FirebaseをFlutterアプリに組み込むという面倒な作業をすべて終えたので、いよいよFirebaseを実際に使用して、そのすべての可能性を活用します。そして、それが言及する良い点は、この段階の前にアプリを実際に実行してテストしたことを確認することです。これは物事が複雑になり始め、クリーンなスタート、つまりすべてが実際に機能していない場合に本当に毛むくじゃらになる可能性があるためですこの時点まで。しかし、準備ができたら、登録画面に進みましょう。

これらのgradleファイルなどの一部をまだ開いている場合は、必ず閉じてください。そのため、ユーザーがこれらのテキストフィールドに入力したデータの一部を実際に受信する準備が整いました。

そして、「電子メールを入力してください」という最初のテキストフィールドには、明らかに電子メールが含まれます。

そして、2番目のパスワードにはパスワードが含まれます。

そのため、テキストファイルが変数内で変更されたときに、ユーザーがここに入力した値を保存しましょう。

登録画面の状態の中で、2つの新しい変数を作成しましょう。

両方とも文字列になります。

1つは電子メールと呼ばれ、もう1つはパスワードと呼ばれます。

これらの変数を作成したら、ここでユーザーが入力した値と同じ値に設定します。

したがって、最初のものは電子メールになり、2番目のものはパスワードになります。

これで、ユーザーがここに入力した内容を保存できます。

そして、登録ボタンを押した瞬間に、パスワードの値と電子メールの値が必要です。

それでは、それをテストしてみましょう。

このボタンのonPressed内で、電子メールとパスワードも印刷してみましょう。

次に、保存をクリックしてアプリをホットリフレッシュし、新規ユーザーとして登録して登録してみましょう。

したがって、私は自分自身をangela@gmail.comと呼び、パスワードを123456として入力します。

そして、レジスタをヒットします。コンソール内を見ると、これら2つの値がキャプチャされ、onPressed内で使用できるようになっていることがわかります。

そして、ここでこれらの値を使用して、ユーザーを電子メールとパスワードで登録します。

ここで、テキストフィールドの現在の外観を見ると、非常に基本的なように見えますが、少し改善するためにできることがいくつかあります。

最初にしたいことの1つは、ここでテキストを中央に揃えることです。テキストフィールドのプロパティとして追加することで、本当に簡単に行うことができます。したがって、textAlignプロパティを追加し、それをtext align centerに設定し、パスワードフィールドに対して同じことを行うことができます。

そのため、この他のテキストフィールドに貼り付けます。ホットリスタートを実行すると、プレースホルダーまたはhintTextが中央に配置されていることがわかります。

また、タイピングを開始すると、それも中央に配置され、見た目が良くなります。

ここでホットリロードを試みて、ここでおもしろい動作と奇妙なテキストが表示された場合は、アプリをゼロからホットリスタートして、テキストフィールドが実際に更新されるようにしてください。

これが完了したら、2番目に気付くのは、パスワードが公開されているということです。

obscure textという2番目のテキストフィールドのプロパティを変更することで、これを標準のパスワードフィールドに変えることができます。

これをtrueに設定すると、このフィールドに入力し始めると、すべての文字または数字がこれらの小さなドットになります。

だから誰かが私の肩越しに見ているなら、彼らは私の極秘パスワードが何であるかを見ないでしょう。

最後に紹介したいのは、これらのテキストフィールドを使用するたびに、ここに表示される標準的なキーボードが表示されることです。

ソフトキーボードが表示されない場合、MacではCOMMAND + Kを押すことができ、Androidでこれらのテキストフィールドのいずれかをクリックすると、小さなキーボードアイコンがここに表示されます。その仮想キーボードの表示切り替えに応じて、キーボードのオンとオフを切り替えることができるはずです。

コードに戻って、ここに表示されるキーボードは両方のフィールドで同じであり、@記号を取得するために記号を入力する必要がある場合にメールを入力しているときは非常に厄介です。

それで、私たちができることは、これである電子メールテキストフィールド内に、キーボードタイプであるもう1つのプロパティを追加することができます。また、キーボードタイプをテキスト入力タイプにし、タイプを.emailAddressにするように指定できます。

そして、私がセーブを押したとき、あなたは違いを見ます。

これで、一番上のキーボードをアクティブにすると、実際にはスペースと@記号を含むキーボード入力キーボードであることがわかります。すべての一般的な語尾、the com、netなどを取得します。

したがって、これによりユーザーの操作が簡単になります。これらは非常に小さな調整です。登録画面のテキストフィールドにこれらを追加したので、ログイン画面でも同じものが必要であることをおそらく理解するでしょう。

これらをログイン画面にコピーしてみましょう。

したがって、最初のテキストフィールドはキーボードタイプを取得し、2番目のテキストフィールドは不明瞭なテキストとテキストの整列を取得するので、これら2つのテキストフィールド内にそれらを配置できます。

それでは、登録画面に戻り、実際にユーザーの登録を行いましょう。

ユーザーを認証するために、先ほどインストールしたパッケージの1つをインポートします。

そして、それがFirebase認証パッケージです。

そして、認証を使用するために、それを登録画面にインポートしましょう。

したがって、authと入力すると、firebase\_auth.dartパッケージが見つかるはずです。

次のステップは、新しい認証インスタンスを作成することです。

そして、ここで登録画面の状態の中でそれを行います。

一度作成したら変更しないので、最終版として作成します。他のクラスが誤って使用しないように、\_authを呼び出してプライベートプロパティとして保持します。この変数を台無しにします。

そして、FirebaseAuth.instanceと等しくなるように設定します。これは、このクラスの静的インスタンスであることがわかります。

そのため、この認証オブジェクトを使用して、実際にユーザーを認証します。

関連する方法、つまり、電子メールとパスワードを使用したサインイン、および電子メールとパスワードを使用したユーザー作成メソッドを使用します。

したがって、認証インスタンスを作成したら、ここでonPressedに移動し、ここで電子メールとパスワードを取得します。

それらを実際に印刷する必要はありませんが、それを使用してユーザーを登録できる必要があります。

ユーザーを登録するために、今作成した認証オブジェクトを利用します。

そして、関連するメソッドの1つを使用して、電子メールとパスワードでユーザーを作成します。また、文字列としてメールとパスワードの2つの必須フィールドのみが必要であることがわかります。そして、ユーザーの認証と作成に時間がかかるため、未来を返します。

そのため、その間、ユーザーインターフェイスがハングするのは望ましくありません。 Enterキーを押して、Eメール変数をEメールとして、パスワードをパスワードとして入力してみましょう。

そして、これは未来を返すことになるので、その未来を保持し、最終的な変数にそれをキャプチャしましょう。

それをnewUserと呼び、このメソッドの出力と等しくなるように設定します。

これは非同期メソッドであるため、新しいユーザーが作成されたかどうかがわからないまま続行したくありません。

したがって、onPressedを非同期メソッドに変更し、メールとパスワードを使用して作成ユーザーの前にキーワードawaitを追加します。

これにより、次の手順に進む前に、ユーザーの認証と作成が確実に完了したことを確認できます。

これはいくつかの理由で失敗する可能性があるため、ユーザーはすでに登録されているメールを入力したり、無効なメールやパスワードなどを入力した可能性があります。

tryブロックとcatchブロックを使用して、発生する可能性のある例外をカットできるようにする必要があります。

そして今のところ、私は単に例外を出力するだけです。

それに対処するはるかに派手な方法がありますが、ユーザー部分の作成に焦点を合わせているので、今のところこれで十分です。

このasyncとawaitに慣れていないようであれば、非同期メソッドとClimaモジュールであるDartでそれをどのように扱うかに焦点を当てたモジュール全体を確認してください。

しかし、これらすべてがあなたにとって良さそうであれば、次のステップに進み、新しいユーザーがnullに等しくないことを確認します。

そのため、実際に登録して戻ってきたユーザーを取得することができました。その場合は、ユーザーをチャット画面に移動します。

したがって、pushNamedを使用し、ChatScreen.idであるルート名を使用します。これはもちろん、チャットクラスからの静的変数です。

したがって、そのchat\_screen.dartをインポートする必要があります。

ユーザーがチャット画面に移動したので、ここから選択できます。名前でメッセージの送信を開始するときに必要になるため、現在のユーザーのメールアドレスを印刷できるようにします。

メッセージを送信者にタグ付けする必要があります。また、署名されたユーザーのメールを使用するよりも良い方法はありますか？

そのため、ここでauthパッケージをインポートし、これらのauthインスタンスのもう1つを作成します。

そのため、Firebase auth.instanceと等しくなり、getCurrentUserというメソッドを作成します。

そして、このメソッドは入力も出力もしませんが、サインインしている現在のユーザーがいるかどうかを確認します。したがって、以前に登録が成功した場合、このユーザーは実際に取得しますチャット画面内で、ユーザーという新しい最終変数を実際に作成し、それを認証オブジェクト.currentUserと等しくなるように設定します。

そのため、現在誰も割り当てられていない場合、これはnullになります。

ただし、誰かが登録している場合、または誰かがログインしている場合、これは現在のユーザーに対応し、そのユーザーの電子メールまたはパスワードを利用することができます。

これは非同期メソッドでもあるため、このユーザーを使用するためにフューチャーを返します。これを再び非同期としてマークし、この行が完了するまで待機する必要があります。

これで完了です。ユーザーがnullに等しくないかどうかを確認します。つまり、現在サインインしているユーザーがいることを意味します。

その場合、新しい変数、つまり新しいFirebaseUserを作成し、これを記録されたユーザーと呼び、現在のユーザーから取得したこのユーザーを記録されたユーザーに割り当てます。 nullでない限り。

これも失敗する可能性があるため、発生する可能性のある例外をキャッチできるように、これをtry and catchブロックでラップする必要があります。

そして最後に、このメソッドをどこかでトリガーする必要があります。

そして、私たちの状態が初期化されるときよりも、それをトリガーするためのより良い場所は何ですか？それを削除して、先に進んでget current userを呼び出しましょう。そして、現在のユーザーの取得を呼び出したら、ログインしたuser.emailプロパティをコンソールに出力して、プロセス全体が機能するように最近登録したユーザーに実際にアクセスできることを確認します。アプリをテストする前にやらなければならない最後のことは、この特定のログインメソッドを実際に有効にする必要があるということです。

そのため、Firebaseのプロジェクトに移動し、[開発]セクションに移動して、認証をクリックします。したがって、ここでは現在、サインインメソッドがありません。そのため、サインインメソッドタブの下に設定します。また、さまざまなソーシャルサインインや匿名サインインの幅広いアレイを使用できることを確認できますが、有効にするのはメール/パスワードです。

それでは、このトグルをクリックして有効にし、保存してください。

これで、実際にアプリをテストする準備が整いました。[実行]をクリックしてホットリスタートを実行します。

そして、準備ができたら、ようこそ画面にドロップされます。

登録をクリックし、最初のユーザーをangela@email.comとして登録します。パスワードは123456です。

ユーザーを登録するときは、パスワードも6文字以上にしてください。

Firebaseパスワードは少なくとも6文字の長さである必要があり、短いパスワードでは登録が失敗します。登録をクリックすると、チャット画面に自動的に進むことがわかります。これは、ここに入力したナビゲーターが原因です。

そして、チャット画面を開くと、メールが表示されるはずです。そのため、登録したばかりのユーザーがコンソールに印刷されます。

そして、Firebaseを見て、[ユーザー]タブに移動すると、ここで作成されたばかりの新しいユーザーが表示されます。その完璧な。

最初のユーザーを作成し、Firebase認証を使用してそれらを登録しました。

次のレッスンでは、ログイン画面のコードを更新することで登録したユーザーにログインする方法を見ていきます。

それで、それ以上については、次のレッスンで説明します。