Riferimenti per Flutter

Cosa è Flutter

Flutter è un toolkit open-source per l'interfaccia utente (UI) sviluppato da Google e utilizzato per costruire applicazioni compilate in modo nativo per piattaforme mobili, web e desktop utilizzando un'unica base di codice. Flutter è stato introdotto per la prima volta nel maggio 2017 e da allora ha guadagnato molta popolarità tra gli sviluppatori grazie alle sue caratteristiche e capacità uniche. Ecco alcune caratteristiche chiave di Flutter:

- Flutter utilizza un modello di programmazione reattivo basato sull'"albero dei widget" che consente uno sviluppo rapido e ricariche a caldo, permettendo agli sviluppatori di vedere le modifiche apportate al codice immediatamente riflesse nell'interfaccia utente dell'app.
- Flutter fornisce un ricco set di widget e strumenti personalizzabili che consentono agli sviluppatori di creare interfacce utente belle e reattive per le loro app.
- Flutter consente di creare app efficienti e ad alte prestazioni che possono essere eseguite a 60 fps o più, offrendo agli utenti un'esperienza fluida e reattiva.
- Flutter supporta un'ampia gamma di piattaforme, tra cui Android, iOS, web e desktop, consentendo agli sviluppatori di utilizzare un'unica base di codice per più piattaforme.
- Flutter è dotato di un proprio linguaggio di programmazione chiamato Dart, facile da imparare e da usare per gli sviluppatori che hanno familiarità con altri linguaggi di programmazione come Java, C++ o JavaScript.
- Flutter fornisce una vasta gamma di librerie e pacchetti che possono essere utilizzati per estendere le sue funzionalità, rendendo più facile per gli sviluppatori costruire applicazioni complesse.
- Flutter ha una comunità di sviluppatori e appassionati in continua crescita, il che significa che c'è un sacco di supporto disponibile in termini di documentazione, tutorial e risorse guidate dalla comunità.

Nel complesso, Flutter è un toolkit per l'interfaccia utente potente e versatile che consente agli sviluppatori di creare applicazioni di alta qualità e performanti per più piattaforme utilizzando un'unica base di codice. Il suo modello di programmazione

reattivo, i widget personalizzabili e il supporto per più piattaforme lo rendono una scelta popolare per lo sviluppo di app.

Piattaforme supportate e Multi-Platform

Lo sviluppo multipiattaforma è la pratica di creare applicazioni software che possono essere eseguite su più sistemi operativi o piattaforme, utilizzando un'unica base di codice. Ciò significa che invece di sviluppare applicazioni separate per ogni piattaforma, come iOS e Android, gli sviluppatori possono scrivere un unico insieme di codice che può essere distribuito su più piattaforme. Lo sviluppo multipiattaforma offre diversi vantaggi, tra cui:

- Riduzione dei costi di sviluppo: Lo sviluppo multipiattaforma può essere più conveniente rispetto allo sviluppo di applicazioni separate per ogni piattaforma, in quanto richiede meno risorse, meno tempo di sviluppo e meno costi di manutenzione.
- Maggiore efficienza: Con lo sviluppo multipiattaforma, gli sviluppatori possono scrivere e mantenere un'unica base di codice per più piattaforme, il che può contribuire a ridurre i tempi e gli sforzi necessari per sviluppare e aggiornare le applicazioni.
- Migliore esperienza utente: Lo sviluppo multipiattaforma può contribuire a garantire agli utenti un'esperienza coerente su più piattaforme, poiché lo stesso codice base viene utilizzato per sviluppare applicazioni per tutte le piattaforme.
- Maggiore diffusione sul mercato: Lo sviluppo multipiattaforma può contribuire ad aumentare la portata di mercato di un'applicazione, in quanto può essere distribuita su più piattaforme con un'unica base di codice, consentendo agli sviluppatori di raggiungere un pubblico più ampio.

- Mobile: iOS and Android apps
- Web
- Desktop: Windows, MacOS, Linux
- Embedded devices

Chi lo utilizza?

- Flutter apps in production

- BMW: Scalare lo sviluppo del prodotto incentrato sul cliente al BMW Group con Flutter
- Google Pay: Diventare globale in Google Pay con Flutter
- eBay: Deliziare gli ingegneri di eBay con Flutter
- Nubank: Scalare con integrità a Nubank con Flutter
- Rive: Rive ha riscritto il suo potente strumento di animazione interamente in Flutter per consentire agli sviluppatori di creare bellissime illustrazioni multipiattaforma
- Supernova: una piattaforma di design system, ha usato Flutter per costruire la propria web app collaborativa per designer e sviluppatori
- Toyota: Miglioramento dei sistemi di infotainment alla Toyota con Flutter
- Superlist: Ripensare la gestione dei compiti e dei progetti con Flutter
- Tonal: Garantire la parità di funzionalità in Tonal con Flutter
- Crédit Agricole: Crédit Agricole mette i clienti al primo posto con Flutter

Metriche di successo

- Google Pay: riduzione del 70% degli sforzi ingegneristici
- eBay: 70% degli sviluppatori ritiene che lo sviluppo sia due volte più veloce rispetto alle soluzioni native
- Nubank: tasso di merge del 600% più veloce
- Google Pay: 35% di riduzione delle linee di codice
- Tencent: 80% di aumento dell'efficienza del debug
- eBay: 100% degli sviluppatori dichiara di preferirlo a iOS o Android
- Push: 20% di riduzione degli sforzi di manutenzione
- eBay: 98.3% di codice condiviso
- ByteDance: 33% di aumento della produttività
- Tencent: 33% di riduzione dello sforzo di sviluppo
- Tencent: 90% di codice multipiattaforma
- Push: 100% di parità di funzionalità

- Crowdsource: aumento del 100% della velocità di sviluppo
- Crowdsource: riduzione del 50% delle dimensioni del codice
- Nubank: 30% di miglioramento del tasso di successo delle merge rispetto a iOS

Marketshare e confronto con altri tool

Mobile

- Android è il sistema operativo mobile più diffuso, con una quota di mercato superiore al 70%. È utilizzato da un'ampia gamma di produttori, tra cui Samsung, Xiaomi, Huawei e altri.
- iOS è il secondo sistema operativo mobile più diffuso, con una quota di mercato di circa il 28%. Viene utilizzato esclusivamente sui dispositivi Apple, tra cui iPhone, iPad e iPod touch.

È importante notare che questi numeri possono variare a seconda della regione e del segmento di mercato analizzato. Ad esempio, in alcuni mercati emergenti, Android potrebbe avere una quota di mercato ancora più alta, mentre iOS potrebbe essere più popolare tra gli utenti di fascia alta in alcune regioni.

Desktop

- Windows è il sistema operativo desktop più diffuso, con una quota di mercato di circa il 75%. È utilizzato da un'ampia gamma di produttori di computer, tra cui Dell, HP, Lenovo e altri.
- macOS è il secondo sistema operativo desktop più diffuso, con una quota di mercato di circa il 16% e viene utilizzato esclusivamente sui computer Macintosh di Apple.
- Linux è un sistema operativo libero e open-source che ha una quota di mercato di circa il 2%. È utilizzato principalmente da sviluppatori e appassionati di tecnologia, oltre che in alcuni ambienti aziendali e governativi.

È importante notare che questi numeri possono variare a seconda della regione e del segmento di mercato analizzato. Inoltre, alcuni utenti possono eseguire più sistemi operativi sullo stesso computer utilizzando la virtualizzazione o le configurazioni dual-boot.

Tool di sviluppo multipiattaforma e nativi

- vs React Native (Flutter for React Native developers)

- vs Xamarin (Flutter for Xamarin.Forms developers)
- vs Web (Flutter for web developers)
- vs Android (Flutter for Android developers)
- vs iOs (Flutter for iOS developers)

Parole chiave del mondo Flutter

Widget

Il widget è un elemento fondamentale dell'interfaccia utente (UI) che rappresenta una parte dell'UI come un pulsante, un campo di testo o un'immagine. I widget possono essere stateless o stateful e possono essere combinati per creare interfacce utente complesse e reattive. Ecco alcune caratteristiche chiave dei widget in Flutter:

- Stateless Widget: Un widget stateless è un widget che non ha uno stato mutabile. I widget stateless sono immutabili e non possono cambiare il loro aspetto o comportamento una volta costruiti. Esempi di widget stateless sono i pulsanti, i campi di testo e le immagini.
- Stateful Widget: Un widget stateful è un widget con stato mutabile. I widget statici possono cambiare il loro aspetto o comportamento in base ai cambiamenti del loro stato. Esempi di widget stateful sono le caselle di controllo, i pulsanti di opzione e i cursori.
- Widget Tree: in Flutter, i widget sono disposti in una struttura gerarchica chiamata albero dei widget. L'albero dei widget rappresenta la struttura dell'interfaccia utente e definisce come i widget vengono annidati e combinati per creare layout complessi dell'interfaccia utente.
- Custom Widgets: Oltre ai widget integrati forniti da Flutter, gli sviluppatori possono creare widget personalizzati estendendo le classi di widget di base fornite da Flutter. Questo permette agli sviluppatori di creare componenti riutilizzabili che possono essere utilizzati in più schermate o applicazioni.
- Composizione: I widget di Flutter possono essere composti insieme per creare interfacce utente complesse e reattive. Ciò consente agli sviluppatori di creare interfacce utente che rispondono dinamicamente alle modifiche degli input dell'utente o ad altri eventi.

- Hot Reload: Una delle caratteristiche uniche di Flutter è la possibilità di utilizzare l'hot reload, che consente agli sviluppatori di vedere le modifiche apportate al loro codice riflesse immediatamente nell'interfaccia utente dell'applicazione. Questo può contribuire a velocizzare il processo di sviluppo e a ridurre gli errori.

In generale, i widget sono un elemento fondamentale dell'interfaccia utente in Flutter. Possono essere stateless o stateful e possono essere combinati in una struttura gerarchica per creare interfacce utente complesse e reattive. La possibilità di creare widget personalizzati, di utilizzare la composizione e di sfruttare l'hot reload rendono Flutter uno strumento potente ed efficiente per la creazione di interfacce utente di alta qualità.

- Introduction to widgets
- Flutter widget index
- Widget catalog
- Tutti i 215 widget: Every Flutter Widget Explained!

Hot Reload

La funzione **hot reload** di Flutter ti aiuta a sperimentare, creare interfacce utente, aggiungere funzionalità e correggere bug in modo rapido e semplice. L'**hot reload** funziona iniettando file di codice sorgente aggiornati nella Dart Virtual Machine (VM) in esecuzione. Dopo che la VM ha aggiornato le classi con le nuove versioni di campi e funzioni, il framework Flutter ricostruisce automaticamente l'albero dei widget, consentendo di visualizzare rapidamente gli effetti delle modifiche

Ecco come funziona l'hot reload in Flutter:

- Durante lo sviluppo di un'applicazione, lo sviluppatore apporta modifiche al codice utilizzando un IDE o un editor di codice.
- Una volta apportate le modifiche, lo sviluppatore salva il file, innescando il processo di ricarica a caldo.
- Durante l'hot reload, il framework Flutter aggiorna il codice Dart in esecuzione dell'applicazione con le nuove modifiche, senza riavviare l'applicazione o perdere il suo stato corrente.
- Il codice aggiornato viene quindi applicato all'interfaccia utente dell'app, consentendo allo sviluppatore di vedere immediatamente le modifiche.
- L'Hot reload può essere utilizzato per un'ampia gamma di modifiche al codice, comprese quelle al layout dell'interfaccia utente, allo stile e alla logica aziendale.

Può anche essere utilizzato per il debug del codice, testando rapidamente le ipotesi ed esplorando il comportamento dell'applicazione.

I vantaggi dell'hot reload sono

- Aumento della produttività: L'hot reload consente agli sviluppatori di vedere le modifiche al codice in tempo reale, riducendo il tempo necessario per vedere l'impatto delle modifiche e accelerando il processo di sviluppo.
- Miglioramento della qualità: l'hot reload può contribuire a migliorare la qualità del codice, consentendo agli sviluppatori di testare e debuggare rapidamente le modifiche, di individuare tempestivamente gli errori e di ridurre la probabilità di introdurre nuovi bug.
- Iterazione rapida: L'hot reload consente agli sviluppatori di iterare rapidamente il codice e il design dell'interfaccia utente, facilitando il perfezionamento dell'esperienza d'uso dell'applicazione e garantendo che soddisfi le esigenze degli utenti.

Nel complesso, l'hot reload è una potente funzionalità di Flutter che può contribuire ad accelerare il processo di sviluppo, migliorare la qualità del codice e aumentare la produttività.

Dart

Dart è un moderno linguaggio di programmazione orientato agli oggetti, creato da Google nel 2011. È stato progettato per essere un linguaggio di uso generale che può essere utilizzato per lo sviluppo web, mobile e desktop.

Ecco alcune caratteristiche chiave di Dart:

- Dart è tipizzato staticamente, il che significa che i tipi di variabili sono controllati a tempo di compilazione piuttosto che a tempo di esecuzione. Questo può aiutare a individuare gli errori nelle prime fasi del processo di sviluppo.
- Dart ha una sintassi simile a quella di altri linguaggi di programmazione popolari come Java e JavaScript, che lo rende facile da imparare e da usare per gli sviluppatori.
- Dart ha un modello di programmazione moderno e asincrono, progettato per facilitare la scrittura di codice in grado di gestire più attività contemporaneamente, come richieste di rete o input dell'utente.
- Dart ha un garbage collector integrato che gestisce automaticamente l'allocazione e la deallocazione della memoria, il che può aiutare a prevenire

perdite di memoria e altri errori legati alla memoria.

- Dart ha un gestore di pacchetti chiamato "pub" che rende facile la condivisione e il riutilizzo delle librerie di codice.
- Dart può essere compilato in codice nativo, il che lo rende veloce ed efficiente. Può anche essere compilato in JavaScript, che ne consente l'esecuzione nei browser web.
- Dart ha un ecosistema in crescita di strumenti e framework, tra cui il framework Flutter per la creazione di applicazioni mobili e desktop, oltre a framework web come Aqueduct e Angel.

Nel complesso, Dart è un linguaggio di programmazione versatile e moderno che ha un grande potenziale per lo sviluppo web, mobile e desktop. La sua forte tipizzazione, il modello di programmazione asincrono e il garbage collector integrato lo rendono uno strumento potente per la realizzazione di applicazioni affidabili e ad alte prestazioni.

- Bootstrap into Dart
- Why did Flutter choose to use Dart?

IDE

Un IDE (Integrated Development Environment) è un'applicazione software che fornisce un ambiente completo per lo sviluppo del software. Gli IDE includono tipicamente un editor di testo, un compilatore di codice, un debugger e altri strumenti che aiutano gli sviluppatori a creare, testare ed eseguire il debug del software in modo più efficiente.

Puoi creare app con Flutter utilizzando qualsiasi editor di testo combinato con gli strumenti da riga di comando di Flutter. Tuttavia, ti consigliamo di utilizzare uno dei nostri plugin per l'editor per un'esperienza ancora migliore. Questi plug-in forniscono il completamento del codice, l'evidenziazione della sintassi, l'assistenza alla modifica dei widget, il supporto per l'esecuzione e il debug e altro ancora

- VS Code
- Android Studio, IntelliJ

Documentazione

- Learn Flutter any way you want

Package e pub.dev

- Flutter Favorite program: identificare i pacchetti e i plugin che dovreste prendere in considerazione per la realizzazione della vostra applicazione. Questa non è una garanzia di qualità o di idoneità alla vostra situazione specifica: dovrete sempre valutare voi stessi i pacchetti e i plugin per il vostro progetto

Building a Flutter business

- Ads in your Flutter app: L'SDK Google Mobile Ads per Flutter funziona sia con AdMob che con Ad Manager. Disponibili anche una varietà di formati di annunci, inclusi banner, interstitial, video con premio e annunci nativi per guadagnare.
- In-app purchases in Flutter: Integra servizi per offrire contenuti aggiuntivi nella tua app come servizi premium, beni digitali e abbonamenti.
- Accepting payments: Semplifica l'integrazione con più fornitori di servizi di pagamento come Google e Apple Pay con il plug-in Pay per Flutter.
- RevenueCat: RevenueCat fornisce un backend per gli abbonamenti e un wrapper attorno a StoreKit di Apple, Google Play Billing e ai pagamenti basati sul web per semplificare la gestione degli acquisti in-app e la centralizzazione dei dati degli abbonati. Non è necessaria alcuna manutenzione del server o codice backend.
- Stripe: L'SDK di Stripe Flutter vi permette di creare esperienze di pagamento piacevoli nelle vostre app native per Android e iOS utilizzando Flutter. Forniamo schermate ed elementi dell'interfaccia utente potenti e personalizzabili che possono essere utilizzati immediatamente per raccogliere i dati di pagamento degli utenti.

Video:

- Monetizing apps with Flutter
- Flutter Update: App Monetization

Adaptive e Responsive

Adaptive: adattamento di una app per l'esecuzione su diversi tipi di dispositivi, come dispositivi mobili e desktop, richiede la gestione dell'input da mouse e tastiera, nonché l'input tattile. Significa anche che ci sono aspettative diverse sulla densità visiva dell'app, su come funziona la selezione dei componenti (menu a cascata rispetto ai bottom sheets), utilizzando funzionalità specifiche della piattaforma (come le finestre di primo livello) e altro ancora.

Responsive: la app ha un layout ottimizzato per le dimensioni dello schermo su cui viene visualizzata. Spesso questo significa ridisporre l'interfaccia utente se l'utente ridimensiona la finestra o cambia l'orientamento del dispositivo. Ciò è particolarmente necessario quando la stessa app può essere eseguita su una varietà di dispositivi, da un orologio, un telefono, un tablet, un laptop o un computer desktop.

- Designing truly adaptative user interfaces
- Platform-specific behaviors and adaptations
- Using Flutter to build a native-looking desktop app for macOS and Windows
- Adaptive and Responsive! Simple practical guide

Windows, MacOS e Linux

Per sviluppare applicazioni per Windows, macOS e Linux, Flutter offre una funzione chiamata "Flutter Desktop". Flutter Desktop consente agli sviluppatori di creare applicazioni native ad alte prestazioni per Windows, macOS e Linux utilizzando lo stesso framework Flutter usato per la creazione di applicazioni mobili.

Con Flutter Desktop, gli sviluppatori possono creare applicazioni che hanno un aspetto nativo su ogni piattaforma, con il supporto per l'input da tastiera e mouse, la gestione delle finestre e altro ancora. Flutter Desktop fornisce anche l'accesso a una vasta gamma di plugin e pacchetti che possono essere utilizzati per aggiungere funzionalità alle applicazioni desktop, come l'accesso al file system, la comunicazione di rete e altro ancora.

Nel complesso, Flutter Desktop è uno strumento potente per lo sviluppo multipiattaforma e consente agli sviluppatori di creare applicazioni desktop native di alta qualità per Windows, macOS e Linux utilizzando un'unica base di codice.

- Desktop support for Flutter
- Building Windows apps with Flutter
- Building macOS apps with Flutter
- Building Linux apps with Flutter

Continuous Integration (CI) and Continuous Delivery (CD)

La Continuous Integration (CI) è una pratica che consiste nel costruire e testare automaticamente le modifiche al codice man mano che vengono apportate, per poi integrarle in un repository condiviso con una certa frequenza, in genere più volte al

giorno. Questo aiuta a identificare e risolvere i conflitti e i bug del codice nelle prime fasi del processo di sviluppo, invece di aspettare la fine del ciclo di sviluppo, quando può essere più difficile e costoso risolverli. La CI contribuisce a migliorare la qualità del software, a ridurre i tempi di sviluppo e ad aumentare la produttività del team.

Continuous Delivery (CD) è un'estensione della CI, in cui le modifiche al codice che sono passate attraverso il processo di CI vengono automaticamente distribuite in produzione, manualmente o automaticamente. Ciò significa che ogni modifica che passa attraverso il processo di CI è candidata a essere rilasciata in produzione e il processo di distribuzione è automatizzato per garantire che sia veloce, affidabile e ripetibile. Il CD contribuisce a migliorare la velocità e l'affidabilità del processo di sviluppo del software e consente ai team di fornire software con maggiore frequenza e sicurezza.

- Codemagic
- Bitrise
- Appcircle

Quali tipi di applicazioni mi consiglieresti di sviluppare in Flutter?

Flutter è un framework versatile che può essere utilizzato per sviluppare un'ampia gamma di applicazioni. Ecco alcuni tipi di applicazioni che potreste considerare di sviluppare con Flutter:

- Applicazioni mobili: Flutter è stato originariamente progettato per lo sviluppo di applicazioni mobili ed è una scelta eccellente per la costruzione di applicazioni di alta qualità e performanti per iOS e Android. Con Flutter è possibile creare applicazioni con un look and feel nativo e sfruttare funzioni come l'hot reload, che possono contribuire a velocizzare il processo di sviluppo.
- Applicazioni desktop: Con l'introduzione di Flutter Desktop, è ora possibile utilizzare Flutter per sviluppare applicazioni desktop per Windows, macOS e Linux. Le applicazioni desktop sviluppate con Flutter possono avere un aspetto nativo ed essere altamente performanti.
- Applicazioni web: Flutter supporta anche lo sviluppo web, consentendo di creare applicazioni web altamente interattive e performanti utilizzando lo stesso framework Flutter che si usa per le applicazioni mobili e desktop.
- Applicazioni IoT: Flutter può essere utilizzato anche per sviluppare applicazioni Internet of Things (IoT), che possono essere utilizzate per controllare e

monitorare i dispositivi intelligenti. Con Flutter è possibile creare applicazioni altamente reattive e interattive che possono essere utilizzate per controllare e monitorare una vasta gamma di dispositivi.

Nel complesso, Flutter è un framework flessibile che può essere utilizzato per sviluppare un'ampia gamma di applicazioni, dalle applicazioni mobili alle applicazioni desktop, alle applicazioni web, ai giochi e alle applicazioni IoT. La scelta del tipo di applicazione da sviluppare con Flutter dipende in ultima analisi dai requisiti e dagli obiettivi del progetto.

Quali tipi di applicazione mi sconsiglieresti di sviluppare con Flutter?

Sebbene Flutter sia un framework versatile che può essere utilizzato per sviluppare un'ampia gamma di applicazioni, ci sono alcuni tipi di applicazioni che potrebbero non essere adatte a Flutter. Ecco alcuni esempi:

- Applicazioni ad alta intensità di CPU: sebbene Flutter sia altamente performante, potrebbe non essere la scelta migliore per le applicazioni che richiedono un uso intensivo della CPU, come le simulazioni scientifiche o le applicazioni di apprendimento automatico. Per tali applicazioni, si può prendere in considerazione l'utilizzo di un framework ottimizzato per le prestazioni, come TensorFlow o PyTorch.
- Applicazioni aziendali di grandi dimensioni: Flutter potrebbe non essere la scelta migliore per sviluppare applicazioni aziendali di grandi dimensioni che richiedono complesse integrazioni con sistemi e database esistenti. In questi casi, si può prendere in considerazione l'utilizzo di un framework di sviluppo di applicazioni aziendali più consolidato, come Java Spring o .NET.
- Applicazioni che richiedono funzionalità native: sebbene Flutter possa essere utilizzato per sviluppare applicazioni con un aspetto nativo, ci possono essere alcuni tipi di applicazioni che richiedono l'accesso a funzionalità native che non sono attualmente disponibili in Flutter. Per esempio, se avete bisogno di accedere a specifiche caratteristiche hardware o API del sistema operativo, potreste aver bisogno di utilizzare un framework di sviluppo specifico per la piattaforma invece di Flutter.

Nel complesso, Flutter è un framework potente e flessibile che può essere utilizzato per sviluppare un'ampia gamma di applicazioni. Tuttavia, come per ogni framework di sviluppo, è importante valutare attentamente i requisiti e gli obiettivi del progetto per determinare se Flutter è la scelta migliore per le vostre esigenze specifiche.

Che applicazioni potrei realizzare per cominciare a muovere i primi passi con Flutter?

Ci sono molti tipi di applicazioni che si possono sviluppare per iniziare con Flutter, ma ecco alcune idee per iniziare:

- Calcolatrice: Un'applicazione per calcolatrice è un ottimo modo per iniziare con Flutter, in quanto consente di esplorare le basi del design dell'interfaccia utente e dell'input dell'utente. Potete iniziare con un design semplice e aggiungere gradualmente altre funzioni, come le funzioni scientifiche o le conversioni di unità.
- App meteo: Un'applicazione meteo è un altro grande progetto per imparare Flutter. Potete usare API come OpenWeatherMap per recuperare i dati meteo e visualizzarli in modo semplice. Si possono anche aggiungere funzioni come la geolocalizzazione, le notifiche push o gli avvisi meteo.
- ToDo List: Un'applicazione per le liste di cose da fare è un classico esempio di semplice applicazione di produttività che può essere sviluppata con Flutter. È possibile progettare una semplice interfaccia utente per consentire agli utenti di creare, modificare e cancellare le attività e aggiungere funzioni come promemoria, date di scadenza o categorie.
- App a quiz: Un'applicazione a quiz è un ottimo modo per esplorare le basi della gestione degli stati e dell'input dell'utente di Flutter. È possibile creare una semplice interfaccia utente per visualizzare le domande e le risposte e aggiungere funzioni come punteggi, limiti di tempo o livelli multipli di difficoltà.
- App di chat: Un'app di chat è un progetto più complesso che può aiutarvi a esplorare le basi della comunicazione di rete e degli aggiornamenti dei dati in tempo reale. È possibile utilizzare Firebase o altri servizi basati su cloud per creare un'applicazione di chat in tempo reale con funzioni quali messaggi di testo, condivisione di immagini o chat di gruppo.

DEMO

Riferimenti e risorse online

1. Documentazione ufficiale
2. Getting Started
3. Widget catalog
4. Cookbook

5. Samples
6. FAQ
7. Canali YouTube
    a. Flutter
    b. Code With Andrea
    c. Fudeo - Corsi online in italiano su Flutter
    d. Reso Coder
    e. RobertBrunhage
    f. Codepur
8. Blog - Siti
    a. Code with Andrea
    b. Fudeo
    c. Robert Brunhage
    d. Reso Coder
    e. Flutter and Other Experiments