# Backend Developer

is a software developer who is responsible for building and managing the server-side logic of an application or website. They focus on the “behind the scenes” part of an application, which includes managing databases, servers, APIs, application architecture, security and updates. Backend developers ensure that data is processed correctly on the server and sent to the frontend (the part that is visible to the user) efficiently.

### **What does a Backend Engineer do?**

* API Development: Design and develop RESTful or GraphQL APIs.
* Database Management: Create, optimize, and manage databases.
* Server Management: Handle server infrastructure, including security and scalability.
* Integrating External Services: Work with third-party APIs and services.
* Optimize Backend Performance: Ensure the system can handle high traffic efficiently.
* Security: Implement security measures like authentication and authorization.
* Collaborate: Work closely with frontend engineers to integrate APIs with the user interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programming Languages** | **Frameworks** | **Object Relational Mappings** | **DBMS** | **Description** |
| C# | ASP.NET Core | Entity Framework Core | PostgreSQL | Relational Database |
| Javascript | ExpressJS, Adonis | KnexJS, Sequelize | MongoDB | NoSQL Database |
| PHP | Laravel, Codeigniter | Eloquent | Redis | Database In-Memory |
| Golang | Fiber, Echo, Gin | GORM | Schema Design |  |

|  |  |  |
| --- | --- | --- |
| **VCS** | **DBMS Operations** | **OS & Networking** |
| GIT Basics | CRUD, Joins & Aggregations | HTTP/HTTPS |
| Branching & Merging | Transactions | Domain Name System |
| Github/Gitlab | Normalization | TCP/IP & WebSockets |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Web Servers** | | **API Design** | | **Authe. & Autho.** | | **Security** |
| Nginx | REST API | | JWT | | HTTPS & SSL | |
| Apache | GraphQL | | OAuth 2.0 | | CORS, CSRF & XSS Protection | |
|  | gRPC | | Sessions & Cookies | | Input Validation & Sanitization | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **App Architecture** | **Performance Opt.** | **DevOps Practice** | **Monitoring & Logging** | **Web Security** |
| MVC | Caching (Redis, Memcached) | Docker & Orchestration | Monitoring (Prometheus, Grafana) | OWASP Top 10 |
| Microservices | Load Balancing | CI/CID Pipelines | Logging (ELK Stack) | Data Encrypt. (AES, RSA) |
| Event-Drive Archi | Content Delivery Network | Cloud Computing |  |  |

### Essential Soft Skills

* **Problem Solving** : Be able to break down complex problems and find efficient solutions.
* **Communication** : Collaborate with frontend developers, designers, and clients.
* **Patience** : Debugging and troubleshooting server-side issues requires persistence.
* **Teamwork** : Work effectively in teams, especially in agile development environments.
* **Adaptability** : Continuously learn new tools, frameworks, and best practices.

### What’s Next?

* **Build Your Portfolio** : Showcase your backend projects, focusing on API development and scalable architectures.
* **Freelance or Internship** : Gain real-world experience by working on backend projects or interning at a tech company.
* **Contribute to Open Source** : Work on open-source projects to improve your skills and contribute to the community.
* **Learn DevOps** : Get familiar with DevOps practices like continuous deployment, monitoring, and infrastructure as code.

### Backend Developer Interview Questions

* **API Design** :
  + How do you design a scalable and secure RESTful API?
  + What are the differences between REST and GraphQL?
* **Database Management** :
  + How would you optimize a slow SQL query?
  + Explain database normalization and denormalization. When would you use each?
* **Authentication and Security** :
  + How would you implement JWT-based authentication in an API?
  + What is cross-site scripting (XSS), and how would you prevent it?
* **Server-Side Logic** :
  + Explain the MVC architecture.
  + How do you handle background tasks (e.g., sending emails or processing data)?
* **Caching** :
  + How would you implement caching in a web application?
  + Explain the differences between Redis and Memcached.
* **System Design** :
  + How would you design a system to handle millions of requests per second?
  + What strategies would you use to scale a backend service?