

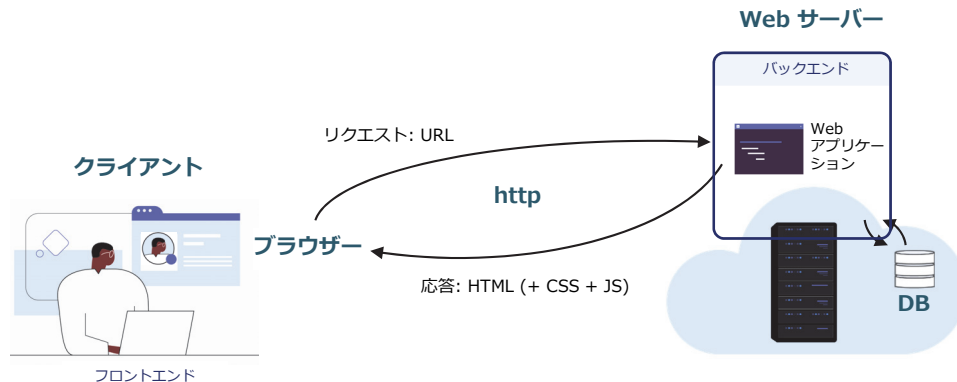
Web アプリケーションと モバイルアプリケーションの アーキテクチャ

類似点と相違点

GeneXusTM

アプリケーションの仕組みを理解するには、そのアーキテクチャについて理解する必要があります。
これから、Web アプリケーションとモバイルアプリケーションのアーキテクチャについて概説します。

従来型 Web アプリケーション (TWA)



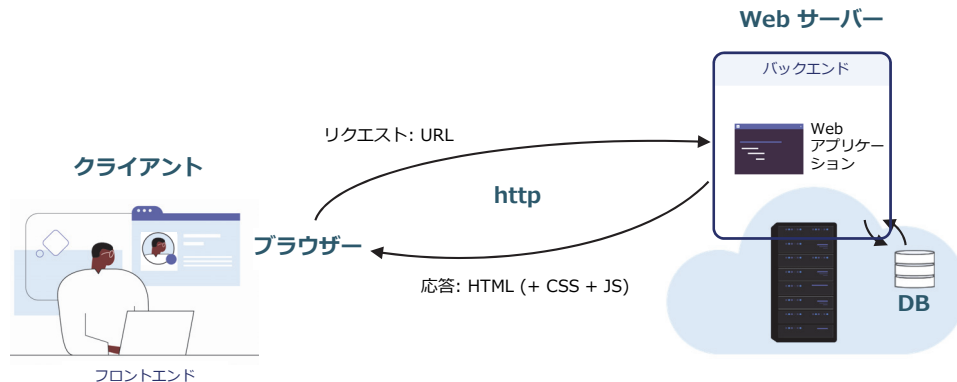
最初に従来型の Web アーキテクチャを見ていきます。
Web アプリケーションには、インターネットブラウザ (クライアント) で実行される部分と、Web サーバー (サーバー) で実行される部分があります。前に説明したとおり、ブラウザで実行される部分はアプリケーションのフロントエンド、サーバーで実行される部分はバックエンドといいます。

ブラウザに Web アドレス (URL) が入力されると、その URL に対応するリソースにアクセスするため、ブラウザからサーバーへ、HTTP プロトコルを使用してリクエストが送信されます。Web サーバーは、リクエストされたリソースのアドレスを見つける役割を担います。リソースは、単一の Web ページか、Web アプリケーションのホームページです。

サーバーはその後、必要に応じてデータベースにアクセスし、HTML コードで応答を組み立てます。HTML は、Web ページのコンテンツの体裁を整えるための言語です。この応答が、リソースをリクエストしたクライアントブラウザに送られます。

ブラウザで HTML コードが解釈され、リクエストされたページの画像、ボタン、メニュー、コントロール、情報が画面に表示されます。書式や機能は、リクエストされたページまたは Web アプリケーションのプログラミング内容に従います。ブラウザでは既定で、HTML エLEMENTがネイティブスタイルで表示されます。つまり、ボタン、テキストボックス、その他のオブジェクトは、使用されているオペレーティングシステムとブラウザに従って表示されます。こうしたエレメントのルック & フィールをカスタマイズするために、HTML には CSS (カスケードスタイルシート) のコードを含めることができます。CSS (最新バージョンは CCS3) は、フォントの色やタイプ、背景色、余白など、HTML ドキュメントのスタイルの定義に使用するグラフィックデザイン言語です。CSS は現在、Web ページにアニメーションやその他の高度な効果を追加できるよう進化しています。

従来型 Web アプリケーション (TWA) (続き)



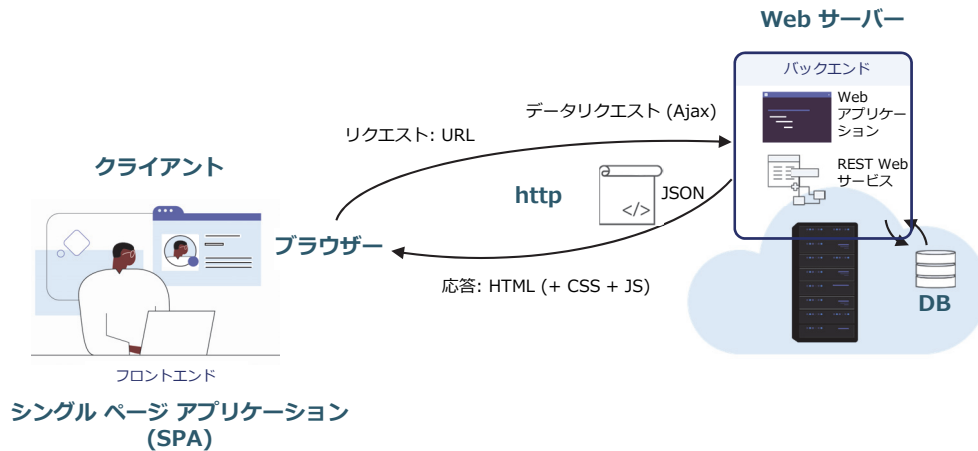
HTML コードには JavaScript コードを含めることもできます。JavaScript を使用すると、ブラウザに表示される Web ページに動きを加え、双方向の操作、優れたユーザーエクスペリエンスを実現できます。JavaScript プログラミング言語により、データのリクエスト、ユーザーイベントへの応答、表示コンテンツの迅速な更新、マップとの連携、2D/3D グラフィックアニメーションの作成、マルチメディアファイルの再生などが Web ページで可能になります。

最近では、TypeScript 言語を使用する Web アプリケーションが増えています。これは Microsoft が開発したオープンソースのプログラミング言語であり、モジュール化、静的なデータのタイプやクラスを加えて JavaScript 言語を拡張することができます。

TypeScript は JavaScript のスーパーセットであるため、より多くの機能がありますが、JavaScript に基づいていて、最終的なコードを一般的な JavaScript に変換します。この方法により、ブラウザで JavaScript コードを実行できる場合、元のコードが TypeScript で記述されていることは認識されず、JavaScript 言語が実行されます。現在、Angular 2 フレームワークの新バージョンが TypeScript で開発されています。

このアーキテクチャは広く普及し、特に Web ページで使用されています。これに対して、Web アプリケーションでは、より優れた双方向性やユーザーエクスペリエンスを実現できる新技術が注目されています。

シングル ページ アプリケーション (SPA)



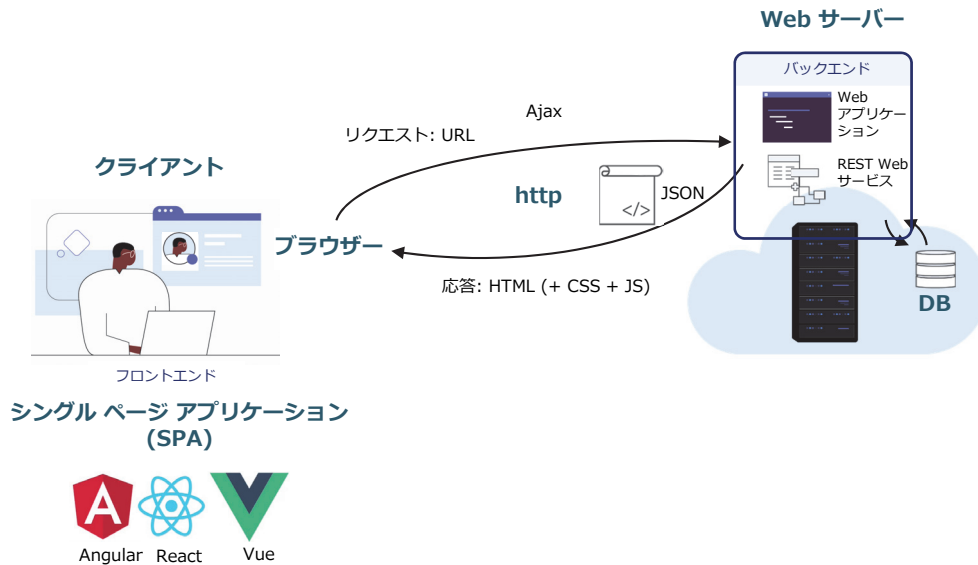
Web アプリケーションの要件として、画面上に大量のコンテンツを表示し、双方向性を高め、従来型 Web アプリケーションよりも応答時間を短くする必要があるときは、シングル ページ アプリケーションという別のアーキテクチャを使用します。この場合、アプリケーションがブラウザに送信されてから、使用中はリロードされないため、デスクトップアプリケーションと同様のユーザーエクスペリエンスが実現します。

これは、HTML コードの生成を含め、ほとんどの機能がクライアントで処理されることを意味します。Web アプリケーションはシングルページで実行され、リソースはユーザー操作に応じて Web サーバーから動的にロードされます。サーバー側のコードにはサービスが含まれます。つまり、データベースからのデータ取得など、クライアントからのリクエストに応じるプログラムです。アプリケーションロジックはクライアント側にあります。

ブラウザからサーバーに URL がリクエストされると、書式設定やスタイルが適用された HTML コードのページが応答として返されます。この部分は、従来型 Web アプリケーションと同様です。しかし、その後、クライアントは Ajax を使用し、対応する API を呼び出してサーバーにデータをリクエストします。これにより、データベースからデータが取得され、情報が JSON 形式でクライアントに返されます。

この方法で、ページのリロードが回避されるため、パフォーマンスやユーザーエクスペリエンスが向上します。

シングル ページ アプリケーション (SPA) (続き)



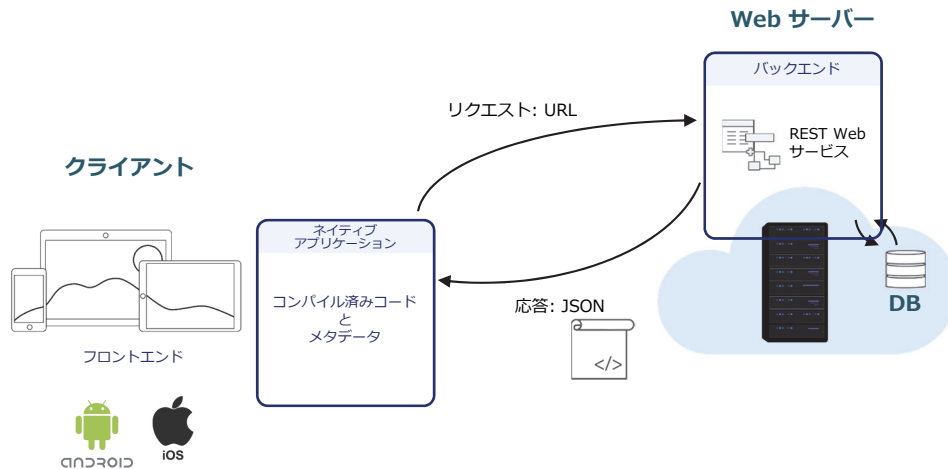
このタイプのソリューションを実装し、サーバーとの通信を必要とするアプリケーションをブラウザでシングルページに収めるには、JavaScript で直接プログラミングもできますが、Angular、React、Vue.js などの JavaScript フレームワークが利用されており、.NET、Java、Python などの言語と連携されることができます。

これらの各フレームワークは、DOM (ドキュメント オブジェクト モデル) の操作が非常に効率的です。DOM は、Web ページの HTML をノードや構造化オブジェクトとしてプロパティやメソッドとともに格納できるモデルです。最近のブラウザはすべてこの技術に対応していて、情報を迅速に解釈し、画面に表示できるようになっています。

また、これらのフレームワークでは、W3C 標準の Web コンポーネントも使用できます。Web コンポーネントは、HTML、JavaScript、CSS をカプセル化のため、GeneXus 開発者が作成するページ向けに主なジェネレーターでサポートされています。これにより、開発者間でのコントロールの再利用が促進されます。

SPA にはパフォーマンス上のメリットが多数ありますが、大規模システムではデメリットもいくつかあり、クライアントの負荷の高い場合は初期ロードが遅れる可能性があります。また、コードがほとんどクライアント側にあるため、セキュリティの監視が難しくなります。

モバイルアプリケーション (オンライン)



これまで Web アプリケーションを見てきましたが、ここではモバイルアプリケーションのアーキテクチャを確認します。

具体的には、Wi-Fi を使って常に接続しているモバイルアプリケーション (オンラインアプリケーション) について考えます。これが最も一般的なシナリオですが、高パフォーマンスや高可用性のモバイルアプリケーションは通常オフラインで設計されています。

前に説明したとおり、モバイルアプリケーションにも、クライアント (モバイルデバイス) で実行される部分と、クライアントに情報を提供するためにサーバーで実行される部分があります。

各モバイルプラットフォームには独自の言語があります: Android の場合は Java、iOS の場合は Swift です。

アプリケーションは、コンパイル後にデバイスにインストールされます。このファイルには、すべてのビジネスロジックと、アプリケーションのメタデータが含まれます。メタデータには、必要な API の URL など、ユーザーインターフェースやその他のリソースの実装に必要なすべてのものが含まれます。

アプリケーションを実行すると、Web サーバーへのアクセスが行われ、必要なデータを返すプログラム (サービス) が実行されます。このデータがアプリケーションで処理され、デバイスのユーザーに表示されます。

メリットの 1 つは、アプリケーションがデータベースに直接アクセスしないことです。アクセスは常にサービスレイヤーを介して行われ、このレイヤーは使用デバイスから独立しています。このため、Android 向けと iOS 向けのアプリケーションをコンパイルする場合に、両方のアプリケーションで同じサービスレイヤーを使用できます。

ネイティブアプリケーションは、パフォーマンスや、デバイスのハードウェアリソースおよびソフトウェアリソースへのアクセスの面で有利ですが、対象となるプラットフォーム向けのプログラミングを習得する必要があるというデメリットがあります。場合によっては、ネイティブアプリケーションを開発する前に、

Progressive Web Applications を検討したほうがいいこともあります。


まとめ

従来型 Web アーキテクチャは今も Web ページで広く使用されているが、Web アプリケーションではもはや時代遅れになっている

SPA が推奨されるが、システム規模が拡大した場合は、セキュリティやレンダリングなどの問題を考慮する必要がある

UX 重視の Web アプリケーションのアーキテクチャは、モバイルアプリケーションのアーキテクチャによく似ている

モバイルアプリケーションの開発では、プラットフォームに関する知識も求められるため、PWA も選択肢となる

GeneXus では、UX に優れた Web アプリケーションに加え、ネイティブ モバイル アプリケーションや Progressive Web Applications (PWA) も開発できる 

<https://www.genexus.com/ja-JP/japan/community-and-support-jp/product-help>

これまでの内容をまとめます。従来型 Web アーキテクチャは今も Web ページで広く使用されていますが、Web アプリケーションでは時代遅れと見られています。

シングル ページ アプリケーションは市場トレンドになっていますが、システム規模が拡大した場合は、負荷の増大やセキュリティ面の懸念から、問題が生じる可能性があります。

高パフォーマンスアプリケーションのアーキテクチャでは、サーバーで REST サービスを使用します。モバイルアプリケーションも同様です。いずれの場合も、クライアントイベントに応じてデータが更新されます。

モバイルアプリケーションは、デバイスリソースへのアクセスなど、Web アプリケーションと比べて多くのメリットがあります。しかし PWA との差は縮まっていて、PWA のプログラミングはネイティブアプリケーションほど複雑ではありません。

GeneXus を使用すると、開発者はユーザーエクスペリエンスに優れた Web アプリケーション (シングル ページ アプリケーションとそれ以外) を生成できるほか、ネイティブコードや PWA のアプリケーションも生成できます。対象となるプラットフォームのプログラミング言語やフレームワークを学んだり、関連する技術に習熟したりする必要はありません。

GeneXus の詳細については、Wiki ページを参照してください。



動画	https://www.genexus.com/community-and-support-jp/training?ja
ドキュメント	http://wiki.genexus.jp/
認定資格	training.genexus.com/certifications