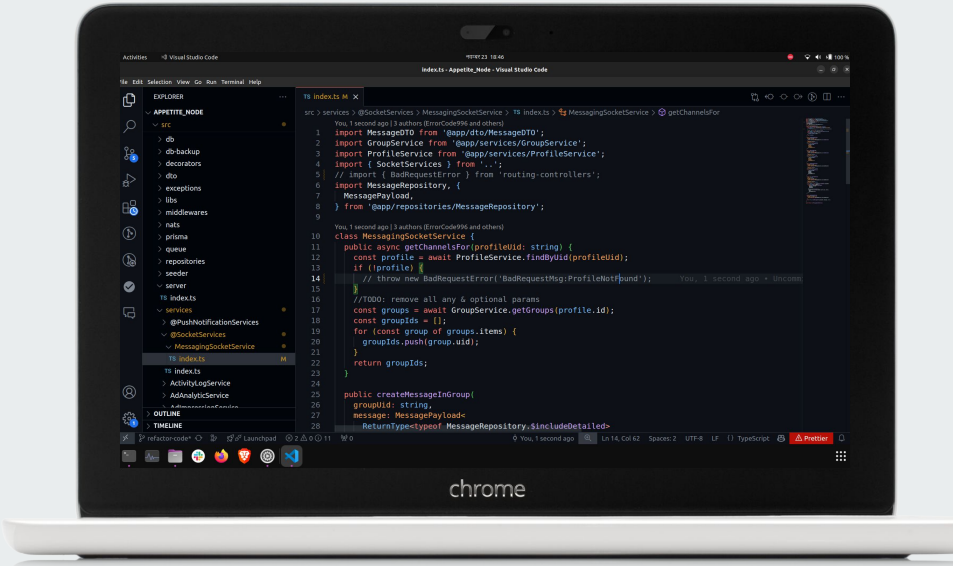
Backend Development

- Sunil Banmala
- Rohit Prajapati



Github Repo Link:
github.com/banmala/todo-api

Outline

Intro to Backend Development

Core Components

Options for Programming Languages

Popular Frameworks

Databases and ORMs

Introduction to Backend Development

01

Parts of Backend Development

- ➔ Providing end points to Frontend
- ➔ Handling business logics
- ➔ Data storage
- ➔ Database Management
- ➔ Handling external services and APIs

Importance of Backend Development

- ➔ Ensuring smooth performance and simplicity in Frontend Development.
- ➔ Ensuring security of applications.
- ➔ Processing user requests
- ➔ Implementing application workflows, and enforcing rules.
- ➔ Storing, retrieving, and updating data efficiently and securely.

Key Skills Required

- ➔ hands-on knowledge of a programming language
- ➔ problem solving skills
- ➔ basic logic implementation
- ➔ patience to debug code

Core Components 02

1. Server
2. Application
3. Database

Server

- ➔ Server is a system that accepts client requests, processes them and responds to the client over a network
- ➔ Provides Services and Resources required to run an backend application.

Server

Key Roles of the server:

- Handle Requests
- Process Business Logics
- Database Communication
- Resource Management
- Security
- Load Balancing

Server in Action

1. A client (e.g., a browser) sends a request: “Get the list of products from the catalog.”
2. The server receives this request and processes it.
3. The server queries the database to fetch the list of products.
4. The server sends the data back to the client in a usable format (e.g., JSON).
5. The client displays the data to the user.

Application

- The server-side program or code that defines core functionality of a server and business logics of a system including all modules of a required backend system.

Roles of Application in Backend Development

- Define and execute rules and workflow of a system
- Receive requests from clients and determines how to respond
- Performs CRUD on database as per system's requirements
- Exposes APIs for the frontend or other systems to interact with the backend.
- Communicates with third-party services
- Ensures secure communication, user authentication

Examples of Application in Backend

- E-Commerce Backend Application
- Social Media Backend Application
- Banking Backend Application
- Streaming Service Backend Application

Components of Application

- Controllers/Endpoints
- Services/Logic Layer
- Database Interaction Layer
- Middleware

Database

- Database is a system used to store, manage, and retrieve data for an application.
- Key Roles:
  - Data Storage
  - Data Retrieval
  - Data Modification
  - Data Relationships
  - Concurrency Management
  - Security
  - Data Backup and Recovery

03

Options of Programming Languages and Their Frameworks

Programming Languages

Factors to Consider:

- Scalability
- Ease of learning
- Nature of the Application
- Development Speed
- Framework Availability
- Library Support
- Community Support

Programming Languages

- Frameworks are:
  - pre-written code, libraries, and tools that provides a structure for developing applications
  - designed to simplify and speed up the development process
  - offers reusable components and a standard way to organize and implement code.

Programming Language

Popular Backend Languages:

JavaScript (Node.js)

- **Why Popular:**
  - Uses a single programming language (JavaScript) for both frontend and backend.
  - Suitable for high-concurrency systems.
- **Best For:**
  - Real-time applications (e.g., chat apps, collaborative tools).
  - RESTful APIs and microservices.
- **Frameworks:**
  - Express.js, NestJS, Fastify, Hapi JS.

Programming Language

Popular Backend Languages (... Contd) :

Python

- **Why Popular:**
  - Simple syntax and readability make it beginner-friendly.
  - Rich ecosystem of libraries and frameworks.
- **Best For:**
  - Data-heavy applications, machine learning integrations, and APIs.
- **Frameworks:**
  - Django (full-stack), Flask (lightweight), FastAPI (modern, high-performance).

Programming Language

Popular Backend Languages (... Contd) :

Java

- **Why Popular:**
  - Known for its stability, scalability, and performance.
  - Widely used in enterprise applications and financial systems.
- **Best For:**
  - Complex backend systems, API services, and large-scale applications.
- **Frameworks:**
  - Spring Boot, Micronaut, Quarkus.

Programming Language

Popular Backend Languages (... Contd) :

PHP

- **Why Popular:**
  - Traditionally used for web development with excellent integration with HTML.
  - Powerful for server-side scripting.
- **Best For:**
  - Content management systems (e.g., WordPress), small to medium-scale applications.
- **Frameworks:**
  - Laravel, Symfony, CodeIgniter.

Programming Language

Popular Backend Languages (... Contd) :

Ruby

- **Why Popular:**
  - Focuses on developer happiness and productivity.
  - Clean syntax and strong convention-over-configuration approach.
- **Best For:**
  - Startups, Minimum Viable Products, and rapid application development.
- **Frameworks:**
  - Ruby on Rails (Rails).

Programming Language

Popular Backend Languages (... Contd) :

.NET Core

- **Why Popular:**
  - Supported by Microsoft and widely used in enterprise and cloud-based solutions.
  - Cross-platform support with .NET Core.
- **Best For:**
  - Windows-based applications, gaming (Unity backend), and APIs.
- **Frameworks:**
  - ASP.NET Core, Blazor.

Programming Language

Popular Backend Languages (... Contd) :

Other Backend Application Development are:

- **Go (Golang)**
  - **Frameworks: Gin, Echo, Fiber.**
- **Kotlin**
  - **Ktor, Spring Boot (Kotlin-compatible).**
- **Rust**
  - **Actix, Rocket.**

Programming Language

Popular Backend Languages (... Contd) :

Other Backend Application Development are:

- **TypeScript (with Node.js)**
  - **NestJS**
- **Elixir**
- **Scala**
- **Dart**

Database and ORMs
04

Database

- It is a structured system used to store, manage, and retrieve data for applications
- Backbone of almost backend system

Database

Types of Databases

1. Relational Databases (SQL):
    a. Use structured data stored in tables with rows and columns.
    b. Examples: MySQL, PostgreSQL, SQLite.
2. Non-Relational Databases (NoSQL):
    a. Store unstructured or semi-structured data, such as JSON or key-value pairs.
    b. Examples: MongoDB, Cassandra, Redis.

ORM (Object-Relational Mapping)

- A tool that allows developers to interact with a relational database using object-oriented programming paradigms, rather than writing raw SQL queries.
- For Eg:
  - SQL: SELECT * FROM student WHERE roll_no = 5 LIMIT 1;
  - Prisma ORM: prisma.student.findFirst({ where: { roll_no: 5 } });

ORM (Object-Relational Mapping)

Benefits of Using ORMs

1. Abstraction of SQL
2. Reduces boilerplate code, allowing developers to focus on application logic.
3. Makes it easier to switch between databases without rewriting queries.
4. Enforces consistency in how the database is accessed and modified.
5. Helps prevent SQL injection by parameterizing queries.

Database

Popular Databases are:

- Relational Databases
  - MySQL
  - PostgreSQL
  - Microsoft SQL Server
  - SQLite
- Non Relational Databases (No SQL)
  - MongoDB
  - Firebase
  - Redis

ORM (Object-Relational Mapping)

Choose an ORM based on:

- Language compatibility with your application.
- Database support for your project.
- Specific features or requirements such as :
  - performance
  - scalability
  - ease of use

ORMs

Popular ORMs supporting different databases and programming languages:

Django ORM (Python)

- **Supported Databases:** PostgreSQL, MySQL, SQLite, Oracle.

Sequelize (Javascript / Typescript)

- **Supported Databases:** MySQL, PostgreSQL, SQLite, SQL Server.

Hibernate (Java)

- **Supported Databases:**PostgreSQL, MySQL, SQLite, Microsoft SQL Server

ORMs

Popular ORMs (... Contd):

Sequelize (JavaScript/TypeScript):

- **Supported Databases:** MySQL, PostgreSQL, SQLite, Microsoft SQL Server

Prisma (JavaScript/TypeScript):

- **Supported Databases:** PostgreSQL, MySQL, SQLite, MongoDB

Mongoose (JavaScript/TypeScript):

- **Supported Databases:** MongoDB (document database)

Doctrine (PHP):

- **Supported Databases:** MySQL, PostgreSQL, SQLite, Oracle

Next What?

- Different type of architectures for different project requirements
- Git for codebase management and development with collaboration
  Git, Github, Gitlab, Bitbucket
- Deployment of the backend Project
  - Node, Apache, Nginx
- Destructuring of big codebase into smaller modules
- Using external APIs to communicate with third party backends
  - Payments
  - User Authentication

Questions?