back-end programming, which exists on and is powered by a **server**. Back-end developers use the above-mentioned tools to create or contribute to web applications with well-documented, clean, and portable code.

A back-end developer takes completed front-end code and makes it functional using **dynamic** programming. For example, a back-end developer can make it possible for values to be populated into a drop-down menu by building the infrastructure that *pulls* values from the database.

The responsibilities of a back-end developer typically include:

- **Database management:** designing, creating, and managing efficient database structures, optimising database queries, and ensuring data integrity and security.
- **Performance optimisation:** optimising server-side code to enhance the website's speed, performance, and scalability.
- **Testing and debugging:** debugging and testing the code and writing code reviews to ensure an error-free experience for the user of the website/application.
- **Frameworks usage:** keeping up with trending new frameworks and technologies.
- **Deployment:** ensuring the integration of cloud computing services and deploying the website to the servers, managing server infrastructure, and ensuring smooth operation of the back-end systems.
- **Server-side programming:** writing code in several programming languages like Python, JavaScript, Ruby, PHP, Node.js, etc., to handle incoming requests and send responses to the server.
- **API integration:** building and maintaining APIs enables communication and data exchange between software systems.
- **Security:** implementing robust security measures such as authentication and data encryption to ensure that websites/applications are not vulnerable to phishing attacks or hacking.
- **Collaboration with front-end developers:** collaborating with front-end developers to integrate both ends of a website/application seamlessly.
- **Version control:** using version control systems like Git to manage code, collaborate with developers, and maintain a well-documented code base.

BACK-END DEVELOPER TOOLS

Programming Languages

Back-end developers may use several languages depending on the needs of particular projects. Some commonly used programming languages are outlined below.

1. **Python**, which supports frameworks such as Flask, Django, and Pyramid, is a popular choice among developers.
2. **Java** enables the construction of enterprise-level systems and supports frameworks like Spring and Java EE.
3. **JavaScript** is a versatile, lightweight, and widely supported programming language that enables developers to build interactive and dynamic web applications.
4. **Ruby** is a popular web application framework that enables rapid development, making it popular among startups.
5. **PHP** is a server-side scripting language that powers content management systems like WordPress.

Frameworks

There are two types of frameworks used by developers. One is a testing framework, and the other is a development framework. Some commonly used development frameworks are outlined below.

1. **Flask** is a Python framework that is favoured by developers for its simplicity, requiring little boilerplate code to start a project.
2. **Django** is a Python framework that has built-in features such as URL routing and administrative interfaces.
3. **Express.js** is a popular framework for building sites with Node.js. It handles routing issues effectively.
4. **Laravel** is a PHP framework that offers an expressive syntax, an ORM for database interactions, built-in authentication, and various tools and libraries.

Some commonly used testing frameworks are outlined below.

1. **Selenium** is an open-source framework widely used for functional testing and regression testing.
2. **Pytest** is a framework for Python that provides an efficient way of writing tests for Python code.

3. **PHPUnit** is a PHP framework to write automated tests for PHP code.
4. **Mocha** is a JavaScript testing framework that allows developers to perform unit tests and organise test suites effectively.
5. **Chai** is an assertion library for JavaScript that works with Mocha, allowing developers to write clear assertions to validate the behaviour of their code.

Databases

The choice of a database depends on the nature of the application and considerations such as performance, scalability, and storage required. The following are some of the most common databases developers use:

- Relational Databases:
    1. MySQL
    2. Oracle
    3. PostgreSQL
    4. Microsoft SQL Server

- NoSQL Databases:
    1. MongoDB
    2. Cassandra
    3. Redis

- Time-series Databases:
    1. InfluxDB

API development tools

Some popular tools that support API design, creation, and development include:

1. Postman assists in the development, designing, and testing of APIs.
2. OpenAPI is a specification and toolset for designing and building RESTful APIs.
3. Insomnia is a platform for testing and debugging APIs.

Servers

There are two main types of servers which will be elaborated on in the next section.

1. Web servers
2. Application servers

These tools are used to create interactive, responsive, and dynamic websites.

SERVERS

To be a full-stack web developer, one needs to know what is happening on the server at the back end of a web application. As mentioned, webservers and application servers are used for the majority of web applications.

Web servers

A web server is a computer with an internet connection and the necessary software stored on it to make web pages available to clients. A web server will store resources associated with web applications, for example, any images, HTML, CSS, and JavaScript files needed to make a web application function. In addition, a web server must have software installed on it that allows it to act as an HyperText Transport Protocol (HTTP) server. An HTTP server (e.g., IIS) is a component that receives HTTP requests and sends the appropriate HTTP responses.

Web browsers use HTTP to communicate with web servers. An HTTP request is sent from your browser to the target web server whenever you click a link on a web page, submit a form, or run a search. Web servers wait for these request messages and then process them when they arrive. The web servers reply to the web browser with an HTTP response message.

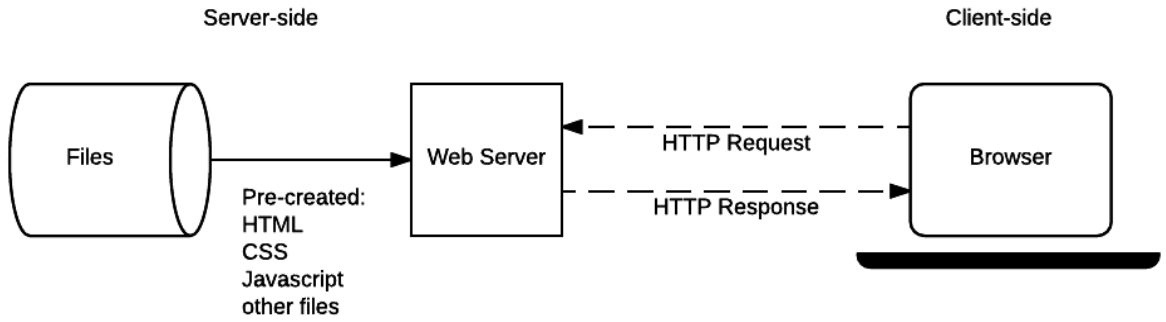
A static website returns the same hard-coded content from the web server whenever a particular resource is requested.

Application servers

Application servers contain the business logic (code) needed to build resources to be dynamically passed back to the browser. An application server will usually have a web framework running on it. For example, a Node application server will usually use Express. The framework can match requests and dynamically generate responses.

Static and Dynamic Applications

If you only want your website to serve static pages, you **only** need a web server as shown in the diagram below.

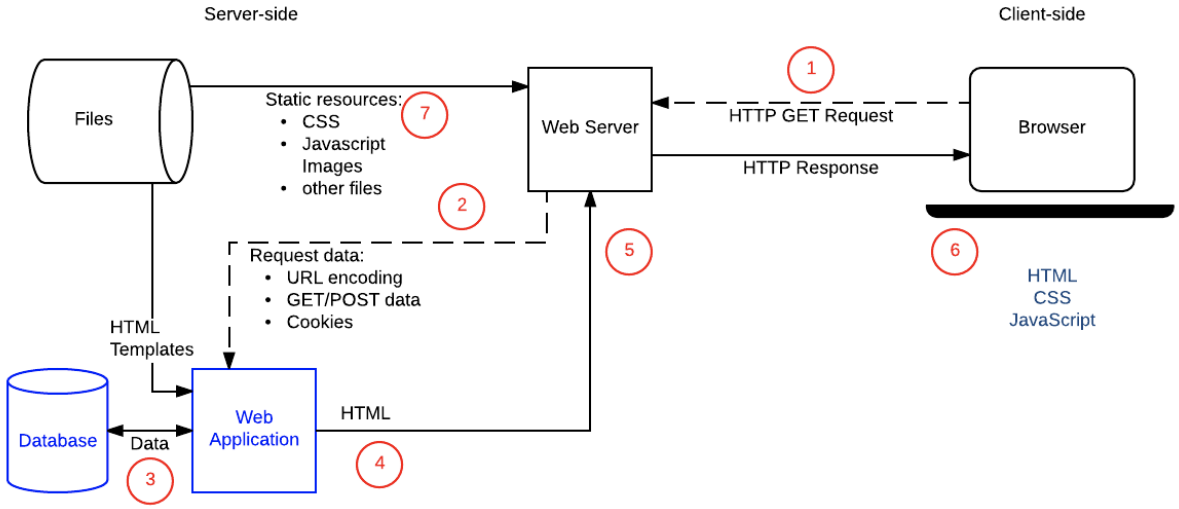


Web Server Architecture for a Static Site

Image Source:

https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

However, if you want to build a web application that responds to requests with dynamic resources, you need a web server and an application server. Review the diagram below to identify the additional elements in a dynamic web application.



Web Server Architecture for a Dynamic Site

Image source:

https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

Typically a reverse proxy sits in front of your web and application servers and routes traffic to the appropriate server. An application server is most likely run as a web server proxy. The web server usually just passes non-static requests to the application server.

Take note:

Here are a few **helpful definitions** from **NGINX**:

"A **proxy server** is a go-between or intermediary server that forwards requests for content from multiple clients to different servers across the Internet."

"A **reverse proxy server** is a proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate back-end server. A reverse proxy provides additional abstraction and control to ensure the smooth flow of network traffic between clients and servers."

FRONT-END DEVELOPER VS A BACK-END DEVELOPER

The following table offers a comparative overview of front-end and back-end developers, highlighting their distinct roles and responsibilities in web development:

Aspect	Front-end developer	Back-end developer
Programming languages	HTML, CSS, and JavaScript	Python, Ruby, Java, PHP, C#, JavaScript.
Focus	User interface (UI) and user experience (UX)	Server-side logic and processing
Frameworks	React, Angular, etc.	Django, Express.js, Flask etc.
Accessibility	Ensures the application is accessible to everyone	Concerned with performance, scalability, and speed
Testing	Conducts UI and cross-border testing	Conducts unit testing, integration testing, and back-end testing
Search Engine Optimisation (SEO) practices	Implements best SEO practices	Less involved in SEO considerations
Performance	Optimises page load speeds	Optimises back-end performances

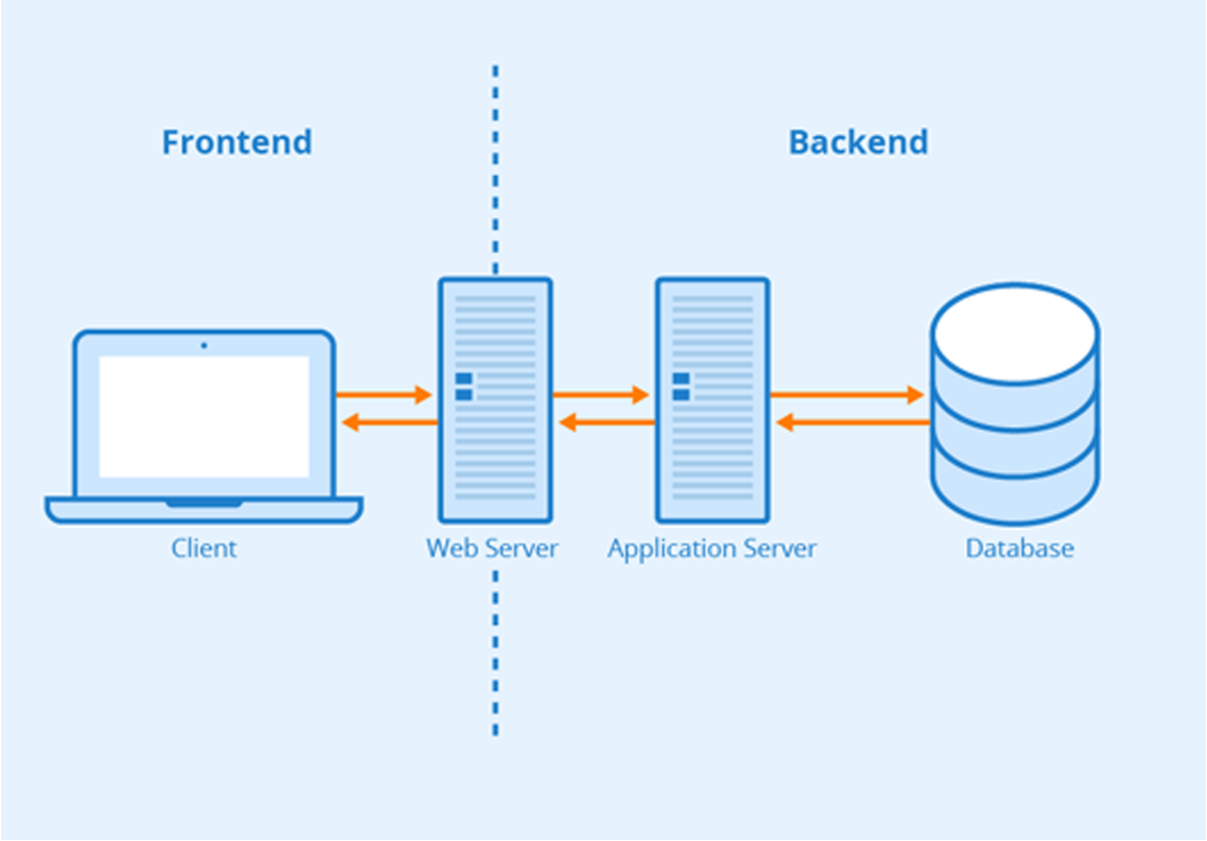


Image source: **TotalVDO Solution** at **http://www.totalvdo.com/back-end-development.html**

The increasing demand for sophisticated web applications makes back-end development a valuable skill to learn. Back-end development enables the functionality, reliability, and performance of web applications ensuring a smooth user experience. In addition, the continuous evolution of technology makes back-end development a rewarding field.

Compulsory Task 1

Use what you have learned in this task, supplemented with some self-study, to **briefly** answer the questions listed below. Leverage keywords from the question to get nuanced answers from Google or your search engine of choice. Submit your answers in a PDF document entitled **BackendWebDev**.

1. Explain the difference between a relational database and a NoSQL database.
2. What is the purpose of using frameworks in back-end development, such as Django, Flask, etc?

3. What are some security considerations when creating a website/application?
4. Explain the role of a web server, application server, and proxy server.
5. Describe the steps involved in handling HTTP requests in a back-end application.

Rate us
Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.