大丈夫。

そのため、アプリを機能させることができました。機能はほぼ揃っています。

しかし、問題についてもっと深く考えましょう。

全体像を見てみましょう。

そのため、このレッスンでは、デザインパターン、アプリアーキテクチャについて説明し、Fluttterの状態管理について詳しく説明します。

しばらくプログラミングを始めると、これらの用語が頻繁に投げ出されるのを耳にするでしょう。「設計パターン、アーキテクチャ、状態管理」、そして質問は本当に私たちが本当に気にするのですか？

なぜそれが重要なのですか？

それはすべて複雑さです。

複雑さは、ソフトウェアのすべての悪の根源です。

そして、これを説明するものがあります。

エネルギーを利用しようと決めたとしましょう。

そこで、風車を作ります。

そして、英国で風が吹くたびに、それはほぼ毎日、風を捕らえ、そのエネルギーをミリングフラワーに変えたり、発電機をつければ電気をつくることができます。

これは実際、比較的簡単に管理できます。

また、YouTubeをチェックしていくつかの本を読んだ場合、おそらくこれらの1つを作成することもできます。

しかし、生産を拡大し、代わりに原子力発電所を建設したい場合はどうでしょう。

今、小説を見た人にとって、それは非常に複雑なタスクであり、あなたが考えなければならないことがたくさんあること、プラントの設計、安全装置、建築、物事がどうなっているのかがわかるでしょう実行されます。

したがって、複雑さが増すと、物事がうまくいかなくなる可能性も高まります。

そして、それについて考えるもう一つの方法は、あなたが店のスペースを借りることを決めて、あなたが正しいレストランを始めることを望んだならば、それが少し言うのに似ているということです？ママとポップの店です。それはただあなたと、料理人と少人数のウェイターだけです。

しかし、一晩で生産を拡大する必要があると言ったとしましょう。

そのため、小さなスタートアップを経営する代わりに、そこにある最大の企業を経営しなければなりませんでした。

スタッフを募集して整理し、一晩でAmazonまたはGoogleに変えるために、たくさんのお金を与えました。

あなたは、小さな会社で働いていたのと同じルールや同じことを想像できます。それは、何千人もの人々が働いていることを知っている大企業ではうまくいきません。

そのため、さまざまな企業がさまざまなアーキテクチャを持っている理由があります。

そして、ある程度の複雑さを感じて一定の規模に達したら、すべてがスムーズに機能し、スタッフができる限り効率的に作業できるようにするための優れたアーキテクチャが必要だからです。

したがって、大手テクノロジー企業を見ると、人々を作成するためのさまざまなアーキテクチャがあります。たとえば、トップダウンピラミッドのように、会社を管理するためです。あなたが従業員の個々のグループと言っている構造のタイプとそれらすべては、あなたが知っているように、相互に通信します。

したがって、ソフトウェアにも同じ種類のものがあります。

そして、少なくともモバイル向けの最も人気のあるアーキテクチャの1つは、Model View Controllerと呼ばれるものです。

そしてこれは、iOSアプリの開発でAppleによって最初に普及しました。

したがって、このアーキテクチャのコンセプトは本当にシンプルです。

データを管理するモデル、画面に表示される内容を管理するビュー、およびこれらすべてのコンポーネント間の仲介者であるコントローラーがあります。

したがって、データとロジックを処理するモデル、ユーザーインターフェイスを処理するビュー、およびメディエーターであるコントローラーが存在するMVCデザインパターンになります。

そのため、ユーザーがアプリの何かをタップすると、ビューのコードがこれらの入力イベントに関するメッセージをコントローラーに送信する場合があります。

次に、コントローラーはそれらの入力イベントを使用して、次に何をすべきかを決定します。そして、データを要求するためにモデルに要求を行うこともできますか？

ユーザーが連絡先アプリをタップし、Bobの電話アドレスを表示したい場合、コントローラーはモデルクラスにそのリクエストを送信します。

次に、モデルクラスがデータベースまたはデータストレージを処理し、最終的にデータをコントローラーに送り返します。

次に、コントローラーはそのデータを使用してビューを変更します。

そして、これはMVCパターンがどのように機能するかを見る非常に単純化された方法であり、大きなアプリ全体を特殊なスキルを持つことができる個別のカテゴリに分離するため、うまく機能します。

したがって、溶接部門があり、レーザー切断部門があります。こうすることで、誰もが自分のしていることを理解し、お互いのつま先を踏む必要がなくなります。

しかし、フラッターについて考えるとき、考慮する必要がある本当に重要なことは、命令型プログラミングと反応型プログラミングの違いです。従来のネイティブアプリは命令型コードで構築されているためです。

ですから、あなたが何をすべきかを指示する上司がいるようなものです。

「今日はあなたがしなければならない」、私は知りません、「この報告書を書いて、あなたは今それを書かなければなりません。

その報告書はどこにありますか？

何が起こっていますか？'

しかし、他の代替手段もありますか？

従業員が自分がしなければならないことを知っていたらどうでしょうか？

彼らがタスクのリストを持っていたり、出し入れのトレイを持っていて、特定のミドルマネージャーによって指揮されていない仕事を続けたら、人生はそれほど良くないでしょうか？

これは、リアクティブプログラミングがどのように機能するかを示しています。

そして、そのように考えると、これらのオブジェクトが正しいようになりますか？

リストビューまたはボタンと言って、作成したらすぐに最初に伝えます。これが人生におけるあなたの役割です。ボタンはタッチイベントに応答するためのもので、ユーザーがあなたをうまくタップすると、これが変更される状態になります。または、テキストウィジェットがある別のケースでは、テキストウィジェットが特定の変数の状態を追跡するのに役立つ可能性があります。その変数が変更されると、テキストウィジェットは正しく再描画する必要がありますか？

リアクティブ。

したがって、この場合、設計パターンについて考えるのではなく、状態管理についてもっと考えています。

アプリの状態をどのように管理して、アプリの複雑さを増やせるようにしますか？ FlutterのデザインパターンやFlutterのアーキテクチャや状態管理に進む前に、重要なことは、異なる開発者が状態とデザインパターンを管理する最良の方法について異なるアイデアを持っているということです。これらのことについて強く。

そのため、状態管理や設計パターンに関するブログ記事やビデオに出くわすたびに、誰もが何を使うべきかについて非常に強力なアイデアを持っていることがよくわかります。

そして、覚えておくべき重要なことは、使用する必要のあるすべてを一度にすべてカバーする方法はないということです。

なぜなら、私たちがFlutterについて話しているだけでも、状態を管理できる方法はたくさんあるからです。

また、Scoped Model、Provider、BloC、Redux、MobXなどについてこれらの言葉が振り回されているのを聞くかもしれません。プログラミングを学び始める人々は常に、学習するのに最適なプログラミング言語は何かをよく尋ねます。

実際、最高のプログラミング言語はありません。

ハンマーの使い方を学んでいるようなものです。ハンマーは素晴らしいツールですが、ハンマーではすべてを行うことはできません。

試してみて、ハンマーで木片を切ることを知っていますが、最終結果は非常にきれいに見えません。

プログラミング言語と状態管理についても同じです。

結局のところ、状態を管理するこれらのさまざまな方法は、単にアーキテクチャの青写真に過ぎず、要件に応じてさまざまなアーキテクチャが存在する可能性があります。

そのため、アプリの特定の要件を満たすため、より優れた特定の設計パターンまたは状態管理方法がある場合があります。

あなたが持っているさまざまなニーズに基づいて、イグルー対ツリーハウス対水上にある必要がある家を作成するための建築パターンと同様に、あなたはあなたの家を建てるのに異なる建築パターンを使用するかもしれません？

木で高くする必要があるものは、湖の上に立つ必要があるものとは対照的に、非常に異なる建築要件を持っています。

現在、建築パターンを選択する際に考えられる他のこともあります。それはスタイルに基づいています。

そして、これがおそらく最大の競合の原因です。なぜなら、あなたが知っているように、私たちは皆、これらの家のどれが最もよく見えるかについて異なる意見を持っているかもしれないのと同様に、プログラマーは、アーキテクチャを作成するための最良のスタイルに基づいて異なる意見を持っていますアプリまたはプログラム用。

最後に覚えておくべきことは、Flutterはまだかなり新しく、成長と進化を続けているため、考慮すべき成熟の側面があるということです。

人々はまだ、状態を管理するための最良の方法は何かを練っています。

たとえば、今年のFlutter状態管理でのGoogle IOトークを見ると、ほとんどの人はプロバイダーパッケージを使用することをお勧めします。

そして、少なくともパッケージの理論がどのように機能するかを理解するには、次のレッスンに進んで、それについてさらに説明します。