前回のレッスンでは、チャット画面を更新して、ここからメッセージを入力できるようにしたので、「どうしたの？」と言うことができました。

そして、送信をクリックすると、そのメッセージはFirebase Cloud onFirestoreのデータベースに直接送られます。そして、ここに新しいメッセージが表示されます。

そのため、その部分は完了し、データベースにメッセージを送信できます。

しかし、どのようにしてデータベースからデータを取得し、それを使用してアプリにデータを入力できますか？

これが、このレッスンで行うことです。まず、get CurrentUserがある場所のすぐ下で、getMessagesという新しいメソッドを作成し、このメソッドをFirestoreに追加します。

それで、私が持っているFirestoreをタップして、メッセージと呼ばれる適切なコレクションをタップします。次に、getDocumentsメソッドを呼び出します。これにより、将来のクエリスナップショットが返されることがわかります。

これで、このクエリスナップショットはFirebaseからのデータ型であり、現在のコレクションにあるデータのスナップショットです。

したがって、先に進んでこのメソッドを呼び出し、このメソッドの結果を使用したい場合は、もちろんそれを待つ必要があります。

最後に取得したデータを保存して、メッセージと呼びましょう。

もちろん、この待機を使用する場合は、このメソッドを非同期でマークする必要があり、これらの2つの部分の間に等号を挿入するので、警告やエラーは発生しません。

これらのメッセージを利用してドキュメントにアクセスすると、ドキュメントのスナップショットのリストを取得できることがわかります。

これらのドキュメントは、ここにあるものを参照しています。ドキュメントを追加する方法を参照してください。

既に3つあります。これらはメッセージコレクションに属します。

したがって、これらのすべてのドキュメントを取得するには、messages.documentsをタップする必要があります。

これはリストであるため、リスト内の個々のアイテムを表示できるようにするには、FORループを使用する必要があります。

私のお気に入りのFORループの1つであるfor inループを作成して、新しい変数を作成します。 messages.ocumentsでvar messageと呼びます。

したがって、これはもちろん、ループしているリストです。これらの各アイテムにはメッセージがあります。

そして、メッセージを把握したら、先に進んでそのドキュメントsnapshot.dataを印刷できます。

これで、キー値のペア、フィールドを参照する文字列、およびテキストを参照する値が得られます。

これでこのメソッドをトリガーできるようになります。ここで、現在サインアウトを実行している場所でこれを実行します。これはコメントアウトが最も簡単な部分だからです。

そのため、コードのサインアウトビットを削除します。このアイコンは、簡単に変更できる場合は変更できます。

今はこの方法をデモするために使用しているだけなので、そのままにしておきます。

そこで、今作成したgetMessagesというメソッドを呼び出します。

保存してアプリをホットリロードすると、getMessagesにリンクされたこのボタンをクリックできるようになります。クリックすると、コンソール内で表示できるようになります。

ここで気を付けてください。アプリの最終バージョンでは、ユーザーがログアウトボタンをクリックしてチャットメッセージを表示することはありません。

ここでコードを一時的に追加するだけなので、簡単にテストできます。

したがって、ここに印刷されているのは、現在私のメッセージコレクションにあるすべてのメッセージです。現在コードが機能している方法は、メッセージコレクション内のすべてのドキュメントを保持し、すべてのメッセージドキュメントから単一のメッセージを作成し、それらのそれぞれを印刷して、

それに関連付けられているデータ。

これが、ここで印刷されたすべてのドキュメントを取得する方法です。

これで非常にうまくいきましたが、ここで言って新しいドキュメントを追加すると、「ドキュメントID」は空白のままになり、フィールドは以前と同じようになります。つまり、送信者とテキストです。そして、ここに何らかの送信者の値とテキストの値を入力して保存をクリックするだけで、それが私のコレクションに含まれていることがわかります。

Firebaseダッシュボードから手動でメッセージを投稿するのではなく、別のユーザーから送信されたチャットメッセージであると想像してください。

別のユーザーがデータベースにチャットメッセージを投稿すると、このように表示されます。

しかし、ここを見てみると、更新された値を取得するために、そのメソッドを再度呼び出す必要があります。

すべてのメッセージを取得する前に、ボタンをもう一度クリックする必要があります。

事実上、Firebaseデータベースからデータを積極的に引き出しています。

はい、データベースを取得してチャットメッセージを取得します。

しかし、ドキュメントの取得を使用してチャットアプリを構築することのマイナス面を考えることはできますか？

なぜこれが私たちがそれを構築したいと思う方法ではないのですか？

友達とチャットしていると想像してください。

その場合、友達のメッセージがすぐに表示されるようにしたいですか？

それは、インスタントメッセンジャーがそれらのメッセージを配信することを望んでいることを意味します。

そのため、これを実現するには、おそらくgetドキュメントを1秒間に2回以上呼び出す必要があります。

友人におやすみなさいと言ったら、朝もチャットを続けると想像してください。

この時点で最後に必要なのは、チャットアプリがFirebaseで1秒間に2回、新しいメッセージをチェックインすることです。

そのため、アプリが新しいチャットメッセージ用にFirebaseをプルするための別のアプローチが必要です。会話パートナーまたはアプリをインストールした他の人から、新しいデータが追加されたときにデータをアプリにプッシュできるとしたらどうでしょうか？メッセージを自動的にプッシュして、画面に表示したいですか？

これが、WhatsAppとほとんどのメッセージングアプリのしくみです。

では、どうすればこれを行うことができますか？

これを行うには、ストリームと呼ばれるものについて知る必要があります。

次に、このメソッドgetMessagesをコメントアウトして、別のメソッドを作成します。

ここで、messagesStreamというメソッドを作成します。このメソッドは再び入力を受け付けませんが、メッセージのストリームをリッスンするために使用するため、非同期メソッドにもなります。これはfirestore.collectionmessages .getDocumentsで待機する代わりにFirebaseから送信されます。これは未来を提供するメソッドです。

したがって、これは将来のクエリスナップショットです。

代わりに、ここで使用するメソッドは、適切な「コレクション」、「メッセージ」コレクションを見つけるためのfirestore.collectionsです。次に、別の方法を使用します。これはスナップショットと呼ばれる方法です。

また、このメソッドは、将来のクエリスナップショットではなく、クエリスナップショットのストリームを返します。

そして、これは先物のリストのようなもので、先物の束です。

そして、そのストリームにサブスクライブすることにより、この特定のコレクションに発生するすべての変更を聞きます。

そして、ドキュメントに書かれているように、私のメッセージコレクションに新しいドキュメントが追加されると、今サブスクライブされてお​​り、新しい結果があれば通知されます。

それでは、これらのストリームをどのように処理しますか？

それについて考える最も簡単な方法は、ほとんどリストと考えることです。前に見たように、メッセージドキュメントのリスト内のすべてのメッセージをループするには、FORループを使用する必要がありました。

基本的に将来のオブジェクトのリストのようなスナップショットをループする必要があります。

そのため、いつでも新しいメッセージを作成し、それをメッセージコレクションに追加することができます。現在、そのストリームを購読しています。

それで、私が書くつもりの構文は、もちろん再び待ちます。

だから、私はそれを使用する前にこれが完了するのを待っています。

また、ここにFORループを追加します。

そのため、このfor inループを待っています。このストリームの結果を、スナップショットと呼ばれるものに保存します。

そして、そのforループは同じ構文を持ちます。

var snapshot in、この一連のスナップショット用です。

そして、このFORループの中かっこ内で、スナップショットにアクセスできるようになります。そして、スナップショットには.documentsと呼ばれるものがあります。

これまでと同じように、これはドキュメントのスナップショットのリストです。そのため、スナップショットをループするために以前から持ってきたこのコードを再び使用できますが、snapshot.documentになるようにいくつかの単語を変更する必要があるかもしれません。

そして、これらはそれぞれメッセージになります。そして、getMessagesを呼び出す代わりに、メッセージストリームである新しいメソッドを呼び出し、saveを押してコンソールを開きます。

そして、今度は、クロス記号を押して、そのメソッドmessagesStreamを呼び出すと、そのコレクション内のメッセージのサブスクライブを開始する必要があります。そして、各メッセージをコンソールに出力する必要があります。

それで、そのボタンを押して、そのメッセージを実行し、メッセージコレクションに現在存在するすべてのメッセージを取得する前と同じようにします。

しかし、これを見てください。

先に進んでFirebaseにドキュメントを追加すると、同じフィールドの送信者とテキストが再び使用されるため、送信者は再びangela@gmail.comになり、値は「Firebaseコンソールから無線で送信」になります。

ここでは、チャットメッセージを送信する別のユーザーをシミュレートしています。

大丈夫。

次に、保存を押してこのドキュメントをコレクションに保存し、Firebaseダッシュボードを使用してこれを実行しますが、すべてアプリでサブスクライブしているのと同じデータベースに移動します。

コンソールに戻ると、このコードの再実行がトリガーされたことがわかります。メッセージコレクションの変更をリッスンするように自分自身をサブスクライブしたため、変更が発生するとすぐにここにすべての結果が印刷されています。そのため、Firebaseデータベースからデータをプルする代わりに、データを効果的にプッシュしています。

これは本当にクールですよね？

私たちのアプリは現在、データベースへの変更をリッスンしています。

レッスンの最初の部分で行ったように、Firebaseをプルしてメッセージをチェックすることはありません。

現在、Firebaseはデータスナップショットのストリームを介してアプリに新しいメッセージを通知しています。

これは、まさにチャットアプリケーションに求める動作です。

そのため、ここで何が起こっているのかを理解するために、ストリームの仕組みについてもう少し理解する必要があります。

それがまさに次のレッスンで行うことです。

そこで、「ストリームとは正確にはどのようなもので、未来とどのように違うのか？」について深く掘り下げてみましょう。