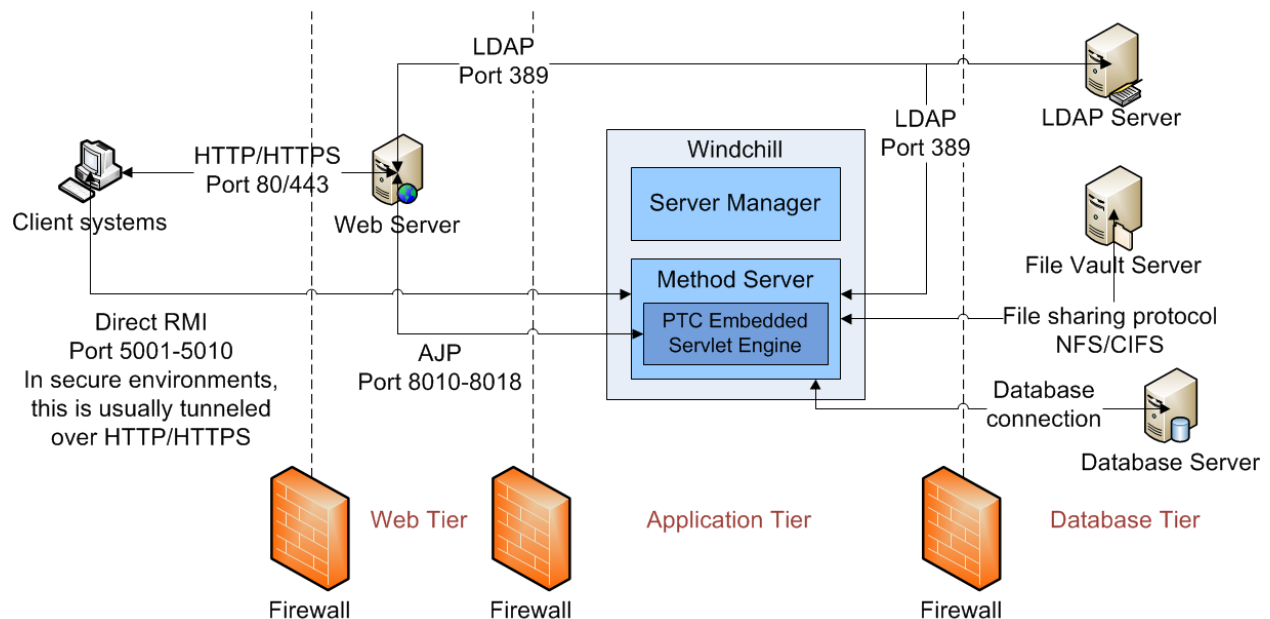
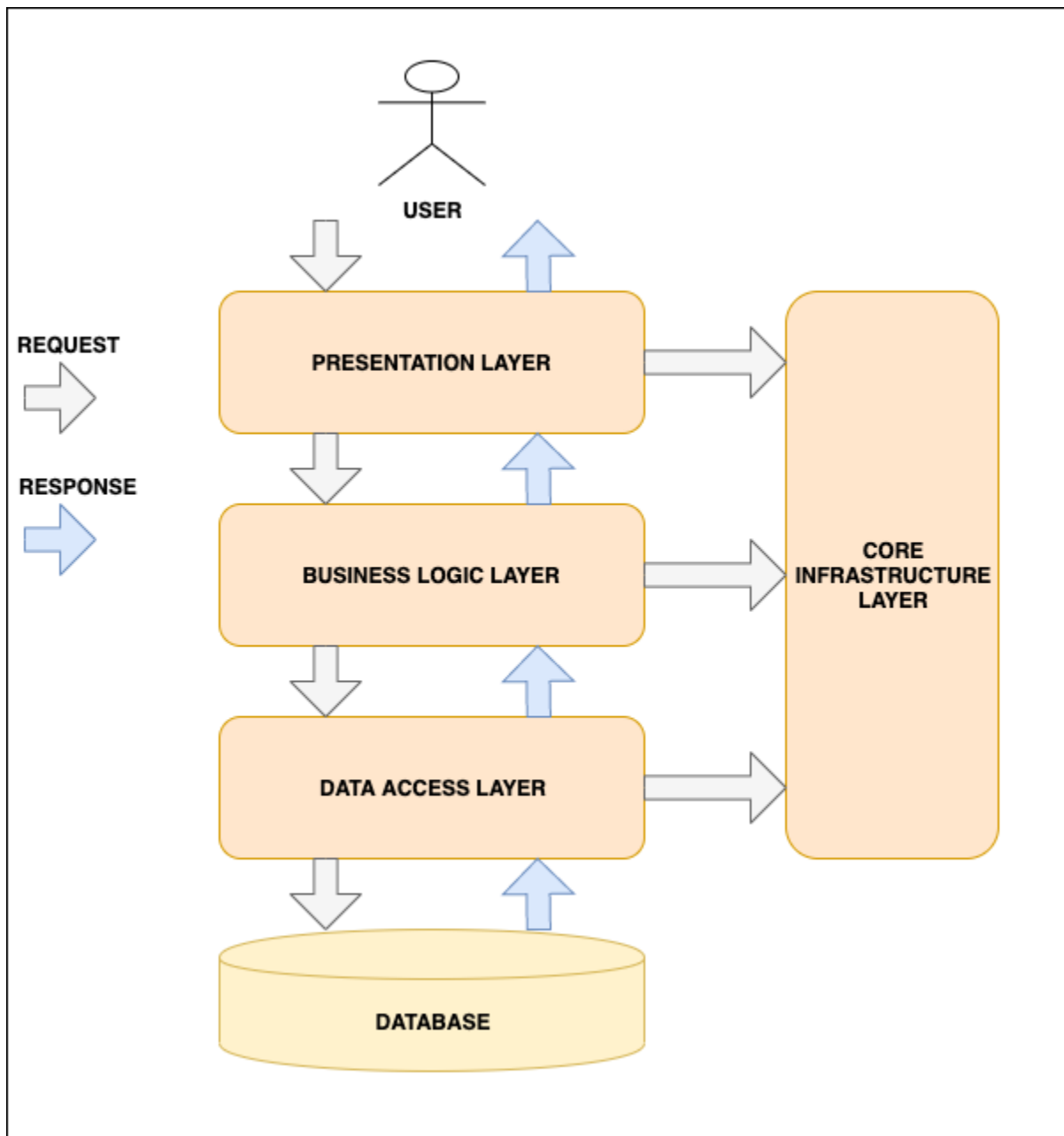


Architecture d'applications Web en 2022 : aller dans la bonne direction

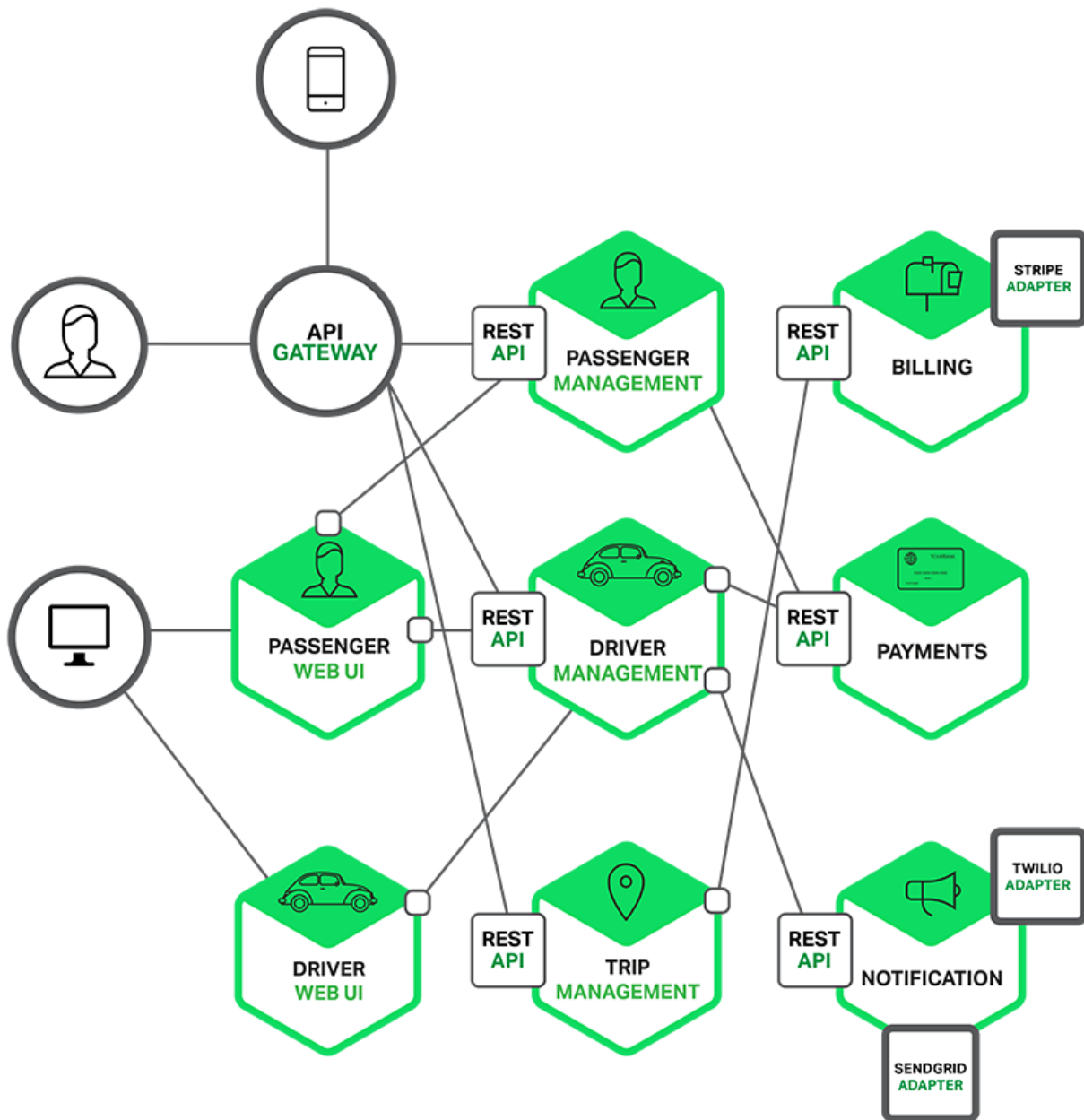
Anciennement, on avait une architecture serveur physique ou virtuelle qui ressemble à ceci :



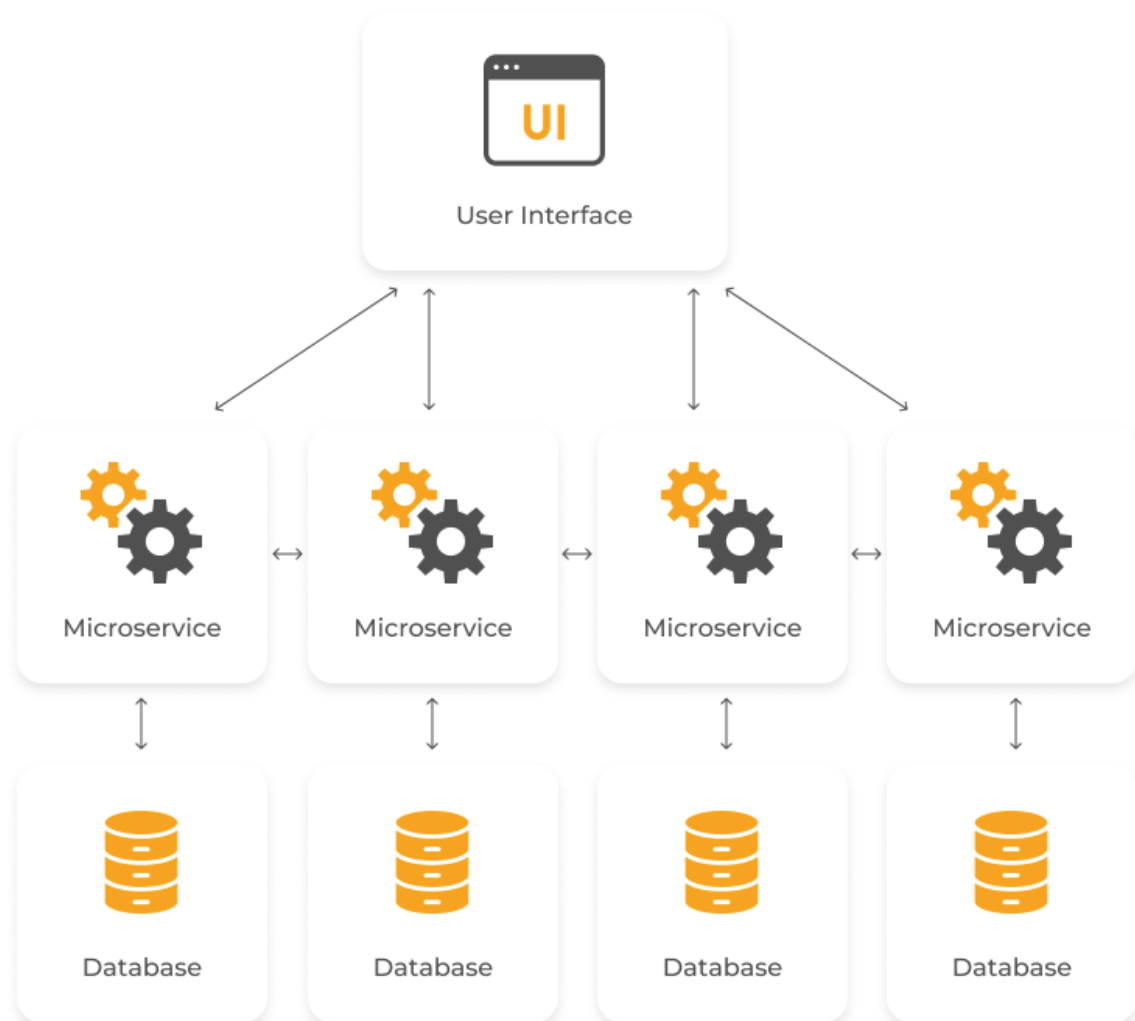
Les applications web étaient majoritairement développées en n-tiers :



Depuis quelques années on voit plutôt des architectures par micro services et un découpage par domaine.



Microservices



L'architecture des applications Web est une structure de haut niveau qui détermine la manière dont votre produit et votre entreprise fonctionneront et évolueront. De nos jours, l'étape du choix de l'architecture d'applications Web est souvent celle où vous vous perdez dans une variété d'options disponibles sur le marché du développement de logiciels. Plus de nouveaux noms et tendances apparaissent, plus il devient difficile de décider. Isomorphes, Progressive Web app, SPA ou SSR – quelle est la meilleure architecture d'application web moderne pour vous, et quels critères utiliser pour l'évaluation ?

Dans ce document, nous couvrons les principaux types d'architectures front-end disponibles pour le Web et expliquons les particularités de leur implémentation.

Qu'est-ce qu'une application Web par rapport à un site Web

Tout d'abord, définissons une application Web. C'est une application client-serveur, où il y a un navigateur (un client) et un serveur Web. La logique d'une application Web est répartie entre le serveur et le client, il existe un canal d'échange d'informations et de stockage de données situé localement ou dans le cloud.

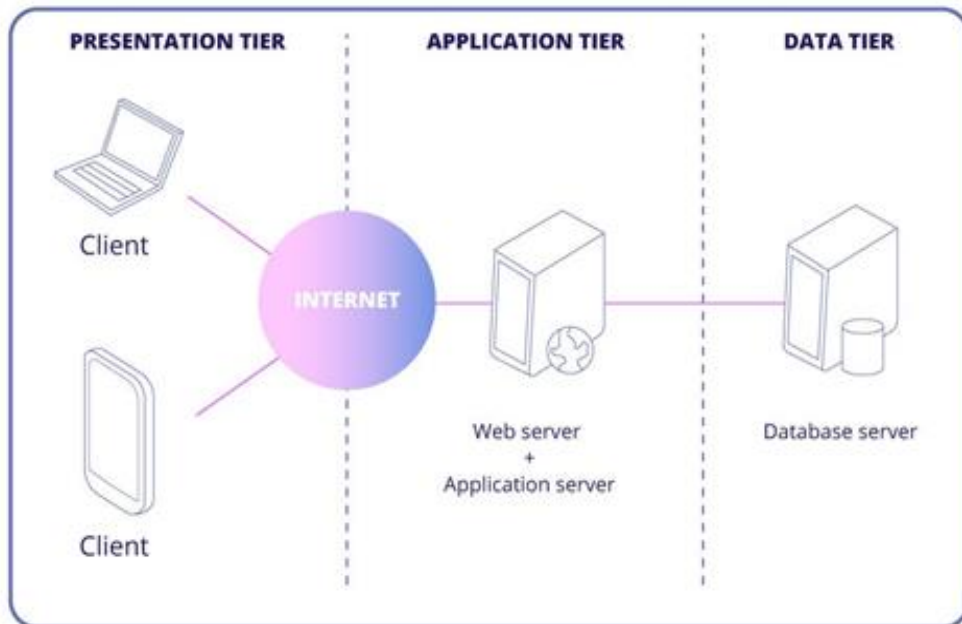
Les applications Web sont apparues comme une étape d'évolution du site Web et ont en effet beaucoup en commun. Les facteurs qui distinguent un site Web d'une application Web sont l'interactivité, l'intégration et l'authentification.



En d'autres termes, plus un site Web est personnalisable, interactif et fonctionnel, plus il s'apparente à *une application Web*.

ARCHITECTURE À 3 NIVEAUX DANS LE DÉVELOPPEMENT WEB

Les applications Web modernes utilisent toujours le concept d' [architecture à 3 niveaux](#) ✓, qui sépare les applications en *niveau de présentation*, *niveau d'application* et *niveau de données*.

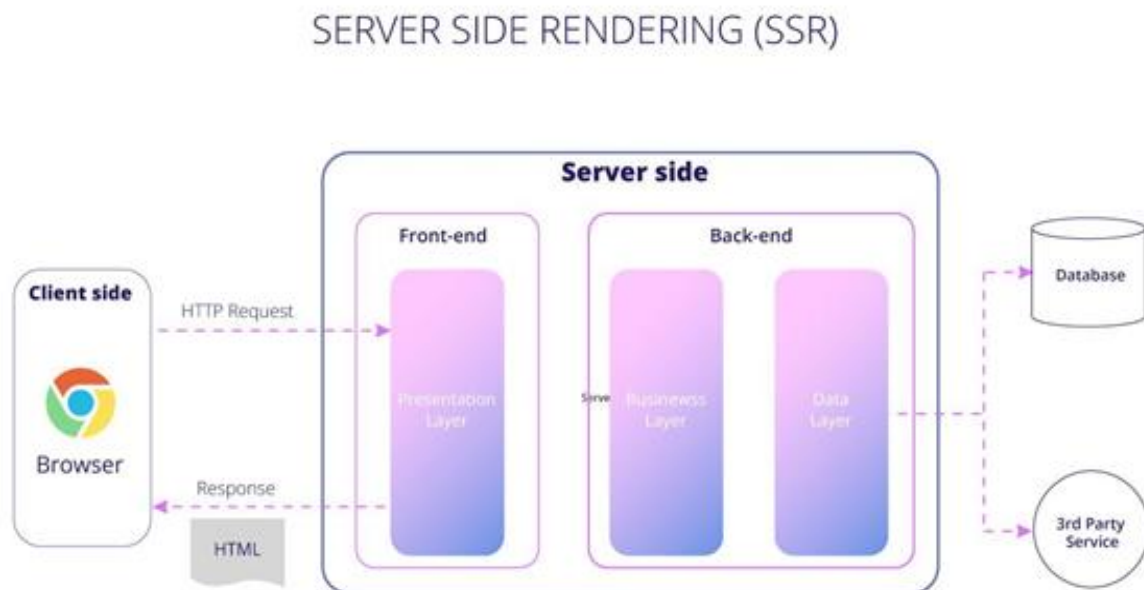


Au sein de l'architecture d'application Web à 3 niveaux, chaque couche s'exécute sur sa propre infrastructure et peut être développée en parallèle par différentes équipes. Une telle structure permet de mettre à jour et de mettre à l'échelle chaque niveau selon les besoins sans impacter les autres niveaux.

Types d'architecture d'application Web

Pour vous aider à comprendre si l'approche suggérée pour votre produit correspond vraiment aux besoins de votre entreprise, nous évaluerons les types d'architecture d'applications Web modernes en fonction des critères que nous considérons comme les plus vitaux pour les entreprises, à savoir les *performances*, *l'interface utilisateur*, *le référencement*, *la liaison* et *la vitesse de réalisation*. Du côté du développement.

RENDU CÔTÉ SERVEUR (SSR)



En parlant des principes de base du Web, nous entendons généralement l'architecture client-serveur. Un client demande du contenu à un serveur, où se trouvent la logique métier et une base de données. À l'aide de JavaScript simple, une page Web statique envoie la requête à un service (éventuellement une API). Le service renvoie les données et affiche une page HTML au client.

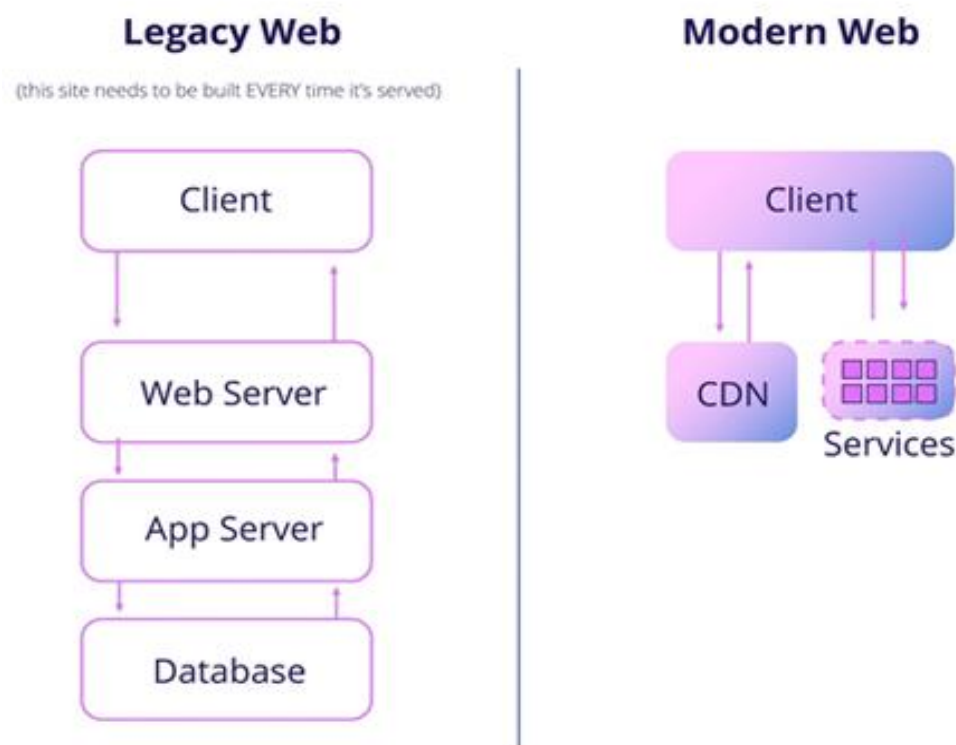
Si votre application est rendue côté serveur, le contenu est extrait du serveur et transmis au navigateur pour être affiché à l'utilisateur. Si la page HTML est rendue côté serveur, l'utilisateur doit accéder à la page avant que le navigateur récupère une page sur le serveur. Cela signifie qu'il faut plus de temps pour afficher le contenu à l'utilisateur. Pour mettre en cache le contenu de la page, ce schéma est souvent fourni avec Nginx, un serveur Web qui peut également être utilisé comme proxy de messagerie et équilibreur de charge.

Avantages et inconvénients.

Le fait que le HTML soit rendu sur le serveur offre un certain nombre d'avantages tels que le référencement, la possibilité de liens et un chargement instantané de la première page. Le rendu côté serveur fonctionne lorsque JS est désactivé dans le navigateur. Le code étant traité sur le serveur, aucune exigence spécifique au navigateur n'est imposée, cela nous permet de repérer instantanément les erreurs. Cependant, SSR ne peut pas gérer de lourdes demandes de serveur (HTML, CSS répétés), ce qui entraîne un rendu lent lorsque le serveur est chargé ou qu'une page complète est redémarrée. Mais le véritable talon d'Achille de ce type d'architecture d'application Web de base est une mauvaise interaction avec l'utilisateur final et l'incapacité de créer une interface utilisateur à part entière. En d'autres termes, SSR est un moyen simple et rentable d'aller si vous avez besoin de créer un site Web simple. La réalisation de ce type d'architecture est possible avec n'importe quel langage de programmation et back-end.

GÉNÉRATION DE SITE STATIQUE (SSG)

Le processus de génération de site statique implique un générateur qui automatise le codage de pages HTML individuelles, en les créant à partir de modèles. En choisissant Static Site Generation, vous recevez un site Web statique simple situé sur un [CDN](#) ou n'importe quel serveur, qui contient une page HTML déjà générée à donner aux utilisateurs sur demande. Donc, pas besoin de le générer à chaque fois que quelqu'un visite votre site Web - le serveur envoie simplement les données déjà existantes via une API.



Avantages et inconvénients. Tout d'abord, cette approche ne convient qu'aux sites Web. Parallèlement à cela, le contenu des pages de site Web générées ne change pas, sauf si vous ajoutez de nouvelles données ou de nouveaux composants. Cela signifie que vous devrez régénérer complètement le site Web une fois que vous souhaitez ajouter du nouveau contenu. C'est l'un des inconvénients majeurs qui limite sérieusement les analyses de rentabilisation auxquelles il s'applique.

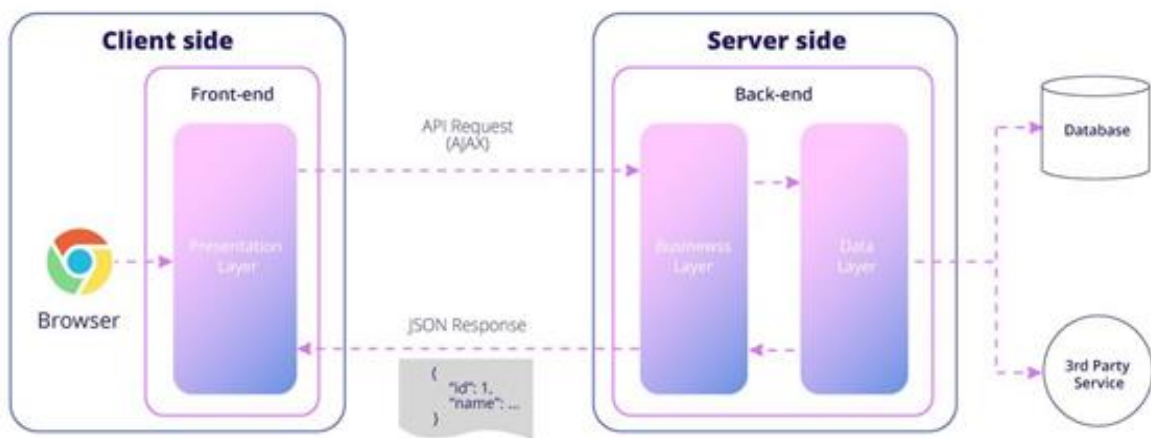
Parmi les avantages, cependant, il y a la vitesse élevée du contenu statique qui est livré via un CDN. De plus, dans SSG, toutes les opérations du serveur et le

travail avec la base de données sont réalisés via une API, indépendante du site Web. Cette option est simple et donc exclusivement abordable à réaliser. Jekyll et Hugo sont des exemples de [générateurs de sites statiques](#) ✓ simples, tandis que Gatsby et VuePress conviennent à la réalisation de solutions plus complexes.

APPLICATION À PAGE UNIQUE (SPA)

SPA est le type d'application Web qui fonctionne dans un navigateur. Il ne nécessite pas de recharger la page lorsque de nouvelles données doivent être affichées. Ce type d'architecture d'application Web est largement utilisé dans notre vie quotidienne : Facebook, Gmail, Google Maps, GitHub et Twitter – toutes sont des applications à page unique.

SINGLE PAGE APPLICATION (SPA)



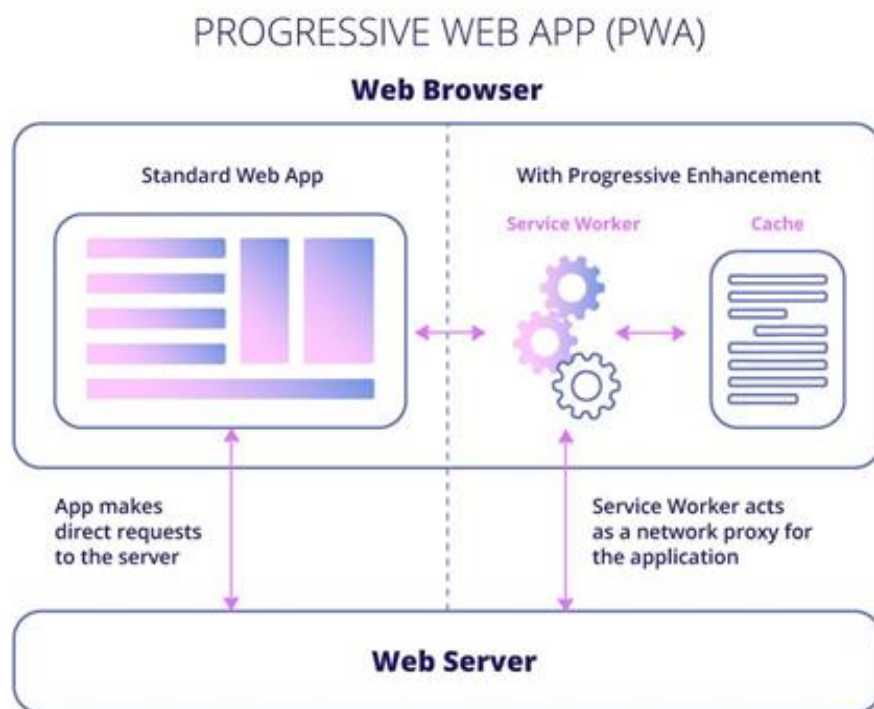
Avantages et inconvénients. Contrairement à SSR et SSG, SPA vous permet de créer une application Web interactive. Il utilise une API pour communiquer avec le serveur. Cette architecture est bonne pour faire évoluer facilement votre produit. De plus, si une application mobile est nécessaire, aucun effort supplémentaire n'est requis pour le développement de l'API : l'application mobile pourrait utiliser la même API que le Web.

SPA accorde un rendu rapide une fois que l'application est complètement chargée dans le navigateur et crée un logiciel très réactif pour l'utilisateur final. En même temps, cela « tue » votre référencement et limite la possibilité de liens, car la réalisation d'une telle fonctionnalité nécessitera des efforts supplémentaires. Parmi les autres inconvénients, citons le temps nécessaire

pour le premier chargement, un mauvais routage et une prise en charge limitée des navigateurs obsolètes. Étant un type d'architecture Web plutôt coûteux, SPA convient parfaitement à la création d'une interface utilisateur réactive pour les utilisateurs B2C.

APPLICATION WEB PROGRESSIVE (PWA)


Il semble qu'au cours des dernières années, tout le monde parle de PWA. Qu'est-ce qu'ils ont de si spécial et résolvent-ils vraiment tous les problèmes du développement d'applications Web ?



L'architecture d'application Web progressive utilise la logique d'une application Web à page unique avec certains services exécutés ensuite, dans le navigateur. Cela signifie que le point principal à prendre en compte est que le navigateur et le système d'exploitation doivent prendre en charge cet ensemble de normes.

Pour un utilisateur final, une application Web progressive signifie physiquement une offre contextuelle permettant d'ajouter l'application sur l'écran de lancement (pas un navigateur, mais l'écran de votre système d'exploitation),

lorsqu'il visite un site Web. Si l'utilisateur accepte, l'application est automatiquement ajoutée à l'appareil.

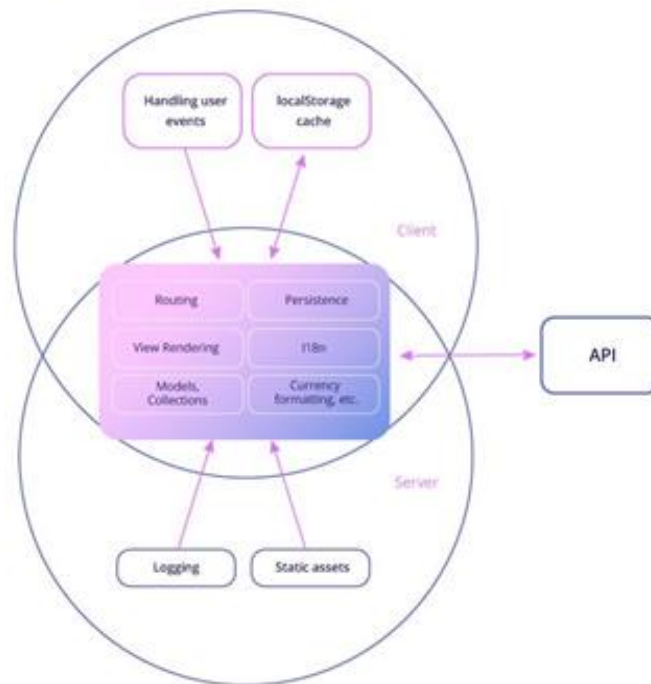
[La mise en œuvre de PWA](#)  permet à votre application Web de prendre en charge les expériences hors ligne, la synchronisation en arrière-plan et les notifications push. Cela ouvre l'accès à la fonctionnalité qui nécessitait auparavant une application native. Dans le même temps, en sélectionnant l'architecture PWA pour votre projet, vous devez garder à l'esprit que la majorité des fonctions ne sont pas disponibles sur iOS. Il vaut la peine d'analyser chaque cas d'entreprise spécifique.

Avantages et inconvénients. L'architecture d'application Web progressive est prise en charge par Windows, Android et iOS (pour iOS, le mode hors ligne est cependant désactivé). Les développeurs peuvent ajouter des mises à jour à une application Web à distance. Une PWA est sécurisée car elle utilise HTTPS. Dans le même temps, les utilisateurs finaux peuvent installer une PWA sans même visiter un Play Market ou une App Store. Parmi les inconvénients de ce type d'architecture, il y a la nécessité de sélectionner le navigateur et le système d'exploitation qui le prennent entièrement en charge.

ISOMORPHE

Une autre architecture d'application Web moderne est appelée isomorphe. Il s'agit d'un type d'application JavaScript qui peut s'exécuter à la fois côté client et côté serveur. Tout d'abord, le client charge un code HTML, où l'application JavaScript est téléchargée dans le navigateur, puis l'application commence à s'exécuter comme un SPA.

ISOMORPHIC WEB APP



Avantages et inconvénients. Contrairement au rendu côté serveur, l'architecture Web isomorphe fournit des mises à jour rapides des données, une réactivité et de multiples options UI/UX. Il assure un rendu plus rapide lorsque le serveur est chargé, car le code traité est transféré au client. Et contrairement au rendu côté client, il offre un affichage instantané dans le navigateur, un routage convivial, un référencement et une liaison. Le seul inconvénient de ce type d'architecture d'application Web est qu'il n'est entièrement pris en charge que par JavaScript. Le plus souvent, cela signifie que la pile technologique parmi laquelle choisir est limitée aux frameworks et outils basés sur JS.

Référence intéressante pour construire des applications isomorphiques avec React :

Isomorphic Web Applications

Elyse Kolker Gordon

 MANNING

Isomorphic Web Applications

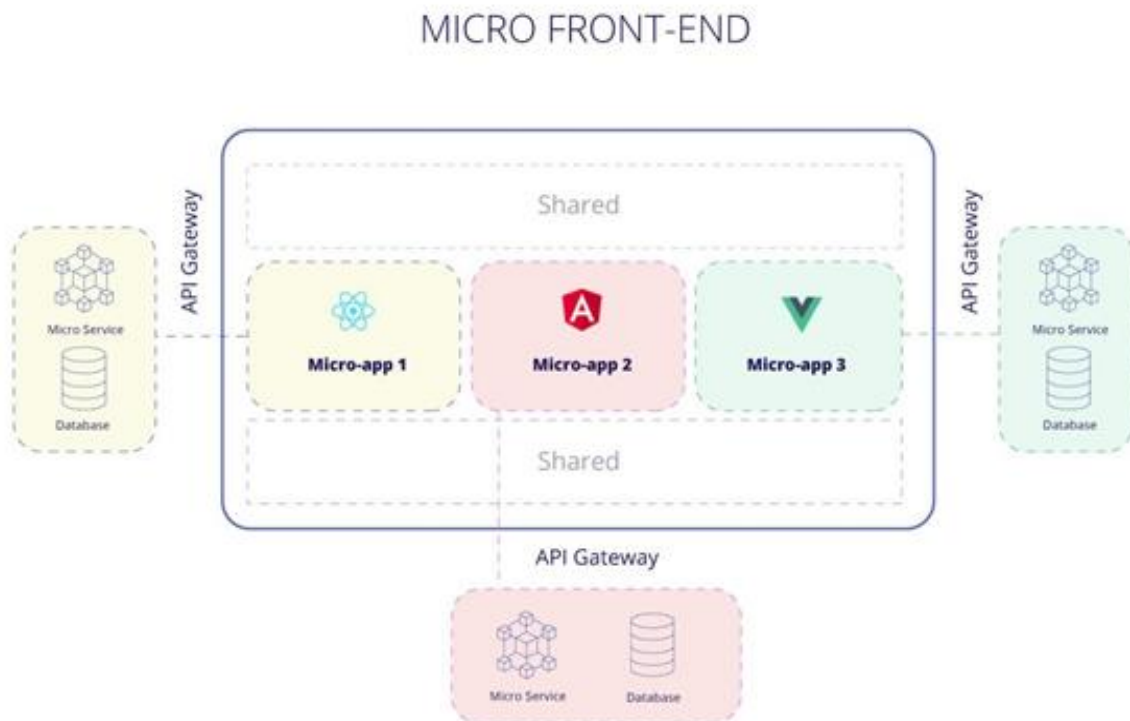
Universal development with React

Elyse Kolker Gordon



MICRO FRONT-END


Parmi les autres principes de conception d'applications Web, nous distinguons le micro front-end, une approche basée sur la décomposition d'une application frontale en « micro-applications » distinctes travaillant ensemble. Pour l'utilisateur final, ils sont tous situés sur une seule page.



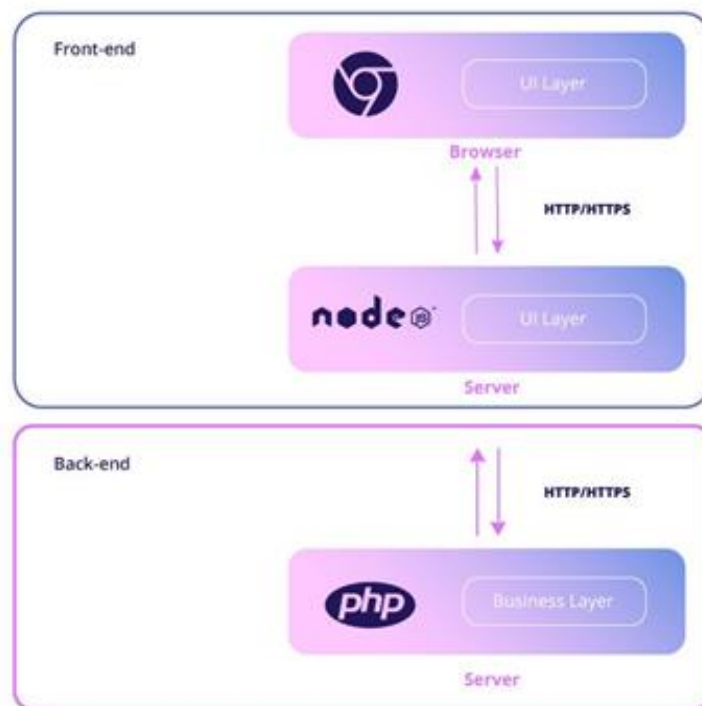
Ce type d'architecture d'application Web est modulaire, ce qui signifie que les pages et les widgets sont des applications complètement indépendantes. Avec une telle approche, le développement et le déploiement se déroulent en parallèle. Mais en même temps, la structure rend votre application compliquée et entraîne une duplication de code.

NODE.JS ET LE NOUVEAU WEB FRONT-END

Ce concept a été formulé pour la première fois en 2013 par Nicolas C. Zakas, et est maintenant utilisé pour des solutions Web complexes.

« Séparer la couche d'interface utilisateur principale de la logique métier principale est tout simplement logique dans une architecture Web plus large. Pourquoi les ingénieurs front-end devraient-ils se soucier du langage côté serveur nécessaire pour exécuter des fonctions critiques pour l'entreprise ? Pourquoi cette décision devrait-elle s'infiltrer dans la couche d'interface utilisateur principale ? Les besoins du front-end sont fondamentalement différents des besoins du back-end », c'est ainsi que Zakas [explique](#)  le grand principe de ce type de développement d'applications web.

Ce type d'architecture d'application Web se compose d'un serveur basé sur Node.js et d'une couche d'interface utilisateur. Parallèlement à cela, le serveur de logique métier peut être écrit dans n'importe quel langage (prenons PHP comme exemple) et utiliser une API pour communiquer avec le serveur.



En effet, la nouvelle approche web front-end nous permet d'exploiter tous les avantages du type architecture isomorphe, SSR ou SPA, API pour appareils

mobiles et linkability. Contrairement aux applications Web isomorphes, aucune restriction concernant le langage ou la plate-forme de logique métier n'apparaît. Cependant, il convient de noter que le développement d'une application Web utilisant la nouvelle logique d'interface Web prend plus de temps que celle utilisant le type SSG ou SSR. Pour gagner du temps pour le développement, nous vous suggérons d'utiliser les [Next.js et Nuxt.js](#) ✓ cadres. Le concept de nouvelle interface Web convient parfaitement au développement d'une application Web isomorphe, où un serveur d'API existe déjà.

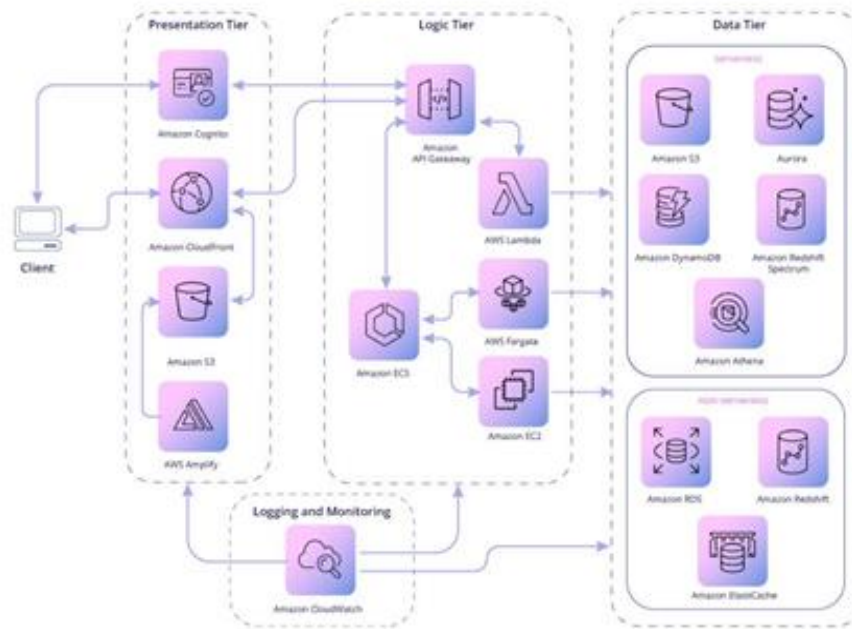
Architecture cloud pour le développement d'applications Web

En parlant d'architecture d'application Web, nous aborderons brièvement sa partie côté serveur. L'architecture cloud implique que les serveurs soient gérés par un fournisseur de cloud, tel qu'Amazon AWS, Azure ou Google Cloud. En plus d'héberger des serveurs de leur côté, ces fournisseurs proposent un ensemble de services qui permettent de construire des applications web hébergées et gérées dans le cloud.

ARCHITECTURE D'APPLICATION WEB SANS SERVEUR AWS

[Les services sans serveur AWS](#) ✓ sont l'une des solutions cloud les plus populaires utilisées pour implémenter des modèles courants tels que les microservices, les backends mobiles et les applications à page unique. Le schéma ci-dessous permet de comprendre comment les services Web AWS peuvent être utilisés pour créer une application Web à l'aide de la logique d'architecture à 3 niveaux que nous avons expliquée précédemment.

AWS WEB APP ARCHITECTURE



Source: AWS



ARCHITECTURE D'APPLICATION WEB AZURE

Azure fournit une variété de [services](#) de [cloud computing](#) qui vous permettent de créer des applications Web sans serveur de base et sécurisées. Une application web de base est construite avec *Azure App Service* et *Azure SQL Database*. Une combinaison de *Web App*, *Front Door*, *Function App*, *Azure DNS*, *Azure Cognitive Search* et *Azure DNS* permet d'améliorer l'évolutivité et les performances d'une application Web Azure App Service.

Une application web sans serveur développée avec Azure sert du contenu statique depuis *Azure Blob Storage* et implémente une API à l'aide d'*Azure Functions*. L'API lit les données de *Cosmos DB* et renvoie les résultats à l'application Web.

Azure App Service Environment (ASE) est utilisé pour déployer des applications Web où une sécurité supplémentaire est requise.

CONCEVOIR DES APPLICATIONS WEB HAUTEMENT ÉVOLUTIVES SUR GOOGLE CLOUD PLATFORM

En fonction des besoins de votre entreprise, de la maturité de l'équipe de développement et d'infrastructure, GCP propose *App Engine* ou *Cloud Run*, *Compute Engine* ou *Kubernetes Engine* pour créer des applications Web à l'architecture évolutive.

Conclusion

En choisissant la meilleure architecture pour le Web et une application mobile, vous pouvez vous retrouver perdu dans une variété de types d'architecture Web disponibles sur le marché. L'architecture d'application Web moderne comprend l'isomorphe, les PWA, le micro front-end et plus encore. Des acteurs massifs du marché comme Amazon, Google et Microsoft fournissent des structures complexes pour créer une architecture cloud aux trois niveaux d'une application Web. Dans cet article, nous avons présenté en détail les principaux types d'architecture d'applications Web. Mais la clé à retenir reste la même pour n'importe quelle entreprise de développement de logiciel. Lors de la sélection d'un type d'architecture, tenez compte des besoins spécifiques de votre entreprise et adaptez les critères que nous avons énumérés pour évaluer les options disponibles afin de choisir celle qui vous convient le mieux.

Références :

<https://mobidev.biz/blog/web-application-architecture-types>