このレッスンでは、スタイリングを改善するために、アプリの外観をさらに改善し続けます。

特に、送信者に応じてメッセージバブルの色と配置の両方を変更します。

しかし、私たちが取り組む必要のある他の問題もいくつかあります。

たとえば、登録画面とログイン画面のヒーロー画像が異なる画面サイズとアスペクト比を処理できることを確認する必要があります。

AndroidとiOSの両方でアプリを実行しているので、まず修正する必要があることを最初に示したいのは、特に後のiPhoneのiOSで、アスペクト比が少し異なることです。アンドロイド。そのため、垂直にはかなり多くのスペースがあり、水平にはほとんどスペースがありません。

つまり、Androidバージョンで登録画面に移動して、キーボードがその小さなアイコンからポップアップし、仮想キーボードまたはソフトキーボードを有効にすると、ボタンの多くが実際に画面から外れていることがわかります。 。

では、どうすれば修正できますか？

ヒーローウィジェットを実際に柔軟なウィジェットにラップすることができます。これは、「これは、それが占有しようとするサイズですから、約200ピクセルです」ということです。

しかし、それができない場合は、それについて柔軟で、画面の他の部分が見えるように小さくすることができます。

そのため、ログイン画面のヒーローを柔軟なウィジェット内にラップして、その根拠もカバーすることをお勧めします。

セーブを押してみましょう。エミュレータを見ると、キーボードがポップアップし、キーボードをポップダウンすると、画像を含むヒーローウィジェットが実際に小さくなります。実際のサイズ。

そのため、画面にこれらのひどい黄色と黒のストライプがなく、UIの一部が実際にオーバーフローしたことを示しています。

これが完了したので、送信者に応じてメッセージバブルの色と配置の両方を変更します。

これは、ユーザーエクスペリエンスの観点から非常に重要です。

私たちのアプリが複数のユーザーで使用するのがどのようなものか見てみましょう。

iOSシミュレーターでangela@email.comとしてログインします。それから、エミュレータに行き、新しいユーザーを登録します。これをjack@jackbauer.comと呼び、パスワードを入力して登録を行います。

これで、両方のユーザーが画面に表示され、ここにメッセージを入力して「Hello Jack」と言って送信すると、そのメッセージがここに表示されますが、順序は言及されていないことがわかりますこれは故障しています。これは修正します。

しかし、誰がそのメッセージを送信したのかを伝えるのは本当に難しいのですか？

このグループチャットには多くの人がいますが、すべて同じ人からのもののようです。

はるかに優れたユーザーエクスペリエンスは、このようなものになります。

ここで、私たちのメッセージバブルは、送信者に応じて異なって配置され、色付けされています。

また、メッセージバブルの隅が送信者を指すようにスタイル設定されていることがわかります。

これがレッスンの終わりまでに目指していることです。最初にチャット画面でのメッセージバブルの表示に取り組みましょう。

バブルを持たせる代わりに、スピーチの矢印のように少ししたいのですが。バブルになりますが、ここには小さな角があり、どちら側から来たのか、どの送信者から来たのかを示しています。

そのためには、その青いマテリアルの境界半径を分割する必要があります。

そのため、丸い丸い境界線だけでなく、実際に境界線を変更して、BorderRadius.onlyと呼ばれるものを使用することができます。

これにより、各コーナーに異なる境界半径を設定できます。したがって、おそらくおそらく円形の30の左上が必要であり、同じ30の円形の半径の左下が必要であると言えます。そして、私たちはついに一番下にも欲しいです

正しい

これも同じ30ピクセルになります。

そのため、左上、左下、および下に丸い境界線を持つ、より広い半径が必要です。

正しい

そして今保存していた場合、吹き出しには送信したユーザーを指す小さな先のとがった部分があることがわかります。

このようになったので、自分以外の他のユーザーからのメッセージの表示を変更できたらいいと思います。

そのため、たとえば、メッセージの色を変えたり、メッセージを右側ではなく左側に表示したりできます。

そして、小さな先の尖った矢印は、左上隅と、私たちから来たメッセージと他の人から来たメッセージを区別するために変更できるあらゆる種類のものを指している場合があります。

では、どうすればこれを行うことができますか？

ここに、loggedInUserと呼ばれるものがあることを知っていますか？

このFirebaseユーザーオブジェクト内に保存されている現在のユーザー。

つまり、このlogsInUserをFirestoreがある同じ場所に移動すると、メッセージストリーム内でアクセスできます。ここでできることは、これらの送信者の電子メールにアクセスしたら、currentUserを作成して、ログインしたユーザーの電子メールアドレスに送信することです。

ループしているこの特定のメッセージを送信した人のメールと、現在ログインしているユーザーのメールがあります。

そのため、これらが一致するかどうかを比較して、現時点でこのメッセージが実際に送信されているのが実際にアプリにログインしている人から送信されているかどうかを比較することは難しくありません。 currentUserがmessageSenderと等しいかどうかを確認するIFステートメントを作成するだけです。

まあ、それは彼らが一致することを意味します？

これは、メッセージがログインしているユーザーからのものであることを意味します。

それでは、どのようにしてメッセージバブルに変数を送信できますか？

たぶん、isMeと呼ばれるブール値の何かを持つことができ、メッセージが現在のものであるかどうかに応じてTrueまたはfalseに設定できます。

ユーザー。

したがって、このisMe変数があり、メッセージバブルを作成するときに初期化します。そして、IFステートメントに基づいて送信できます。

ただし、IFステートメントを使用する代わりに、条件を使用できます。 double equalsは、現在のユーザーとログインしているユーザーの電子メールの同等性をテストし、trueまたはfalseを提供します。

currentUserがmessageSenderと等しい場合、およびその場合は、isMeをtrueに設定する必要があります。

それがfalseの場合、isMeはfalseに設定する必要があります。

つまり、メッセージバブルのビルドメソッド内で、isMeというこのブール値にアクセスできるようになりました。

だから、私からのメッセージで構成されるバブルを作成するときを知っています。

では、どのように挑戦しますか？そのプロパティに基づいて、メッセージバブルの色を変更してほしい。

だから、それが私なら、バブルはその明るい青のアクセントを持つべきです。しかし、それが私でない場合は、代わりに白い泡である必要があります。

そのため、ビデオを一時停止して、このチャレンジを完了することができるかどうかを確認してください。大丈夫。

繰り返しになりますが、この3項演算子を使用する必要があります。

そして今、私たちはisMeの値をチェックしています。それが本当なら、私のメッセージは明るい青になり、それが私から送られていないなら、背景色を白に設定します。

そのため、保存ボタンを押してアプリを再起動すると、iPhoneにログインしている現在のユーザー（angela@email.com）によって送信されたすべてのメッセージが青色で表示されます。しかし、angela @ gmail.comやjack@jackbauer.comのような他のユーザーによって送信されたすべてのONEは、よく白です。

唯一の問題は、メッセージも白であるため、メッセージが表示されないことです。

それでは、先に進んで、他のユーザーが送信するメッセージバブルのテキストの色を変更してみましょう。これをblack54に変更します。

したがって、IFステートメントを削除して、TextStyleがある場所からすべてを白にする代わりに、isMeがtrueであるかどうかを確認し、それが白である場合は色を変更します。黒54。

次に変更したいのは、他の人が送信したすべてのメッセージを左に移動して、送信したすべてのメッセージを右側に保持することです。

そのため、ビデオを一時停止し、その方法を理解できるかどうかを確認します。

大丈夫。

そのため、メッセージにはCrossAxisAlignmentと呼ばれるこのプロパティがあります。そして、デフォルトでは終了に設定されています。つまり、右側にあります。これを変更して、CrossAxisaAlignmentに変更できます。

左側に並べたい場合は始めます。

したがって、アプリを再起動して再度ログインすると、他の人から送信されたメッセージが左側に表示され、黒のテキストと白の背景が表示され、見栄えがよくなります。

唯一奇妙なのは、この小さなコーナーです。

この小さなまっすぐなコーナーが他の人から送信されたときに左上隅のメッセージの中心を指すようにここに来たら素晴らしいでしょう。そのため、ビデオを一時停止し、最後の課題を解決します。

大丈夫。

そのため、境界半径と呼ばれるものがあり、それがそのポイントの場所を決定しています。

基本的に、左上、左下、右下が円形であると言っています。

これは、先のとがったビット、まだ正方形として残っている唯一のビットが上部であることを意味します

正しい。今私たちはそれを変更する必要があります。しかし、そうでない場合は、その場合、異なる境界半径を提供する必要があります。

したがって、再びBorderRadius.onlyを使用し、左下と右下の両方を丸い角にする必要があります。

そして、丸くない角でなければならないのは左上だけです。

したがって、丸くする必要がある最後のものは右上です。

ちょっと紛らわしいし、たくさんのコーナーがあります。

しかし、考えてみれば、それは理にかなっているはずです。

だから今、jack @ jackbauer.comとしてここにメッセージを入力し、送信をクリックすると、白いメッセージとして届きます。

しかし、ここにangela@email.comとして入力すると、青いメッセージとして表示され、すべてのバブルが正確に表示されるはずです。

良いもの。

それで、他に何が残っていますか？

他にどのような問題を修正する必要がありますか？

それでは、ユーザーエクスペリエンスについてもう一度考えてみましょう。

通常、チャットアプリでは、リストの下部に最新のメッセージが表示されます。

また、新しいメッセージが到着すると、画面が更新されて最新のメッセージが表示されることが期待されます。

言い換えると、リストビューが更新されるたびに下にスクロールする必要があります。

最新のメッセージが下部に表示されるといえば、チャットメッセージが時系列に並べられないのはなぜですか？

なぜすべての注文がとても奇妙なのか、なぜここに来るのですか？

これは、Firebaseデータベースをかなり前に初期化し、さまざまな変更を行っており、このモジュール全体でそれをいじっていたためです。

そのため、順序を修正するには、まずFirebaseに移動して、コレクション全体を実際に削除する必要があります。

そして、これはとんでもないことなので、彼らは実際にあなたにそのコレクションの名前をタイプすることによって確認するように頼みます。

アプリからすべてのデータを入力するため、このコレクションは空白のままにします。また、コレクションを削除するとすぐに、アプリもメッセージを消去して応答したことがわかります。

ここで、ジャックアカウントからの最初のメッセージを入力し、アンジェラアカウントからのメッセージを入力します。

この時点で、もう1つの問題があります。これは、メッセージがこのような奇妙な方法で注文されている理由です。

なぜすべてが一番上に表示されるのですか？

また、ここにメッセージを追加すると、ある時点で画面に表示される範囲を超えていくことがわかります。そのため、そのメッセージはすでにそこまで到達していることがわかります。

そのため、別のメッセージを追加すると、表示されず、スクロールすることもありません。

では、どうすればそれを実現できますか？

位置までスクロールさせるにはどうすればよいですか？

これを行う方法は、リストビューにプロパティを追加することです。

そのため、現時点では、パディングと子の2つのプロパティのみを持つリストビューがあります。

ここで追加できるプロパティの1つは、reverseと呼ばれるものです。

これをtrueに設定すると、リストビューがビューの下部に向かって固定されていることがわかります。

つまり、ここにメッセージを入力しても、それが何であるかは問題ではありません。

常に一番上に表示されますが、リストビューの一番下には常に、常に一番上にとどまるときの代わりに表示されます。

だから今、私たちは別の問題を抱えています。つまり、すべてのメッセージが追加されますが、それらは一番上に行くということです。

では、どのようにしてメッセージのリストを逆にできますか？

snapshot.data.documentsからメッセージを取得すると、これはドキュメントのスナップショットのリストであることがわかっているため、どのリストでも「反転」と呼ばれるものを使用できます。

これにより、リストの順序が逆になります。

つまり、今すぐ新しいメッセージを作成すると、それがリストに追加され、リストが逆になります。

したがって、送信ボタンを押すと、一番下に表示されていることがわかります。

したがって、すべての新しいメッセージは最後に届き、リストの一番下に表示されます。これは順番が逆になっていますが、スクロールせずに常に画面に表示されるリストビューの一番下にもあります。

今やらなければならないことは、コードのテスト時に以前にコメントアウトした部分を含むコードの最後の部分を整理し、getMessagesメソッドとmessagesStreamを含む必要がなくなった部分を削除することですveをmessagesStream widgetに置き換えました。また、onPressedからも削除できます。

鮮やかさ。

そのため、このレッスンでは、メッセージバブルとリストビューを構成することで、スタイルとユーザーエクスペリエンスを大幅に改善しました。 次のレッスンでは、チャットアプリの最後の部分、つまり認証とセキュリティに関するルールの設定に取り組みます。