Introduction à la formation

Codabee

Qu’est ce que Flutter ?

Flutter est un framework open-source développé par Google pour créer des applications multiplateformes à partir d'une seule base de code.

Il permet de développer des applications mobiles pour Android et iOS, des applications web, et des applications de bureau pour Windows, macOS, et Linux.

Flutter utilise le langage Dart mais peut aussi intégrer d’autres langages de programmation comme par exemple:

- Swift
- Java
- Kotlin

L’ajout de ces langages permet de personnaliser le code pour une plateforme spécifique.

Codabee

Flutter multiplateforme

Flutter permet de créer des applications qui fonctionnent sur différentes plateformes avec **une seule base de code**.

Cela réduit le temps et les efforts nécessaires pour développer et maintenir des applications pour:

- Android
- iOS
- web
- Desktop

La seule chose que vous aurez à faire est de légèrement adapter UI et un peu de code pour satisfaire aux exigences des différentes plateformes. Plus besoin de repartir d'une page blanche pour déployer son application sur un nouveau support.

Codabee

Tout est Widget

Dans l'univers de Flutter, un mot revient sans cesse. Et d’ailleurs, nous allons souvent l’évoquer. Il s’agit des widgets.

Ce sont les éléments fondateurs qui permettent de construire des interfaces utilisateur (UI) riches et interactives.

Ils représentent les briques élémentaires avec lesquelles vous assemblez vos applications, comme des blocs Lego.

Imaginez un mur de briques : chaque brique individuelle contribue à la structure globale du mur, tout en apportant sa propre contribution esthétique et fonctionnelle. De la même manière, chaque widget dans Flutter joue un rôle spécifique dans la construction de l'interface utilisateur de votre application.

Codabee

Les différents Widgets

Flutter propose une large variété de widgets, chacun avec ses propres caractéristiques et fonctionnalités. On peut les classer en deux grandes catégories :

- **Widgets de base** : Les widgets de base sont les éléments fondamentaux de l'interface utilisateur, tels que les boutons, les textes, les images, les conteneurs et les icônes. Ils fournissent les éléments structurels et visuels de base de votre application.
- **Widgets avancés** : Les widgets avancés offrent des fonctionnalités plus complexes, permettant de créer des interfaces utilisateur plus interactives et dynamiques. Parmi les exemples, on trouve les listes déroulantes, les cartes, les formulaires, les animations et les transitions.

Les propriétés des Widgets

Chaque widget possède un ensemble de propriétés qui définissent son apparence et son comportement.

Ces propriétés peuvent être personnalisées pour adapter le widget à vos besoins spécifiques.

Par exemple, vous pouvez:

- modifier la couleur d'un bouton
- la taille d'une police de texte
- la source d'une image
- La dimension d’un container
- etc...

Codabee

Le WidgetTree

Les widgets sont organisés de manière hiérarchique, comme un arbre. On s’appelle d’ailleurs cette organisation le WidgetTree.

Un widget parent peut contenir un ou plusieurs widgets enfants, créant ainsi une structure arborescente qui représente l'interface utilisateur complète de votre application. Cette approche hiérarchique facilite la construction d'interfaces utilisateur complexes et organisées.

La plupart du temps, les Widgets ne pourront accueillir qu’un seul enfant (child). Cependant, certains Widgets comme les colonnes, les lignes ou encore les listes, spécifiques à la mise en page pourront accueillir plusieurs enfants (children) .

Des performances natives

Comment Flutter parvient-il à offrir des performances natives sur Android et iOS, tout en utilisant une base de code unique ?

1. Flutter possède son propre moteur de rendu: Skia. Il utilise des techniques de rendu (Vulkan et Metal) exploitées nativement par Android et iOS. Cela permet de contourner les limitations des frameworks web traditionnels.
2. Lorsque vous compilez une application Flutter, le code Dart est transformé en code natif pour la plateforme cible (Android ou iOS). Ce qui garantit que l'application s'exécute directement sur l'appareil, sans passer par des interpréteurs ou des machines virtuelles.
3.Flutter propose un ensemble de widgets optimisés pour les performances, construits à l'aide de primitives graphiques natives. Flutter utilise un système de déclaration impérative permettant de prédire précisément la disposition des éléments à l'écran.
4. Flutter tire parti de l'architecture native d'Android et d'iOS pour optimiser les performances. Il accède aux fonctionnalités natives des appareils, telles que le processeur, la mémoire et le capteur, de manière efficace, garantissant une expérience utilisateur fluide et réactive.

Codabee

Histoire de Flutter

2015 : Naissance d'une idée

L'aventure Flutter commence en 2015 lors d'une conférence interne chez Google. Des ingénieurs et des chercheurs explorent des moyens de créer des applications mobiles avec des performances natives et une interface utilisateur fluide, tout en tirant parti du langage de programmation Dart, créé par Google en 2011.

2017 : Premiers pas et nom de code "Sky"

En 2017, Flutter fait ses premiers pas sous le nom de code "Sky" lors du "Google Developer Day" à Shanghai. Ce prototype suscite l'intérêt des développeurs par sa promesse de performances natives et d'une expérience utilisateur fluide sur différentes plateformes.

2018 : Lancement officiel et version 1.0

2018 marque une année charnière pour Flutter. Le framework est officiellement lancé lors du "Flutter Live" en décembre, avec la publication de la version 1.0. Cette version stable offre un ensemble complet de fonctionnalités pour créer des applications mobiles multiplateformes.

Codabee

Histoire de Flutter

2019 : Adoption croissante et Flutter for Web
En 2019, Flutter connait une adoption croissante au sein de la communauté des développeurs. Sa facilité d'utilisation, ses performances et sa flexibilité attirent de nombreux utilisateurs. Google présente aussi "Flutter for Web", une extension permettant de développer des applications web avec Flutter.

2020 : Maturité et Flutter 2.0
2020 est l'année de la maturité pour Flutter. La version 2.0 du framework est publiée, apportant des améliorations significatives en termes de performances, de fonctionnalités et de stabilité. Flutter se consolide comme une solution incontournable pour le développement multiplateforme.

Histoire de Flutter

2021 : Flutter pour le bureau et "Fuchsia"
En 2021, Flutter s'étend au bureau avec la prise en charge officielle de Windows, macOS et Linux. Cette expansion renforce la polyvalence du framework et sa capacité à créer des applications pour un large éventail de plateformes. De plus, Flutter joue un rôle important dans le développement de "Fuchsia", le nouveau système d'exploitation de Google.

Depuis 2022 : Flutter 3.0 et au-delà
2022 marque l'arrivée de Flutter 3.0, une version majeure qui introduit de nouvelles fonctionnalités puissantes, telles que la prise en charge de la réalité augmentée (AR) et les améliorations des performances sur les appareils mobiles. Flutter continue d'évoluer à un rythme soutenu, avec de nouvelles fonctionnalités et une communauté de développeurs en pleine expansion.
Les itérations de Flutter s’enchainent et apportent constamment de nouvelles fonctionnalités comme par exemple l’ajout de WASM pour le Web en mai 2024

Introduction à Dart

Dart est un langage de programmation développé par Google, principalement utilisé pour développer des applications mobiles, web, et côté serveur. C’est le langage que nous allons utiliser pour concevoir des applications avec Flutter

Il est conçu pour être simple à apprendre et à utiliser, tout en étant très performant.

C’est un langage de programmation orienté objet qui se veut facile à apprendre, lisible et sûr.

Dart a été créé en 2011 par Lars Bak et Kasper Lund, deux ingénieurs de Google. Depuis, il a gagné en popularité et est utilisé par une communauté croissante de développeurs dans le monde entier.

Codabee

Dart en quelques points clés

1. **Polyvalence** : Dart permet de créer des applications pour une large gamme de plateformes, ce qui le rend idéal pour les développeurs qui souhaitent créer des solutions logicielles unifiées.
2. **Facilité d'apprentissage** : La syntaxe de Dart est simple et lisible, s'inspirant de langages comme C++ et Java, ce qui le rend accessible aux débutants.
3. **Performances** : Dart peut être compilé en code natif pour les plateformes mobiles et de bureau, ou en JavaScript pour le web, garantissant des performances optimales sur chaque appareil.
4. **Fiabilité** : Le système de typage statique de Dart et sa gestion automatique de la mémoire permettent d'écrire du code robuste et moins sujet aux erreurs.
5. **Outils et bibliothèques riches** : Dart dispose d'un écosystème complet d'outils et de bibliothèques open source pour faciliter le développement d'applications web, mobiles et de bureau.
6. **Utilisations concrètes** : Dart est utilisé par de nombreuses entreprises et organisations reconnues, telles que Google,New York Times, et GROUPON pour créer des applications performantes et évolutives.

Histoire de Dart

2009 : Les prémices

L'histoire de Dart commence en 2009 au sein de Google, lorsque des ingénieurs et des chercheurs entament des réflexions sur la création d'un nouveau langage de programmation. L'objectif est de concevoir un langage plus moderne, performant et adapté aux besoins du développement web et mobile, tout en tirant parti de l'expérience acquise avec Java et JavaScript.

2011 : Naissance officielle et premières versions

En 2011, Dart est officiellement présenté au public lors de la conférence GOTO à Aarhus, au Danemark. Les premières versions du langage sont publiées, permettant aux développeurs de découvrir sa syntaxe, ses fonctionnalités et son potentiel.

2012 : Évolution et adoption progressive

Au cours de l'année 2012, Dart continue d'évoluer avec l'ajout de nouvelles fonctionnalités et la publication de nouveaux outils de développement. Le langage commence à gagner en popularité auprès des développeurs web et mobiles, attirés par ses promesses de performance et de flexibilité.

Histoire de Dart

2013 : Lancement de Dart 1.0 et premiers succès

Publication de Dart 1.0, une version stable et mature du langage. Cette version s'accompagne d'une large adoption par des entreprises et des organisations, telles que Toyota, Khan Academy et Google Ads. Dart commence à se faire une place dans le paysage des langages de programmation pour le web et le mobile.

2014-2017 : Maturité et expansion

Les années qui suivent sont consacrées à la stabilisation et à l'amélioration de Dart. De nouvelles fonctionnalités sont introduites, la documentation s'enrichit et les outils de développement se perfectionnent. Dart s'impose comme un langage de programmation fiable et performant, utilisé pour créer des applications web et mobiles de grande envergure.

2018 : Dart 2.0 et le virage vers le multiplateforme

Publication de Dart 2.0. Cette nouvelle version apporte des changements significatifs à la syntaxe et aux fonctionnalités du langage, le rendant plus puissant et plus polyvalent. Dart 2.0 met l'accent sur le développement multiplateforme, pour créer des applications web, mobiles, de bureau et de serveur avec une seule base de code.

Histoire de Dart

2018-2022 : Adoption croissante et Flutter

Depuis 2018, Dart connait une adoption croissante au sein de la communauté des développeurs. Son utilisation s'étend à des domaines variés, allant du développement web et mobile aux serveurs et aux jeux vidéo. L'arrivée de Flutter, un framework open-source pour le développement d'applications multiplateformes natives avec Dart, contribue à renforcer la popularité du langage.

2023 et au-delà : Un avenir prometteur avec Dart 3

Aujourd'hui, Dart est un langage de programmation mature et en pleine expansion. Sa communauté de développeurs active, ses outils performants et son écosystème riche en bibliothèques en font un choix de premier plan pour la création d'applications modernes et performantes sur une multitude de plateformes. L'avenir de Dart s'annonce prometteur, avec de nouvelles fonctionnalités et des évolutions attendues pour répondre aux besoins croissants des développeurs et aux défis du monde numérique.
Ex: L’arrivée des Macro en mai 2024

Codabee

7 raisons de choisir Flutter

1. Développement Multiplateforme

Flutter permet de créer des applications qui fonctionnent sur Android, iOS, web, Windows, macOS et Linux à partir d'une seule base de code.

Cela réduit considérablement le temps et les efforts nécessaires pour développer et maintenir des applications sur plusieurs plateformes.

Codabee

7 raisons de choisir Flutter

2. Hot Reload

L'une des fonctionnalités les plus appréciées des développeurs Flutter est le "hot reload", qui permet aux développeurs de voir instantanément les changements de code dans l'application sans avoir à redémarrer celle-ci.

Cela accélère le processus de développement et permet de tester rapidement de nouvelles idées.

Vous voyez ainsi quasiment en temps réel les changements que vous apportez.

Codabee

7 raisons de choisir Flutter

3. Interface utilisateur riche

Flutter propose un ensemble complet de widgets personnalisables pour construire des interfaces utilisateur modernes et attrayantes.

Vous pouvez créer des interfaces utilisateur complexes avec des animations fluides et des effets visuels époustouflants.

Codabee

7 raisons de choisir Flutter

4. Performance native

Flutter offre des performances natives sur Android et iOS grâce à son moteur de rendu Skia et à la compilation en code natif.

Cela garantit des applications fluides et réactives, indistinguables des applications développées nativement sur chaque plateforme.

Codabee

7 raisons de choisir Flutter

5. Grande Communauté et Support de Google

Flutter est soutenu par Google et bénéficie d'une communauté active de développeurs. Il y a de nombreuses ressources disponibles pour apprendre Flutter, y compris des documentations officielles, des tutoriels, des vidéos, et des forums de discussion.

Le soutien de Google garantit également des mises à jour régulières et des améliorations continues.

D’ailleurs, si vous avez regardé les dernières GoogleI/O, avez-vous entendu les acclamations du public lorsque Flutter était évoqué?

Codabee

7 raisons de choisir Flutter

6. Simplicité d'apprentissage :

La syntaxe de Dart est claire et intuitive, s'inspirant de langages populaires comme C++ et Java, la rendant accessible aux débutants.

Les Widgets sont facile à comprendre et à intégrer.

Seule la partie configuration de Flutter sur votre machine peut être complexe à appréhender pour une personne ne sachant pas forcément utiliser le terminal.

Codabee

7 raisons de choisir Flutter

7. Design Consistant sur Toutes les Plateformes

Flutter permet de créer des interfaces utilisateur qui ont un design et une expérience utilisateur cohérents sur toutes les plateformes. Grâce à ses widgets, Flutter peut imiter les éléments de design spécifiques à chaque plateforme (Material Design pour Android et Cupertino pour iOS), tout en permettant des personnalisations poussées pour s'adapter à la marque ou aux besoins spécifiques de l'application.

Codabee