

GeneXus アプリケーションの アーキテクチャ

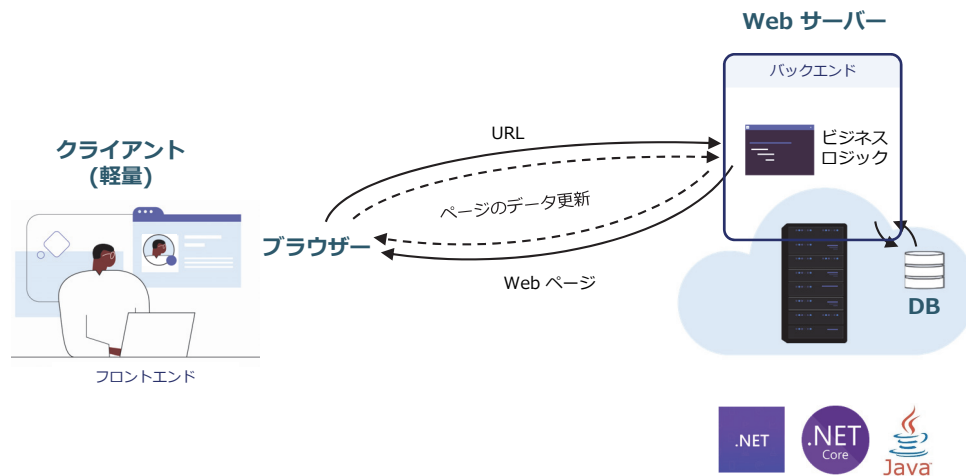


コースの初めで確認したように、GeneXus では、さまざまなプラットフォームを対象に、さまざまなタイプのアプリケーションを作成できます。

ここでは、アーキテクチャそのものと、そのアーキテクチャがどのようにアプリケーションのロジックと関係するのか、つまり、どのようにそれらが開発されるのかを見ていきます。

また、Work With パターンで生成された画面（後述する Web Panel オブジェクト）では、グリッドのページを変更するか、フィルタを適用すると、情報が動的にロードされ、サイズが自動調整されることも確認しました。

バックオフィス重視の Web アプリケーションのアーキテクチャ



こうした処理の一部は Web アプリケーションのクライアント側 (ブラウザー) で実行され、データベース検索などその他の処理はサーバー側で実行されます。

アプリケーションのロジックはサーバー側にあるため、アプリケーション内でオブジェクトを実行すると、ブラウザーからサーバーに、対応するページがリクエストされます。サーバーは情報を準備し、必要に応じてデータベースからデータを取得し、画面を組み立てて、それを表示のためにブラウザーに送信します。

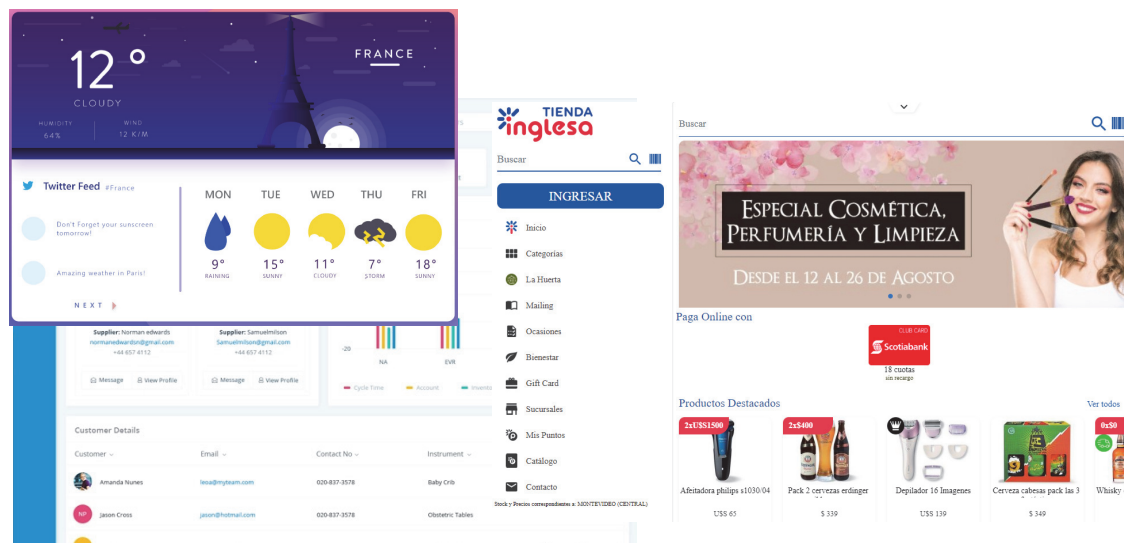
ブラウザーにページが表示されたら、何らかの操作を開始することがあります。たとえばグリッド内でクエリにフィルタを適用します。この場合、クライアント側のコードがサーバー側のコードと通信します。フィルタの条件を満たすデータがサーバーから返されたら、画面上で操作に関係のある部分のみが更新されます (この例ではグリッドのみ)。グリッド内での列の並べ替えや、トランザクションでフィールドを離れたときのメッセージ表示などは、クライアント単独で解決され、サーバーへのアクセスは不要です。

このアーキテクチャでは、クライアントも一定のインテリジェンスを持ちますが、決定はサーバーで行われます。アプリケーションのロジックは完全にサーバー側にあるからです。

これまで開発してきたのは、基本的にアプリケーションのバックオフィスです。データベースを操作する方法 (トランザクション)、情報を階層的に照会し、追加、削除、変更を行う方法 (Work With パターンで構築するものすべて) を見てきました。しかし、この同じアーキテクチャで顧客向けアプリケーションをビルドすることもできます。

GeneXus でこれらのアプリケーションをビルドするには、.NET、.NET Core、または Java を使用します。GeneXus では、クライアント (フロントエンド) とサーバー (バックエンド) の両方のコードにこれらの言語を使用します。

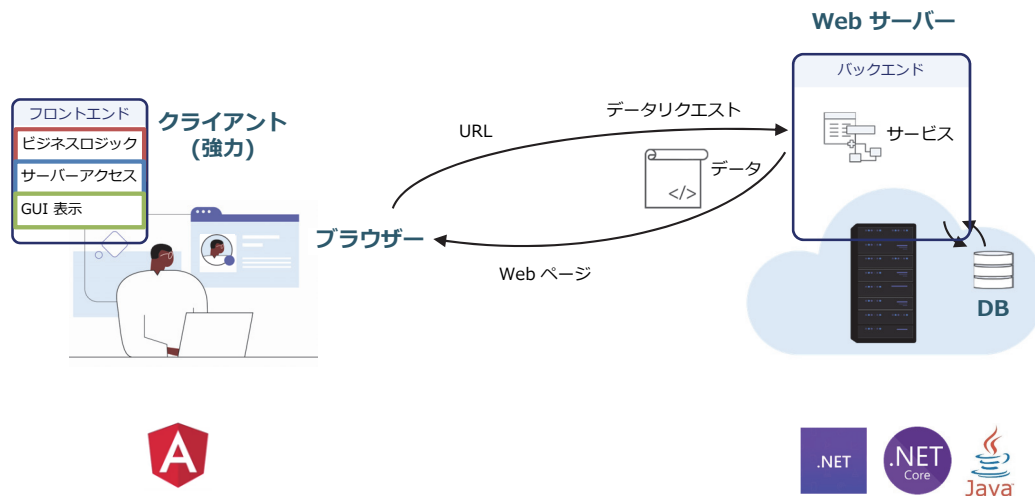
対話型コンテンツを豊富に含む高パフォーマンスアプリケーション



アプリケーションのタイプとしてはほかに、情報の表示が高速で、デザインと双方向性に重点を置いたものがあります。これは Web 向けの場合と、モバイル デバイス ネイティブの場合があります。

これは顧客向けアプリケーションであり、ユーザーに重点が置かれ、画面上の情報が充実しています。極めて高いパフォーマンス、優れたユーザーエクスペリエンスを目指して設計されているため、ページは必要なときのみ部分的にロードされます。

UX 重視の Web アプリケーションのアーキテクチャ



このアーキテクチャは、クライアントで実行される部分とサーバーで実行される部分がある点は同じですが、アプリケーションのほとんどのロジックがクライアント側にある点が異なります。クライアントには 3 つの異なるレイヤーがあります。ビジネスロジック、サーバーとの通信、そしてブラウザー内の画面表示の部分です。

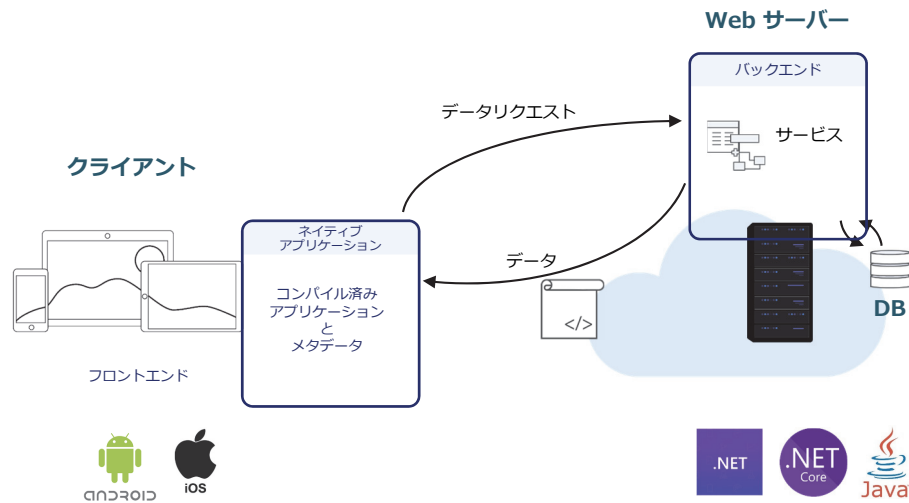
この構成では、ほとんどの機能がクライアント上で実行されます。サーバーにアクセスするのは、データベース内のデータや、サービスが提供するその他のリソースをリクエストまたは変更するときのみです。

このタイプの顧客向けアプリケーションは、優れたユーザーエクスペリエンスや双方向性を実現するために強力なクライアントを必要とし、前に見たレスポンスアプリケーションよりも新しいものです。

こうした需要に対応するため、このようなアプリケーションに必要なスマートクライアントをビルドできる Angular、React、Vue などのフレームワークが市場に新たに誕生しています。

前述のように、GeneXus では、Java、.NET、.NET Core でも顧客向けアプリケーションを開発できます。Angular でプログラミングを行うメリットは、ユーザーインターフェース用にデザインする同じ Panel オブジェクトをほとんど変更せずに再利用して、Android や iOS などのネイティブ モバイル アプリケーションを生成できることです。

モバイルアプリケーションのアーキテクチャ (オンライン)



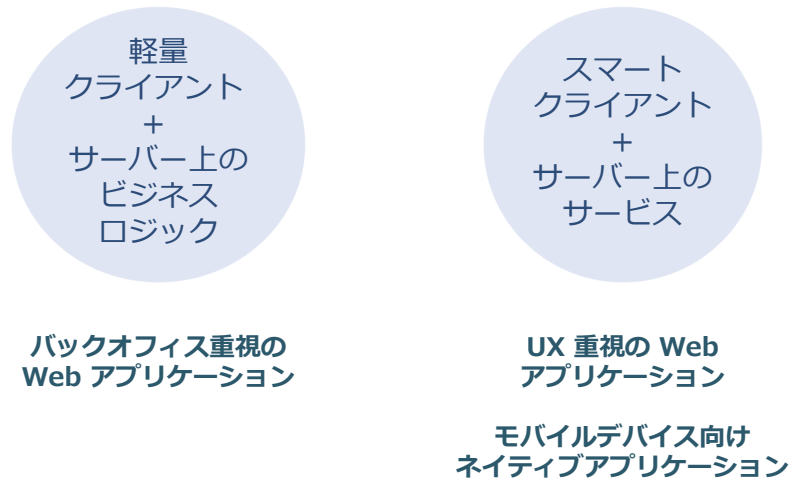
これまで Web アプリケーションを見てきましたが、ここでネイティブ モバイル アプリケーションのアーキテクチャを確認します。
ここに示すのは、Wi-Fi を通じて常に接続しているモバイルアプリケーション (オンラインアプリケーション) のアーキテクチャです。ネイティブアプリケーションにとって常時接続は一般的です。ただし、GeneXus ではオフラインアプリケーションをビルドすることもできます。

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。アプリケーションコードはデバイス上で実行され、データが必要なときにのみサーバーにアクセスします。

GeneXus では、Android デバイスと iOS デバイス向けのアプリケーションを生成できます。サーバーのサービス (クライアントへのデータ提供) は、Java、.NET、.NET Core で生成できます。

図から分かるように、このアーキテクチャは、高パフォーマンスの Web アプリケーションとよく似ています。このため、プログラミングの方法も似ています。いくつかの違いがありますが、詳細については後述します。

まとめると、プログラミング方法は 2 つ



アプリケーションのアーキテクチャが、アプリケーションの多くの側面、特にプログラミング方法を左右します。

軽量クライアントの場合、アプリケーションの要求のほとんどは、サーバーによって解決されます。このため、プログラミングは、データベースへのアクセスなど、常にサーバー処理に重点を置きます。

これに対して、強力なクライアントの場合、多くの処理を、サーバーに頼らずにクライアント上で実行できます。ただし、主にデータベースアクセスを伴う処理にサーバープログラムが必要となります。

このため、特定のコードをクライアント側で実行する場合と、サーバー側で実行する場合とで、異なる構文が必要となります。これは、モバイルデバイス向けネイティブアプリケーションのプログラミングにも当てはまります。

次に、アプリケーションタイプごとのプログラミング方法を見ていきます。



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