生成AI基礎

4 Webアプリケーションの概要

Webアプリケーションとは何か

Webアプリケーションの基本的な定義

- インターネット上で動作するアプリケーション
- Webブラウザを介してユーザーとやり取りする
- サーバー上で処理を行い、結果をユーザーに返す
- プラットフォームに依存しない

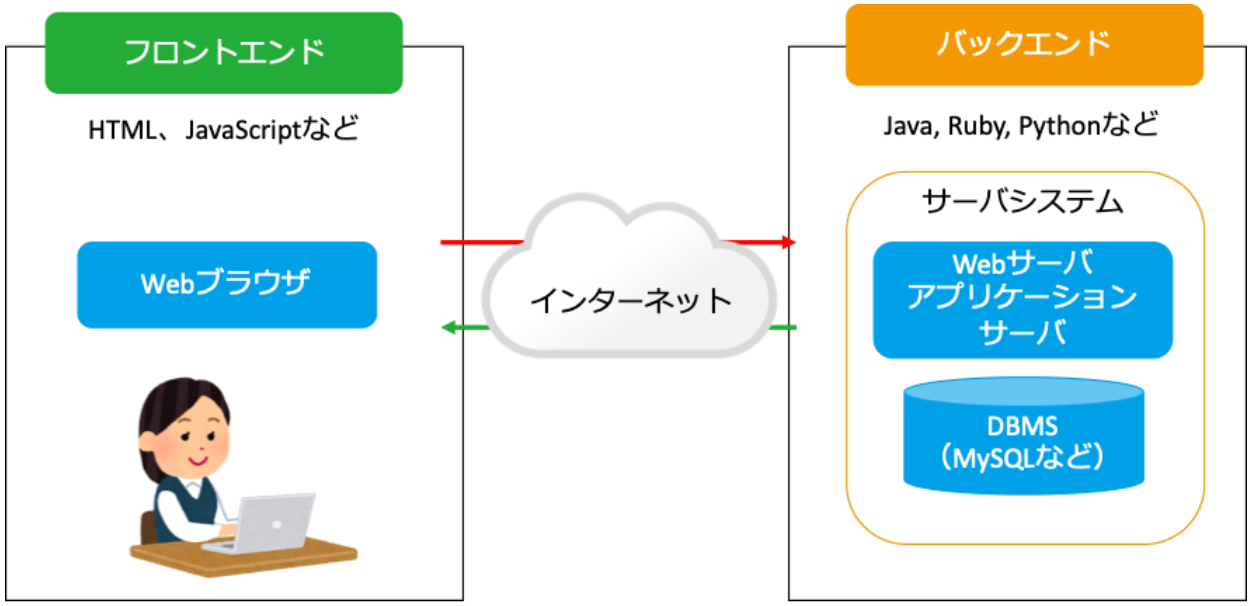
Webアプリケーションの例

- ECサイト（Amazon、楽天など）
- SNS（Facebook、Twitterなど）
- クラウドサービス（Google Drive、Dropboxなど）
- 予約システム（ホテル、飛行機など）
- オンラインバンキング

基本的な要素の説明

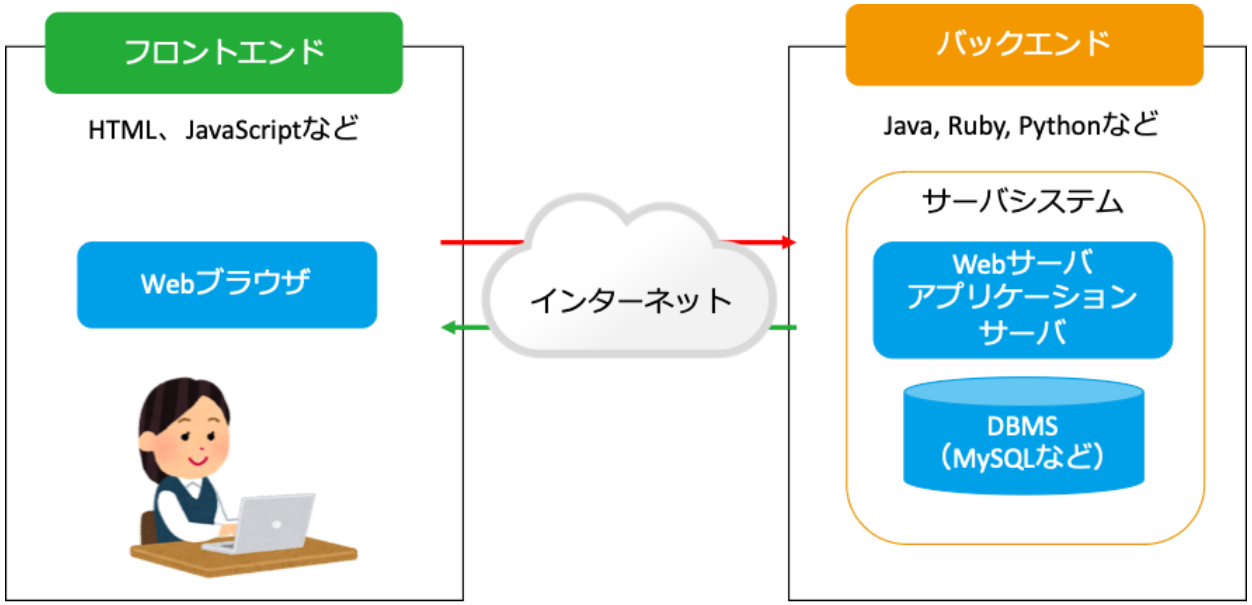
フロントエンドとバックエンド

- フロントエンド：ユーザーが直接操作する部分（HTML、CSS、JavaScript）
- バックエンド：サーバー側で処理を行う部分（Java、Python、PHPなど）
- 役割分担により、開発効率と保守性が向上



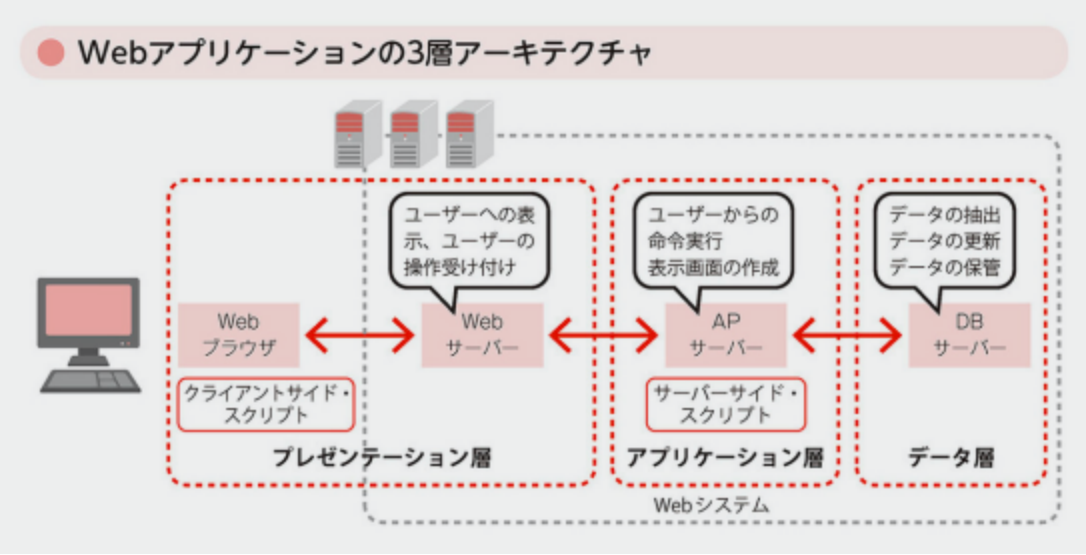
アプリケーションサーバーとデータベースの役割

- アプリケーションサーバーサーバー：クライアントからのリクエストを処理し、レスポンスを返す
- データベース：データの永続化と管理を行う
- サーバーとデータベースが連携してアプリケーションを動作させる



Web3層構造

- プレゼンテーション層（ユーザーインターフェース）
- アプリケーション層（ビジネスロジック）
- データ層（データの保存と管理）
- 各層が独立しており、役割分担が明確
- 変更の影響を最小限に抑えられる



Webアプリケーションの開発プロセス

要件定義

- 目的と対象ユーザーを明確にする
- 機能要件と非機能要件を洗い出す
- 優先順位を決定する
- ステークホルダーとのコミュニケーションが重要
- 要件変更に柔軟に対応できる体制を整える

設計

- システムアーキテクチャを決定する
- データベース設計を行う
- UIデザインを作成する
- APIの仕様を定義する
- セキュリティ対策を考慮する

実装

- 設計に基づいてコーディングを行う
- フレームワークやライブラリを活用する
- コーディング規約を順守する
- コードレビューを実施し、品質を担保する
- バージョン管理システムを利用する

テスト

- 単体テスト、結合テスト、システムテストを実施する
- 自動化テストを導入し、効率化を図る
- テスト計画を作成し、網羅性を確保する
- バグを早期に発見し、修正する
- テスト結果を分析し、改善につなげる

デプロイ

- 本番環境へのリリース手順を定義する
- CI/CDパイプラインを構築する
- ロールバック方法を確立する
- モニタリングツールを導入し、運用状況を監視する
- 障害対応手順を整備する

運用・保守

- ユーザーからのフィードバックを収集し、改善に活かす
- 定期的なパフォーマンスチューニングを実施する
- セキュリティパッチの適用と脆弱性対策を行う
- 運用コストの最適化に努める
- 技術的負債を認識し、計画的に対処する

Webアプリケーションのセキュリティ

ログインとユーザー権限

ログイン:

- ログインとは、コンピューターやオンラインサービスにアクセスする際に、自分のアカウントを識別するプロセスです。
- ユーザー名とパスワードを入力することで、システムはあなたが登録済みのユーザーであることを確認します。
- ログインすることで、自分のアカウントに関連付けられた情報やサービスを利用できるようになります。

ログインとユーザー権限

ユーザー権限:

- ユーザー権限とは、システム内でユーザーができる操作の範囲を定義したものです。
- 権限には、読み取り、書き込み、変更、削除などがあります。
- 管理者は通常、システム内のすべての操作を行う権限を持ちます。
- 一般ユーザーは、管理者によって割り当てられた権限の範囲内で操作を行うことができます。
- 権限の設定は、システムのセキュリティと適切な運用を維持するために重要です。

ログインとユーザー権限

例えば、学校の成績管理システムでは、以下のような権限設定が考えられます:

- 生徒は自分の成績を閲覧することはできますが、変更することはできません。
- 先生は生徒の成績を入力・変更する権限を持ちます。
- 学校の管理者は、システム全体の管理権限を持ち、必要に応じて先生や生徒のアカウントを作成・削除できます。

HTTPとHTTPS

- HTTPとHTTPSは、インターネット上でWebサイトにアクセスする際に使用される通信プロトコル（通信規約）。
- HTTP（Hypertext Transfer Protocol）は、WebブラウザとWebサーバー間でデータをやり取りするための基本的な方法。
  - HTTPではデータが暗号化されずに送信されるため、第三者に傍受されるリスクがある。

HTTPとHTTPS

- HTTPS（HTTP Secure）は、HTTPにSSL/TLS暗号化を追加したもの。
  - Webブラウザと Webサーバー間の通信が暗号化され、データの機密性と完全性が保護される。
  - オンラインショッピングでクレジットカード情報や個人情報を送信する際、HTTPSを使用することで情報が保護される。
- 現在、ほとんどのWebサイト、特にオンラインバンキングやショッピングサイトなどの機密性の高いサイトでは、HTTPSが使用されている。
- WebブラウザのアドレスバーにURLを入力する際、「https://」から始まるURLを使用することで、そのサイトがHTTPSを使用していることがわかる。

脆弱性とその対策

- 脆弱性（ぜいじゃくせい）は、コンピュータのOSやソフトウェアにおけるセキュリティの欠陥で、プログラムのバグや設計ミスにより生じます。
- これらの脆弱性を放置すると、不正アクセスやウイルス感染のリスクが高まります。
- 脆弱性が発見された場合、ソフトウェアメーカーは通常、更新プログラムをリリースしますが、新しい脆弱性が次々に発見されるため、完全な対策は難しいです。
- 特に、インターネットに公開されたサーバは、脆弱性を悪用されることで、サイトの改ざんやウイルスの発信源にされる危険があります。

まとめ

- Webアプリケーションの概要
- Webアプリケーションの開発プロセス
- Webアプリケーションのセキュリティ