Veille Technologique sur Flutter

Noa Caglioti

Sommaire

Introduction

Présentation du contexte

Dans le cadre de ma formation en BTS SIO (Services Informatiques aux Organisations), spécialité SLAM (Solutions Logicielles et Applications Métiers), je suis amené à réaliser une veille technologique sur un langage de programmation que je ne connais pas. Cette veille a pour objectif de m'informer sur les nouvelles technologies utilisées dans le monde du développement et de comprendre leur intérêt pour les professionnels du secteur.

Pour ce travail, j'ai choisi de m'intéresser à Flutter, un Framework développé par Google qui permet de créer des applications mobiles, web et de bureau à partir d'un seul code source. J'ai choisi de faire une veille sur ce Framework car sa popularité est croissante dans le monde du développement mobile.

Définition d'une veille technologique

La veille technologique est une démarche de recherche, de collecte et d'analyse d'informations sur les évolutions techniques, les innovations et les nouvelles tendances dans un domaine spécifique, comme ici la programmation ou le développement logiciel.

Elle consiste à se tenir informé en permanence des nouveautés comme des nouveaux frameworks, des mises à jour majeures ou même des usages en entreprise

Objectifs de cette veille

À travers cette veille, je vais étudier les origines, les caractéristiques techniques, les avantages et limites de Flutter, tout en comparant cet outil avec d'autres technologies similaires. Ce dossier me permettra donc de mieux appréhender les enjeux de la programmation multiplateforme ainsi que renforcer mes connaissances personnelles. Cela va aussi me permettre de comprendre l'environnement professionnel dans lequel je vais évoluer

Présentation de Flutter

Qu'est-ce que Flutter ?

Flutter est un framework open-source. Il permet de créer des applications mobiles (iOS et Android), web et desktop (Windows, macOS, Linux) à partir d'un seul et même code source.

La particularité de Flutter repose sur son langage de programmation, Dart, également conçu par Google. Dart est un langage orienté objet, avec une syntaxe proche de Java et JavaScript, ce qui facilite sa prise en main pour les développeurs familiers avec ces languages.

Flutter se distingue par son moteur de rendu graphique intégré (Skia), qui lui permet d'afficher des interfaces utilisateur fluides et personnalisées n'importe quels sont les composants de chaque support. Cela garantit d'avoir le même visuel et le même comportement sur tous les supports.

Historique

Flutter a été développé par Google. Sa première version stable a été publiée en décembre 2018, mais avant cela, Google commençait déjà à développer Flutter en 2015 sous le nom de code "Sky", avec l'idée de créer une solution rapide pour créer des applications mobiles. En 2017 une version Beta est sortie, puis c'est en Décembre 2018 que le lancement de Flutter 1.0 se concrétise, mais avec seulement le développement mobile. Il faut attendre 2020 pour la beta du support de Flutter Web et Flutter Desktop, puis Mars 2021 pour avoir la version 2.0 de Flutter, et pouvoir faire officiellement du multiplateforme.

En Mai 2022, Flutter 3 voit le jour. C'est actuellement la version la plus récente. Cette version ajoute un support stable pour les applications desktop, ce qui permet de faire du multiplateforme encore plus optimal.

Qui utilise Flutter ?

Depuis sa création, Flutter a très vite été utilisé grace à sa promesse de développement multiplateforme efficace, à sa communauté active et à l'investissement de Google dans les mises à jour pour enrichir et stabiliser l' outil. Il est utilisé aujourd'hui par des startups ou même des entreprises reconnues pour développer des applications modernes tout en étant performantes.

Fonctionnement et caractéristiques

Langage utilisé

Dart est un langage de programmation orienté objet et fortement typé, ce qui permet d'écrire du code robuste, mais avec des options simples d'utilisations comme "var".

Développé par Google en 2011, il a été spécialement conçu pour construire des interfaces utilisateur performantes, et c'est le langage utilisé exclusivement pour développer avec Flutter.

De plus, Dart a plusieurs fonctionnalités comme le fait d'éviter les erreurs dues aux valeurs nulles, et il gère facilement les opérations asynchrones. Ce sont deux des fonctionnalités proposées qui sont essentielles pour les applis modernes.

La syntaxe de Dart aussi est très proche de Java, JavaScript et C#, ce qui le rend assez simple à prendre en main pour les développeurs venant de ces langages.

Exemple simple en Dart :

```
void main() {
  for (var i = 0; i < 10; i++) {
    print('hello ${i + 1}');
  }
}
```

Compilation de Dart

L'un des gros atouts de Dart, c'est sa souplesse de compilation. Il prend en charge deux types de compilation :

en code interprété (Just-in-Time, JIT)	en code natif (Ahead-of-Time, AOT)
le code Dart est compilé au moment de l'exécution, ce qui est parfait pendant la phase de développement. Il n'y a plus besoin de relancer toute l'application à chaque changement, et il permet le Hot Reload, une fonctionnalité de Flutter qui actualise instantanément l'application quand on modifie le code, sans perdre l'état de l'appli.	cette compilation transforme le code Dart en code machine natif avant l'exécution, ce qui est idéal pour les applications en production. Il y a un temps de démarrage plus rapide et de meilleures performances globales grace à une optimisation plus poussée, mais le hot reload n'est pas présent et la compilation est plus longue

Flutter tire pleinement parti de ces deux méthodes selon les étapes du développement. Cela permet une expérience fluide pour le développeur et une performance optimale pour l'utilisateur.

Principe du "code une fois, déploie partout"

Flutter repose sur le principe du développement multiplateforme : on écrit un seul code source, et ce même code peut être déployé sur plusieurs plateformes comme Android, iOS, Web, Windows, macOS et Linux. Cela est rendu possible grâce à son propre moteur de rendu graphique (Skia), qui n'a pas besoin des composants natifs, et son langage Dart, qui permet une compilation en code natif sur chaque système

Ce principe permet un gain de temps et une maintenant plus simplifiée. Il n'y a plus besoin de recréer une interface par plateforme. Cela permet aussi à ce que le visuel soit cohérent entre toutes les plateformes

Pour les entreprises ça peut être un plus aussi, une seule équipe peut gérer le développement complet.

Architecture

Flutter utilise une architecture basée sur des widgets. Dans Flutter, tout est widget, cela signifie que chaque élément visible à l'écran, du plus petit bouton au conteneur principal, est

représenté par un widget. Ces widgets sont ensuite imbriqués pour construire toute l'interface.

Les différents types de Widget

Il existe deux types de widgets

le StatelessWidget	le StatefulWidget
Il est utilisé quand l'élément ne change pas (pas d'interaction, pas de mise à jour visuelle)	Il est utilisé quand l'élément peut changer pendant l'exécution (ex. : appuyer sur un bouton).

Exemple :
 	
import 'package:flutter/material.dart';
 	
class BonjourWidget extends StatelessWidget {
* @override*
* Widget build(BuildContext context) {*
* return Text('Bonjour Flutter !');*
* }*
}
	Exemple :
	
	class CompteurWidget extends StatefulWidget {
	  @override
	  _CompteurWidgetState createState() => _CompteurWidgetState();
	}
	
	class _CompteurWidgetState extends State<CompteurWidget> {
	  int compteur = 0;
	
	  void incrementer() {
	  setState(() {
	  compteur++;
	  });
	  }
	
	  @override
	  Widget build(BuildContext context) {
	  return Column(
	  children: [
	  Text('Compteur : $compteur'),
	  ElevatedButton(
	  onPressed: incrementer,
	  child: Text('Incrémenter'),
	  ),
	  ],
	  );
	  }
	}

L’arbre de widgets

Flutter organise ces widgets dans un arbre hiérarchique. Chaque widget peut contenir d’autres widgets, ce qui forme un arbre de widgets. Flutter reconstruit uniquement les parties nécessaires de cet arbre à chaque mise à jour, ce qui le rend réactif.

Hot Reload

Le Hot Reload est une des fonctionnalités les plus puissantes de Flutter. Il permet de modifier le code pendant l’exécution, de voir les changements instantanément à l’écran, et de conserver l’état de l’application, ce qui est pratique pour tester rapidement une interface. Cela rend le développement très fluide et interactif, surtout pour l’interface utilisateur.

IDE compatibles

Flutter peut être utilisé avec plusieurs environnements de développement (IDE) très populaires, ce qui facilite son adoption par les développeurs. Les plus utilisés sont Android Studio.Visual Studio Code, IntelliJ IDEA

Avantages et inconvénients

Avantages	Inconvénients
Un seul code pour Android, iOS, Web, Desktop	Taille des fichiers générés assez grande
Performance proche du natif grâce au moteur Dart	Language Dart peu utilisé en dehors de Flutter
UI cohérente sur toutes les plateformes (grâce à Skia)	Intégration web un peu moins puissante que mobile
Grande flexibilité graphique, personnalisations faciles	Flutter est encore jeune comparé à d'autres technologies
Documentation officielle très claire et complète	Moins de ressources communautaires que certains autres technologies comme par exemple React Native
Hot Reload pour un développement rapide	

Comparaison avec d’autres technologies

Flutter vs React Native

Critère	Flutter	React Native
Language utilisé	Dart	JavaScript
Rendu graphique	Skia	Utilise les composants natif de chaque OS
Performance	Très élevée, compilation en code natif	Généralement bonne, mais tout dépend de la liaison entre JS et les composants natifs
Hot reload	Oui, très performant	Oui, mais moins stable que Flutter
Expérience UI	Uniforme sur toutes les plateformes	Avec des différences entre iOS et Android
Facilité d'apprentissage	Demande d'apprendre Dart	Avantage si on connaît déjà JavaScript
Écosystème	Moins de plugins que React Native, mais en croissance	Très riche
Support communautaire	Grandissant (soutenu par Google)	Très large communauté (soutenu par Meta)
Stabilité	Très stable pour le mobile	Stabilité variable selon les versions ou OS

En résumé, Flutter a une meilleure performance, un contrôle total de l'UI, mais moins populaire et un peu niche à cause de Dart, tandis que React Native est plus populaire, JavaScript aussi est connu, mais parfois des soucis de performance sur des apps complexes

Flutter vs Kotlin/Swift natifs

Critère	Flutter	Kotlin (Android) / Swift (iOS)
Language utilisé	Dart	Kotlin pour Android / Swift pour iOS
Code multiplateforme	Oui, un seul code pour tout le monde	Non, 2 codes différents
Performance	Très proche du natif	Performance maximale en natif
Personnalisation UI	Très libre et rapide	Très précis mais plus long à implémenter

Temps de développement	Plus rapide (1 base de code)	Double de travail, développement séparée pour Android / iOS
Maintenance	Plus rapide (1 base de code)	Maintenance séparée pour Android / iOS
Accès aux fonctionnalités natives	Possible via plugins ou channels	Accès complet et direct
Utilisation professionnelle	Idéal pour des apps multiplateformes	Idéal pour des projets complexes très spécifiques ou natifs purs

En résumé Flutter est excellent pour des projets rapides, multiplateformes, avec un design personnalisé, tandis que Kotlin ou Swift sont à préférer si l'objectif est d'avoir le maximum de contrôle, de performance et d'intégration native.

État actuel et évolutions récentes

Dernière version stable

La version stable actuelle de Flutter est la 3.29, publiée en février 2025. Cette mise à jour apporte des améliorations significatives en termes de performances, de compatibilité avec les dernières versions d'Android et d'iOS, ainsi que des optimisations pour le développement web et desktop

Nouveautés récentes

Flutter 3.7 a introduit une prise en charge améliorée de Material 3, le système de design de Google. De nombreux widgets ont été mis à jour pour refléter les nouvelles directives de conception, offrant ainsi une interface utilisateur plus moderne et cohérente.

Exemple d'activation de Material 3 :

```
MaterialApp(
  theme: ThemeData(
    useMaterial3: true,
  ),
  home: MaPagePrincipale(),
);
```

Flutter facilite désormais l'intégration de fonctionnalités d'intelligence artificielle grâce au Flutter AI Toolkit. Ce kit permet d'ajouter facilement des fonctionnalités telles que la génération de texte, les chatbots, la reconnaissance vocale, et plus encore, en utilisant des modèles d'IA avancés comme Google Gemini AI ou Firebase Vertex AI.

Exemple d’intégration d'un chatbot avec Flutter AI Toolkit :

```
dependencies:
  flutter_ai_toolkit: ^latest_version
  google_generative_ai: ^latest_version
  firebase_core: ^latest_version
```

Évolutions à venir

Flutter 4.0 prévoit d'étendre son support aux systèmes embarqués, permettant ainsi le développement d'interfaces utilisateur pour des appareils au-delà des environnements mobiles et desktop traditionnels.

Avec l'évolution rapide de l'IA, Flutter envisage une intégration plus profonde des capacités d'intelligence artificielle, facilitant le développement d'applications plus intelligentes et réactives.

Flutter 4.0 devrait également apporter des améliorations significatives aux outils de développement, offrant une meilleure expérience pour les développeurs, notamment en termes de débogage, de tests et de performances.

Exemples concrets d'utilisation

Applications célèbres faites avec Flutter

Flutter est utilisé par de nombreuses entreprises renommées dans le monde entier. Grâce à ses performances et à sa capacité de créer des interfaces élégantes sur plusieurs plateformes avec un seul code source, il a séduit de nombreux grands noms comme Google Ads, BMW / MINI, eBay Motors, Tencent ou même Nubank

Utilisation en entreprise

Flutter n’est pas réservé aux grandes entreprises. Il est utilisé à la fois par des startups, qui apprécient sa rapidité de développement et sa capacité multiplateforme, des PME, qui peuvent maintenir une seule base de code et donc réduire leurs coûts, et aussi des grandes entreprises, qui l’adoptent pour des projets internes ou grand public (ex. : apps bancaires, apps de livraison, outils de gestion…)

De plus, Flutter dispose d’une communauté active et d’un écosystème riche, ce qui rassure les entreprises quant à sa pérennité et sa maintenabilité.

Conclusion

Ce que j’ai retenu de cette veille

Cette veille m’a permis de découvrir Flutter, un framework moderne, puissant et polyvalent. Bien que je ne le connaissais que de nom au départ, j’ai pu comprendre comment il permet de créer des interfaces soignées pour de nombreuses plateformes à partir d’un seul code.

J’ai aussi appris comment se structure une application avec des widgets imbriqués, les avantages de Hot Reload et le potentiel de Flutter en général à réduire le temps et les coûts de développement.

Pourquoi Flutter est intéressant dans le contexte du développement moderne

Flutter s’inscrit parfaitement dans les besoins actuels du développement logiciel, il répond à la demande de multiplateforme, sans sacrifier la performance ni la qualité visuelle. De plus il est soutenu par Google, garantissant stabilité, documentation et mises à jour, tout en s’intégrant bien dans les environnements de développement modernes. Il offre ainsi une expérience fluide pour les utilisateurs comme pour les développeurs.

Flutter est une solution pertinente et prometteuse pour de nombreux projets modernes, que ce soit dans un cadre professionnel ou personnel.

Sources

Pour cette veille technologique j’ai utilisé les sites Flutter.dev, Dart.dev, Stack Overflow, GitHub; Je me suis aussi servi du discord Flutter et de tutoriels sur Youtube