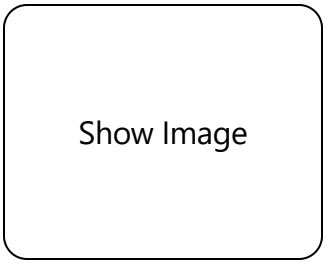
Web Application Architecture - Shortnotes

Website vs. Web Application

- **Website**: Provides static content for end users
- **Web Application**: Designed for interaction with end users

Three-Tier Architecture



Key Benefits

- Each tier runs on its own infrastructure
- Tiers can be developed independently by different teams
- Each tier can be updated and scaled independently
- Better security due to isolation of data and logic from presentation

Presentation Layer (Frontend)

Technologies

- HTML/CSS/JavaScript
- React.js
- Angular
- Vue.js
- jQuery and AJAX (legacy)

Architectural Patterns

- **Model-View-Controller (MVC)**
  - Model: Data and business logic
  - View: User interface
  - Controller: Handles user input, updates model/view
- **Model-View-ViewModel (MVVM)**

    - Similar to MVC but with data binding between View and ViewModel
- Others: MVP, Component Architecture, Micro frontends

Application Layer (Backend)

Technologies

- **Programming Languages/Frameworks**:
    - Java (Spring/Spring Boot)
    - Node.js (Express.js/Koa)
    - Python (Django/Flask)
    - PHP (Laravel, Symphony)
    - .NET Framework
- Virtual Machines (VMs)
- Serverless Computing
- Containers

APIs (Application Programming Interfaces)

- Contract of services offered between applications
- **Benefits**:
    - Reuse existing functionality
    - Reduce development costs
    - Improve collaboration and connectivity
- **Types**:
    - REST API
    - SOAP API
    - RPC (Remote Procedure Calls)
    - WebSockets
    - GraphQL APIs

Architectural Patterns

- **Monolithic Architecture**
    - Single-tiered application with all components in one codebase
    - **Drawbacks**:

        - Slower development speed
        - Can't scale individual components
        - Lower reliability (single point of failure)
        - Barrier to adopting new technologies
        - Lack of flexibility
        - Full redeployment required for small changes

- **Microservices Architecture**
    - Application composed of many loosely coupled, independently deployable services
    - **Benefits**:
        - Independent deployment
        - Independent scaling
        - High reliability
        - Supports polyglot programming
    - **Challenges**:
        - High global complexity
        - High infrastructure costs
        - Debugging challenges

Architectural Anti-Pattern: Big Ball of Mud

- Unstructured, tangled codebase without clear architecture

Supporting Technologies

- API Gateways
- Content Delivery Networks (CDN)
- Caching Tools (e.g., Redis)
- Load Balancers
- Message Queues
- Cloud Computing and Storage
- Virtual Machines

Interesting Topics for Further Reading

- Component-Based Architecture

- Micro Frontends
- Backend For Frontend (BFF)