Flutter Zero to Hero — Katalog Widget & UI

Transkrip referensi — folder sumber `1_flutter_zero-to-hero`

12 bagian · dibuat otomatis

Daftar Isi

1_key-concepts

1. Create a project

Once you have installed Flutterer, the next thing you need to do is to create a new flutter project.

And this is what we will do in this video.

So you will open the terminal.

So I will go inside.

My Mac will open terminal and maximize this thing so you can see something.

Next we need to create the project.

What I like to do is to always create the projects at the same place.

So it's inside my MacBook, inside my development.

And this is the flutter installation and this is my project.

So we will create one inside this.

So I will say CD and I will CD the day, the Lopman folder and then CD the projects folder.

Great.

So this is where we are now I will say flutter create and we will call this one my flutter app or flutter app.

You can write anything you prefer, but just make sure that you don't have spaces.

And next you will say run perfect.

So this will create our flutter project.

We need to open this inside Visual Studio.

So I will go inside.

I will say open and then I will go inside development projects Subjects.

And I have the flutter app right there.

So you will click open.

Great.

So now we are inside our flutter project and we are ready to run the application.

So we'll go inside the library and the main.dart.

And then you will press this little button if you are on Visual Studio Code.

And this will start the application, it should take like 30s to launch.

All right.

But if you are on the Android Studio or something else, you can go in the terminal and you can write flutter run and press enter.

So if you press enter this will launch the flutter app.

The next thing I need to tell you is if you are on Visual Studio Code, you can also create some sort of configuration.

You can do this on Android Studio also, but on Visual Studio it's pretty simple.

You just click this little thing and now you have all your configuration, and you can launch the application by going and clicking on this little thing instead.

It doesn't really matter how you launch it.

At least the only thing you need is to launch flutter, right?

So this is the basic application.

And on the next video we will start building a real application.

Right.

So that's about it.

And I'll see you in the next one.

Bye bye.

2. Widget inside Widget

Okay, so the next thing you need to understand is flutter is really a widget inside widget.

Inside widget.

This is the key concept that once you understand this you will see building with flutter will be so simple.

So let me show you an example.

Inside the Main.dart, which is the entire code required to build this application right now, we will change a couple of things.

So for example, if you change the you have pushed this button many times, blah blah blah, you can press this little thing which is called the hot reload.

So the lightning bolt or whatever.

And this will restart and just update the screen.

Right.

So you can also just write something else and press Ctrl S to save.

And this will automatically update the screen.

So I will always do this.

The next thing you need to understand is flutter is widget inside widget.

So I have a column widget inside.

I have an argument called children which I will put inside a list of other widget.

So this is why we can have one widget and another over under because we use a column widget.

So this is what I need you to understand is flutter is widget inside widget.

Another example is we have the center widget which will put everything in the center.

And then we have the column which will allow you to put two widget the text and the number.

All right.

So if we take another text and we just copy copy paste right under like this, you will see that we have two numbers.

Great.

Another thing we can do is maybe we can go inside the text and we can go and press, I don't know, text style or just style.

This is an argument, by the way, but it doesn't really matter if you don't understand right now.

We will explain everything later.

So I will say I will use a textile widget like this.

And you can you can click the arrow to have more information about it.

I prefer not to click it because it kind of hide everything in the screen.

So I will press textile and I will press ctrl space because this is the way you can see all the arguments inside.

And I will use by example color and we'll say colors dot blue and I will save.

So now you can see that the text is blue because inside the text widget I've put a text style widget which change the style.

So flutter is really like that widget inside widget inside widget.

And in the next video I will show you how to understand and how to put all these pieces together.

And yeah, that's about it.

So I'll see you in the next video.

Bye bye.

3. Capital (Widget) vs Lowercase (argument)

Okay, so how do you put a widget inside a widget?

Or how do I know when do I need to use tile, or when do I need to use Chill child or when do I need to use children's?

Like it's so confusing, right?

So what I will need to do is by example, I will remove the column.

I will delete all of this garbage right away.

And what we'll do is we'll just put a text widget like this.

And inside I will press control space.

And you can see that we can put many many arguments.

But the text widget is a little bit funky because if you go over you will see that you need to put right away the string data.

And then you can put some arguments.

So this one is a little bit funky.

So we will put a string, we'll say hello and save.

But now how do we put arguments.

How do we understand to put widget inside widget.

To put arguments what you need to do is you press the comma and you press control space.

This will show you all the possible arguments.

So for example we used style previously.

So I will put my arrow I will click this little thing and I will see that the description of the thing.

All right.

So I will click on style and I will put a text file like we did earlier.

But the thing you have to understand at this point is a widget is a capital letter.

An argument is a lowercase letter.

The first letter.

And after this you do this forever.

You see that after the argument we have another widget which have the capital letter and you guessed it, if we go inside and we use an argument by example, the color color is a lowercase letter.

And after this again capital letter colors dot amber.

So you understand right now that flutter is widget inside widget.

But you understand also that in order to put another widget inside another one, you need to use an

argument.

Arguments use lowercase letter and widget use capital letter.

So like this.

You have some sort of visual idea of how to use flutter.

Right now this is the key concept.

Once you understand this, you are ready to jump and create your own app.

And this is what we will do in the next video.

So I'll see you in the next one.

Bye bye.

4. Types

Okay, so the next step you need to have or understand is how to create the data type with flutter.

Once you have this it's really like let's go.

And I don't know what I mean by that.

But anyway, so um, what you will need to do is you will say string and you will say name is equal to by example, um, mammoth or whatever.

It doesn't matter.

So this is a string.

After this we have the int which, which will be a number.

And this will be equal to one.

We have the double which will be nobody care.

And the double is like uh number with a comma.

And after this we have a list.

This is how you create a list.

So my list is equal to the square bracket.

The square bracket will let no flutter that it's a list.

And you can put many items inside each item need to be separated with the comma like this.

All right.

So the last thing we need to do and talk about is a map.

So how to create a map.

My map is equal to.

Now we use the curly bracket.

We put the semicolon at the end everywhere.

And next you need to put by example a string and, um, what else?

A string, string, whatever.

It doesn't really matter for now.

Just so you have seen how to create each of them.

Uh, first time later inside this course when we need them.

I will explain more if needed.

All right.

So this is how you do it.

Last thing with the map.

If you want to put the type for the map, you can write it just like this.

This will be a map string string.

But if you have something else, for example, I don't know, uh, something weird you can use dynamic and dynamic will be all the type.

Kinda.

Right.

So by example, if I use this, I put my comma to create another thing, the key two.

And we change this for a number for for for now this will work because we have the dynamic which will be either a string or either a number.

So this is how you use the type with flutter.

Later we will do like loops and stuff like that.

But we don't need it right now.

So we will not do it right now.

That's about it.

And I'll see you in the next video to start building a real application.

That's it.

Bye bye.

5. Types cheat sheet

int – whole numbers: int x = 5;

double – decimal numbers: double y = 3.14;

bool – true or false: bool isOn = true;

String – text: String name = "John";

List – list of things: List<int> nums = [1, 2, 3];

Map – key → value: Map<String, int> scores = {"A": 1};

var – Dart figures out type: var z = 10;

dynamic – can change type: dynamic a = 5; a = "hello";

final / const – can’t change after set: final pi = 3.14;

null – means nothing: String? maybe = null;

6. Mouse over

Okay, so I know we were supposed to start building like, apps and widgets, but I need to tell you this thing.

It's so crazy as a tip that it will help you build with flutter like crazy fast.

So I don't I don't understand why people don't tell about this more often.

It's just so good.

So for example, if you want to create another argument, we will.

Right now what we will do is we will work on this little debug banner.

I hate it because I don't think it's beautiful.

I think it's ugly.

But if you want to have a debug banner to know if you are in debug mode, you can keep it, but I will remove it.

So we will go inside the material app and I will write show debug banner or something like this.

Uh show debug banner.

Yeah.

This one.

And I will select this argument.

And if you go over the argument you will see that flutter will give you the answer.

You need to put a true or false value.

So I will say I want this as false.

And I will press Ctrl s to save.

And you see that the banner is not available anymore.

If you like it, keep it.

Personally I don't so we will do again this little tricks but with another argument.

So let's go inside the color scheme and inside the color scheme we have the argument and we have the widget color scheme.

Inside this one we'll go inside the arguments I will press comma Kumar control space and we will use the brightness argument.

So this one I will press enter.

And now we need the answer.

So we'll go over brightness.

And you see we need to write brightness dot light.

So this is a widget.

So I will do exactly that.

Brightness.

But instead I will say dot dark.

And if you save you will see that the application is not dark.

But another thing is you can see that the code is a little bit weird.

So if you put a comma after each argument or each argument in widget and save, this will automatically reboot the code like in a good way so you can see it if it doesn't work for you.

When you press Control Save.

This means you need to like format document.

So you need to press this.

All right so that's about it.

So now you know that when you press you put your mouse over an argument.

You have always the answer.

Flutter is giving you the answer.

So just use them so you understand that flutter is widget.

Inside widget.

You understand that when you put your mouse over the argument, it gives you the answer, and you understand

that the arguments start with a lowercase and the widgets start with a capital letter.

So right now you have like all the tools to understand what we will do with flutter.

So I see you in the next video to start building.

Bye bye.

2_basic-layout

1. Container

Okay.

So let's talk about container.

A container is just like an empty box that you can fill with anything.

Like you can decorate it.

So I will go over the text widget.

I will right click and I will say refactor.

Then I will use the wrap with a container.

But I have some bug on my computer.

I will right click.

I will say refactor and I will select wrap with the container.

But you can see also we have the the center padding size box column row.

We will do all of them after like in further video.

So I will say wrap with a container after this.

Inside we will go in the arguments control space and I will say color.

The color will take a color widget.

So I will say colors dot and we will use maybe red and let's save.

So this is the container the red color.

So what we can do also is we can change the height for 100 and zero and the width for 100 and zero.

So let's save.

So this is our container.

All right.

After this what we can do is we can change the corners of the the container.

I will go inside the decoration Argument and if you go over, you will see that this takes a decoration.

But this is a little bit misleading because it's better to use the box decoration.

Perfect.

So after this inside the box decoration we will put the corners or the border radius.

Again, if we go over you will see that this said border radius geometry.

But it's better to just use the border radius dot circular.

And then you just put like 20 5.0 inside and let's save.

And now we have an error.

So in further video we will show you how to fix the error by yourself.

Like it's a two step formula.

It's super simple.

But for now I will just fix it pretty quick.

If you don't understand, don't worry because we will talk about it later.

So what I will do is you can see that we have the color is equal to null or the decoration is equal to null.

And right now we put a color and a decoration.

So flutter is like a just put one of them.

And after this after this it's also said box decoration color color.

So you can change the color inside the border the box decoration.

Okay I know this means a lot of things and maybe it's confusing, but I will just take this.

I will cut it and I will paste it inside and maybe it will make more sense for you.

All right.

So now we have the rounded borders to 25.

And that looks pretty cool.

Can we change.

Can we do other stuff inside this.

The constraint padding I can talk about padding.

Uh color.

Yeah.

Let's use padding.

So I will say the padding go edge and set that all.

This is what you put inside.

If you go over padding you will see that it's called geometry.

It's better to use the edge and set that.

All right.

After this I will use the value maybe 50.0.

If I save, you will see nothing change.

The reason is it's like kind of invisible the padding.

So what we will do is we will change the height in the width.

And I can show you something cool at the same time.

It's called the double dot infinity.

All right.

This will tell flutter I will put my comma.

This will tell flutter take as much place as possible.

So inside the body argument or inside the body of the scaffold take as much place for the height.

And after this we will do the same thing for the the width.

We will say double this infinity and let's save.

Okay.

So you can see that the padding does again nothing.

But it's because the padding is inside.

Like this.

If we change the padding for margin, you will see that now we have the 50, the 50 margin.

But if we do it for the padding, this means what is inside this thing.

So let me show you.

We will take another container.

I will copy this entire container like this and I will go inside the child.

I will change the text for the container.

And now I will paste another one and we'll change the color for maybe orange.

And let's save.

So you can see that the hello.

Maybe the hello is a little bit uh, I will put black.

Let's save.

So now you can see that we have a margin.

We have a no no, no margin.

We have the padding and we can put a margin also like this.

Here we go.

So this is everything about container or pretty much everything you need to know about container.

I'll see you in the next video.

Bye.

2. Column

Let's talk about column.

This one is pretty cool because most of the time you will need to put widget one over the other because currently it's all one inside the other.

So what you can do is instead use a column widget.

If we go inside the body, we have the container right there, and instead of this container I will delete what is inside.

I will delete all the child like this and save.

Okay.

So what we will do is we will wrap this container with a column.

I will right click and I will say refactor.

And then I will use a wrap with a column.

All right.

So as as you can see usually a container will take a child widget.

Okay.

We don't have it right now but you could have a child and then is what you put inside.

But the column is pretty much the same thing.

But it's children instead because you can put multiple child inside.

We have the list.

This is the list.

And then we have the widget container.

So what you can do is you can copy this container and paste it right under and save.

And now we have a problem.

And it's because we use a infinity height inside a column.

So a column is infinite and we put an infinite.

So it's like not it's not like possible.

Anyway so I will put Ctrl J and you can see inside the debug console.

Also we have some sort of problem with the height has size.

All right.

So I will remove this.

And what we'll do is we will just remove the height like this and we will save.

Great.

So we have one container and we have another one right under.

And we have two type of padding.

So we will remove all the paddings in the margins and padding margins.

And let's save.

So if we save now we have another problem.

It's not working because we don't have any height.

All right.

So a lot of problems.

But this is how you learn.

So height we will put 100 dot zero and we will put another height for this one and save.

So we have our two containers inside a column.

All right.

That's pretty cool.

But I have another thing to show you.

Instead we will replace the two double dot infinity.

You can press Ctrl d.

Usually this will select the next thing and you will.

Right.

Maybe 100 dot zero.

Let's save.

So we have our two containers.

After this what we can do with them is maybe we can put them in the center.

So let's use the column.

And what we will do is we will go inside the arguments.

We will say the main axis alignment.

Main axis alignment dot center.

And now the widget are inside the center.

So let me explain you what is the main axis argument and the cross axis alignment.

Because we can also change this one.

For example end and it will be at the end.

And I will show you why it is not right now anyway.

So a lot of things to explain.

This main axis alignment is for by example the column.

The column is like this and the main axis is vertical.

All right.

So this is why when we put it in the center it goes in the center.

If we put it at the end it will go at the end.

All right.

But why this one doesn't work.

The reason is simple.

If we take the column and we wrap it, refactor, wrap with a container, and then we change the container for the color.

Colors dot blue.

And let's say this is the container.

That's the reason.

But if we take by example the container and we put it at width, double that infinity.

Oops that infinity and we save.

Now you can see that the cross axis element is working and it goes up to the end.

And if we change it to start it goes to the start.

Because now the column takes all the width, because by default the column will take as less place as possible.

All right.

I know this is a lot of things.

We will redo this exercise in the in the next video because we will do the row instead of the column.

So you will see it's a little bit different.

But anyway.

And another thing we can do is we will delete.

Maybe we'll put this as a black 38 so it's less aggressive.

And next we can change maybe the width remove this and save.

Great.

So now we have our column.

And what we will do is I will remove the cross axis alignment.

We will save and we will change another argument called the main axis size.

So if we do this we can change because by default, as I told you, the column take as much space as

possible for the vertical.

So we can change this.

We can say the main axis size the minimum.

So now it will take as less space as possible.

And the main axis alignment will not work anymore because it doesn't take all the the height.

All right.

So you can remove this and save.

So this is a way to put it in the center.

Also you can remove this and it will go back up.

All right so this was a lot of things about the column.

In the next video we will talk about the row and that's about it.

See you in the next one.

Bye bye.

3. Row

Okay, so let's talk about Rome.

Instead of using the column, we will replace this just by writing Rho and I will save.

Great.

So now you can see that this is the row.

We have the color black 38 which define the color of the row.

And after this we have two container.

So let's change a couple of things.

Let's go and try to put this in the center.

So what we could do we could change the main axis size.

Um is it main axis size.

It's not inside the row.

Sorry.

So yeah it's the main axis size like this.

And we'll say main axis size the minimum.

Try to think what it will do before running it.

So for example if I click on it this will change the main axis size to the minimum.

So all the black will be reduced.

So this will go in the center.

So let's save.

Great.

After this what we can do is we can change the cross the main axis alignment.

And if we say main axis alignment dot.

That, uh, let's say start.

It will do nothing for now.

So let's save it.

Do nothing, because the main axis size is minimum.

So if we remove this now, it will go to the start.

But I think by default it is by start.

Yeah it is by default.

So what we could do, we could change it for center.

And let's save or end and save.

Great.

So there is also another thing called the space around or space between space evenly.

Let's use this one.

This one can be pretty cool sometime.

So yeah you can play with that.

And one last is the cross axis alignment because the row is horizontal.

So the main axis is horizontal.

And the cross axis of the row is the invert which means vertical.

All right.

So we will say the vertical axis we want it to be cross axis alignment dot start.

But this should not change anything.

Does it cross axis the start.

So I will save again.

Yeah.

This does nothing because it was just the main axis previously, so that's why it moved.

But if we just remove this and we put save and then we put this cross axis element cross axis element dot start.

It was you see nothing changed right.

So um that's good.

But how do we make it change?

The reason is because the row, the height of the row is like the minimum as possible.

So we can change this by going in the container.

So the parent widget and we can change the height to put double that infinity.

And now it will work if we save.

You see now it worked for the cross element.

So I know this was a lot of things and maybe it's confusing for you, but it's pretty simple.

It's just if you have the row what is the main axis of the row.

It's like this.

And what is the cross axis?

It's like this.

And if you go with the column, the main axis is like this.

And if you go with the the column the cross axis is horizontal.

So that's about it.

See you in the next one.

Bye bye.

4. Center

Okay.

So now let's talk about the center widget.

So you can see that we have a center that is wrapping everything right now.

But why is this not everything in the center in this case.

Right.

Let's figure out why.

First we have a height of infinity.

So let's remove this.

This might help a little bit.

Great.

So now we are going to the center.

After this we have a row.

But the row is saying that the cross element is going to the start which means um, this should do nothing.

So if I remove it, it should do nothing.

Yes, sir.

Perfect.

But why?

This is still not in the center.

The reason is the row.

Take as much space as possible for the main axis alignment.

So if you go inside, you will see that the main axis size is maximum.

So let's change this for main axis size main axis size dot minimum.

And let's save.

So now everything is in the center.

So this is the goal of the center widget.

But if you change the argument within other widget inside this will not take action anymore.

All right.

So this is the center widget.

What else can we do.

For example if we go in the container we want to put a text inside and we want it to be in the center.

So what we could do is we could write the child to go inside the container.

And next you will say center widget.

Let's click and then we will go inside child again because this will go in the center and we will write text and then we will put, I don't know, uh blue blue blue blue blue and let's save.

So now we have the text in the center of the container.

All right.

So this is how to use the center widget.

I don't think there is anything else inside.

We have the some width factor.

Height factor I never use them so it's not really useful.

That's about it.

And I'll see you in the next video.

Bye bye.

5. Images

Okay, so how do you add an image inside your flutter app?

For this, you have two ways to do it.

First, you can either use a URL or you can download the image and put it inside your project.

We will do both ways.

So first I will find an image on the internet wallpaper.

I will right click on it and I will say open image in a new tab.

Great.

So after this you will go at the end of this URL and you will search for something called jpg or PNG

or any image format.

After this you can copy all of this.

And if it's something weird then maybe it will not work.

So just now, you know after this we will go back inside the code and we will replace the scaffold body.

So we will change all the container.

And one tip for you is I always click on the bracket to know where the next bracket is.

You see I always click like Container.

I know it's right there because it's highlighted.

So I will take the the center delete all of this and then we'll use the image dot network.

Perfect.

Now we can delete the source.

And we can put um something like this the URL.

And we can put the comma format document save.

And now we have the image inside our application.

Great.

So what is the next step.

The next step is maybe to put the image on the full height.

So we will use the height argument.

We'll say double dot infinity and let's save.

But this is still not like all the all the image like all the screen.

If we want all the screen, we need to use another argument called the fit.

And the fit will have the box fit dot multiple options.

So container which is currently the case, the cover or the fill fill is a little bit weird.

So we will use the cover.

And now let's save.

Great.

So this is the first way.

By using a URL with the image.network.

So let's do the other way around.

Instead we will create a new file.

And we will call it assets.

Inside this we will create another one called the image.

And finally inside assets we'll create another one called fonts.

Because later inside this course we will learn how to add your own fonts.

And so inside assets let's go back inside the the internet and I will download this image.

I will say right click save image as and then I will save it inside my image.

So you can see this is my project asset inside flutter app image.

And I will save it inside this.

Let's change the name for something like background bg jpg GPG save.

Okay, now we can close this and we can go inside image and see if we have our background jpg.

Great.

The next step is to go inside the image.network and change network for asset.

And after this you can change the path which is inside.

And what we will put instead is the path for the image which is assets slash image slash bg dot png jpg.

Great.

But if you save, you will see that this is not working because we need to tell flutter that we have a path to image.

So you will go inside the pubspec.yaml and inside this one you will scroll down until you find the section asset, and you need to make sure that everything aligned perfectly.

Otherwise it will not work.

So you will go over the A and you will delete twice.

Like this.

Delete.

Delete.

After this you will do the same thing.

You will go after and you will delete twice.

Delete, delete.

Great.

So you will remove this and call it assets.

Slash.

Slash.

Image and slash again.

So now you have access to all the image within this file.

And you will click this.

You will save and you will click this little button flutter pub get all right.

If you don't have this on Visual Studio Code you can open the terminal and write Flutter Pub.

Get like this and it will do the same thing.

So I will close this and close the pubspec.yaml.

And now if we save the application, we have the background.

If it's still not working for you, just close the application and relaunch the application.

And now it should work.

So this was pretty much everything about image.

So that's about it.

And I'll see you in the next one.

Bye bye.

6. Extensions

Okay, so if you are using Android Studio and not Visual Studio Code, this video might not be for you.

So you can skip to the next video right now because we will talk about extensions with Visual Studio Code.

So you can see that when I put my path of my image, I have a little image preview.

This is because I have an extension called Image Preview.

All right.

So you can download and enable this if you want.

After this we have the better command.

Better comment is just a way to put colors for your comments because normally this is a command.

But if you put this now the comment is red.

And if you put like question mark, now it's blue and stuff like that.

So you use it if you want.

After this I have the awesome flutter snippet.

This one is pretty cool because you just write like a shortcut and it will write a lot of code for you.

So we will use this.

Inside this course you will see I will use a lot of time the shortcuts and we will save a lot of time.

So you can enable this also.

The last one is the material icon theme.

So I have already this installed and it's inside the explorer.

You can see all my files.

We have some sort of icons before each files and I think it's cool.

So that's about it.

You can enable them if you want and I'll see you in the next video.

Bye.

7. Stack

In this video we will talk about the stack widget.

So most of the time you will have a child inside a widget.

So you will be able to put a widget inside and nest all the widget together.

But by example this time the image that asset don't have any child.

So how do you put a text over it in the center?

For this you will need to use the stack widget.

So what you will do is you will go over the image, you will right click and you will say refactor with a column like this.

If you do a column and you put a text after.

So right now it's because we use the height infinity.

I will remove this and uh, yeah, if we save now, we should have something.

Let's reload the application.

Yeah, we have our image.

The reason was inside the column.

It's a widget infinite.

And then we have another infinite.

This is why it was not working.

So what we will do is we will say text, and inside we will say, I don't know, flutter or anything.

You can put anything.

And now you see that the text is under the image, but we want it over the image.

So instead of using a column, we will use a stack widget and it works just the same.

It uses a children inside which you have a list of widget separated with all commas.

So now if we put the comma right there also and save, we have the text flutter right there.

So what you could do is you could put the text in the center by right clicking on text, refactor and

wrap with a center widget.

All right.

So let's save.

And now yeah you can see that this is not working because it is going inside the center of the the application.

So what we could do to fix this real fast is we could define a height of maybe 300 for each of them.

So this one and the center, maybe we could wrap it, say refactor with a size box and put a height

of 300 also.

And now you see that it's in the center.

So this is how you play with the code.

All right.

So that's about it for the stack widget.

Um, and I'll see you in the next one.

Bye bye.

8. Padding

Now let's talk about another widget called the padding.

So it's pretty much like the container because if you remember by example, if we go over the stack,

we wrap this refactor wrap with a container inside the container.

We can put the padding argument and then we say dot all.

And we put a value maybe 50.0.

So we will have a padding all right.

But there is also the widget just padding.

You could replace the container with padding and save and it will do just the same.

The difference is padding is only for padding.

But if you use the container you can see that you have many other arguments and stuff that you could put inside.

So if you only need a padding, use the padding widget because it will be like less expensive for the application.

So that's about it.

And I'll see you in the next one.

Bye.

9. Sized Box

In a previous video, I've used a widget called Size Box without any warning and never explained to you what it was, so I will explain inside this video.

So a size box is pretty much like a container, but only for the height and the width.

So you can see if we go inside the arguments we only have the width.

So the container again is the one that has pretty much everything.

It can use a padding or it can use a height and width.

But if you only need to use the height and the width, you should use the size box because again, it's less expensive than the container.

It's the same thing with the padding.

Okay, so the size box is just that.

It's an empty box that you can put a height and a width and then a child inside.

That's about it.

See you in the next one.

Bye.

10. Icon

Okay, so how do you add an icon inside your app?

So for this you will go inside the section app bar of the scaffold.

And we will work inside one argument called the leading.

All right.

And inside the leading which is the start of the app bar.

What you will do is you will put an icon like this.

And inside you use the same logic as the colors you remember.

When you use a color with the flutter, it's a colors with the s dot a color.

Then you will do the same thing.

So we'll say icons with the S dot and then you have all your icons if you want to see it.

You click the little blue thing like this and you select the one you want.

So we'll just say notification or anything and we will save.

So you can see that we use an icon inside our app bar.

Great.

So we could also change the color by going inside the arguments.

You press control space.

And then you use the argument color.

And you say colors with dot uh amber or anything and you save.

So that is how you use an icon.

We can delete it because we will not need this later.

Uh, but we will reuse this inside another widget that I will show you inside the next video.

That's about it.

See you in the next one.

Bye.

11. List Tile

Okay, so another widget I want to show you is called the List Style widget.

So we will go inside the stack and we will change the list style.

I will use the list style and inside we will say title.

And the title takes a widget inside.

So if we go you see that this takes a widget.

So we'll say text and I will remove this because it takes a lot of place.

And I will say text.

And inside you can write anything.

It doesn't really matter.

So this is a list style.

But after this we will change the color because I want you to see what is the list style.

So the title color will say colors dot red and I will save.

Okay.

So you cannot see it right now because we have an image and the the list style title color has a weird a weird thing that the color goes behind every stack.

I know it's weird, but it is how it works.

So what we will do is I will delete the image and you will see this is the style.

So because we are using a padding I will remove the padding also.

Now you can see the entire list style.

It's how it looks.

But what we will do also is we will put an argument called the leading and the leading.

What we will put inside is an icon and we will use icons dot and inside we put anything.

So join for.

All right.

And another thing you can use inside is the trailing.

Inside the trailing.

What you will put is by example a text.

And again like this.

And you will see it at the end.

Perfect.

Another thing again we can use is the on tap.

So this will create a function okay.

If you don't have the option you just press Ctrl space and you will have the option of using both like this.

You use this one.

And now you can put a function inside.

And now you can save.

So that's about it.

And I'll see you in the next one.

Bye.

12. Print

Okay, so we created some sort of empty function.

But how do we see when we tap on it?

So what we will do is we will use a simple print function like this.

And you will write, I don't know, clicked on or anything.

So I will put my semi-coma, my semicolon, I will save and I will open the terminal inside the debug console.

Now you will see that when I press on this style, boom, you have clicked on.

And then that's about it.

Every time you clicked on it you will see a number count.

So this is how you print something with flutter.

Very simple stuff.

And you can use this to debug sometime.

That's about it.

See you in the next one.

Bye.

13. Wrap

Another widget I want to show you before we move on is called the wrap widget.

And this one is pretty cool.

Okay, so we have talked about the list style, and the list style is a little bit like a row.

So we will replace the list style.

And we'll go inside the body.

And we'll replace all of this with a row widget.

Inside.

We will put the children which will be multiple widget back to back, and we'll say text and I will say anything and put a comma.

Let's save.

So we have one text widget.

Let's add another one.

And let's add like maybe five and save.

You see how we have an error right there is because we don't have enough place to display all the widget inside the screen.

And this is the thing with the row widget, it can happen.

So if you want to make sure that this doesn't happen, you can replace this with the wrap.

And now let's save.

And you see how everything goes down if you don't have enough place.

Another thing is let's try to create something even bigger.

So maybe this, this, this, this and save you see that even if the text widget doesn't enter, it will still put it down.

So this is a really helpful widget to remember.

The wrap widget.

If you have problems with the row then you can use this.

That's about it and I'll see you in the next one.

Bye.

3_master-layout

1. Material App & Scaffold

It's now time to talk about the master layout.

So we have talked about a lot of little widgets that you can put together to create the app.

But what about like the big picture, how the skeleton of the app work?

This is what we'll do.

So what we'll do, we will go a little bit crazy and we will delete everything.

All right, let's do this.

Let's just keep the the main run my app.

My app.

Okay.

So what we'll do is we will start from scratch.

First thing first there is like three things we will talk about.

We will talk about the state less.

Then we will talk about the material app.

And then we will talk about the scaffold.

Okay.

So you have a mental picture of what we'll talk about.

The first thing we'll do is we will write the state less widget.

So if you just write Sta or Sta you will find stateless widget.

You click this one and this will build automatically the widget for you.

After this you will write my app because it just needs to be the same name as this.

So if you change my app for something else, you will need to change these also.

Great.

So this is a stateless widget and it's the first part we will talk later about what is the stateless widget in stateful widget.

But as a quick explanation.

Stateless means that the screen will not change, and a stateful widget means that when the user click things, then the screen refresh with new stuff.

It doesn't matter if you understand right now, but what we will talk about in this video is more about the material app, because later we will talk about this and more about this.

So let's use the material app.

You will replace the container for a material app.

So this is like the theme of your application.

So you can see that right now it's all black.

And you remember the debug banner.

What we'll do is we will remove this.

We will say show debug banner False unless you want to keep it.

After this, we just have a black screen.

So what we need is we need the home of our application, which is an argument.

And inside we just put a widget and the widget is the scaffold.

The scaffold widget is the third part which I told you is the skeleton of the app.

So this is more like the theme of the app.

And this is the the the skeleton.

All right.

We will work on this on the next video.

But for now let's work on the material again.

Let's change the theme.

And if you remember previously we had something called the color.

Um, was it color?

It was theme.

Yeah.

So we have the theme of the application and the team takes the theme data.

So we'll say theme data.

And then inside we will say dot from is it from the theme data.

Inside we put the color scheme exactly.

And the color scheme what you will do is you will say color scheme.

Color Scheme that from seed.

This is the easiest way to do it right.

There is like ten ways that you can put colors inside your app.

I just find it that the color scheme that from seed is the easiest and that's about it.

So you will remove the color and you will say colors dot.

Let's create our application color teal because I think it's cool.

I will put all my commands I will save.

And now you see that we have some sort of teal white.

Right.

So if you put red you will see that the color background is a little bit different.

So now we have some sort of theming inside our app.

But also it will be cool to have our application dark.

So I will say brightness and I will say brightness dot dark.

I will put my comma and save.

So now our application have some sort of theme of colored t teal and the brightness dark.

On the next video we will start working on the scaffold.

That's about it.

For the third things, I will delete them and I will see you on the next video.

Bye.

2. App Bar

Okay, so the scaffold is the cool part of the application because it's the skeleton and it's where you start to see your application take life.

So inside the scaffold we have something called the app bar argument.

If you go over you will see that the app bar suggests something called the preferred size widget.

But it's just better to use the app bar because everything is built up for you.

Inside the app bar, you have the title, which you will use inside a widget.

You can see right there and we will use the text widget inside.

We will just call it flutter.

You can put anything.

We'll put flutter map and now you have the title.

If you have the title on the left side is because you are using more like Android stuff.

If you use Android, you can fix all of these by using by example.

It's center title and you can say true.

Or if I say false, I will go on the left side.

Okay, so this is how you use it.

The difference is with flutter iOS and Android have some differences.

All right.

And the differences, uh, if there are I will try to tell you.

And it's very rare.

It's like 1% of widget that have differences.

So you should not have that big of problems.

All right anyway so we have the app bar.

Next what we have inside is other arguments called the leading.

We have used this one previously.

We will put a simple icon, but an icon, an icons dot.

What do we put?

Login I don't know, let's save.

After this we will put another thing called the action argument.

So this one have a list and it will be a list of multiple things that goes at the end of the app bar.

So what we will do is we could put an icon again or let's use a text widget.

Doesn't really matter and we can put anything and save.

So you can see that this is at the end.

And you could just copy also the icon and paste an icon also and save.

You have multiple things you can do with that.

So after this inside the app bar you can press Ctrl space and it will change the color.

So say background color and you can say colors dot teal.

And let's save.

So this was a lot of things about the app bar.

And it's pretty much the thing you will need.

There are other arguments but you don't really need them.

It's because uh yeah they are a lot.

One one maybe you could use is the bottom.

The bottom will allow you to put like more stuff behind, uh, under the title.

But yeah, I don't really use it.

So that's about it.

So I'll see you in the next video.

Bye.

3. Nav Bar

Okay, so the next thing we need to talk about is how to add a navigation bar at the bottom of your app.

So what we'll do first thing first is we will clean a little bit the application because it's kind of ugly.

So inside the app bar we will keep the title.

We will remove center title I will say true and we will remove the action and leading and background all of these and save.

Great.

We will also delete delete this and it's better.

Next thing we'll do is we'll go inside the scaffold.

We have the app bar and after the app bar we have the bottom navigation bar.

So we'll use this.

And if you go over you will see that it does take a widget.

And the widget we'll use is the navigation bar.

So the navigation bar will take destinations right which is a list of widget.

What we'll do is we will remove this and we will put a list and we will say navigation bar destination or navigation destination like this.

Now we need to give it an icon.

So we'll say icon icons dot home I will format document so you can see better.

I will put all the commands Comes and save.

After this, we need a label.

The label will take a simple string.

So we will say home and save.

Now we have an error.

So I will put all my commas also and we will need to fix this.

So again I told you that inside this course I will show you my two step to fix errors.

This, uh, will just fix it real fast if you understand it's good.

If you don't, it doesn't really matter, because I will explain you later.

So what we'll need to do is we will just read this.

The destination dot land need to be bigger or equal to two, which means this thing needs to be have at least two items inside.

So I will copy the navigation navigation destination and I will paste another one and save.

So now we have two items which make the application work.

We will change the second one for profile and the icon will be uh I think it's person.

And let's save.

Great.

So now it looks better.

But after this you can see that it's not clickable.

Nothing happened when you click on it.

We will fix this later by using the stateless and stateful widget.

But for now what we could do is we could just change it uh, manually.

So for example, if you go inside after destination inside the navigation bar, you, you press Ctrl space and you have the index maybe selected index.

So if you put zero it will be the first one.

And if you put one then it will be the second one.

So you see we do it manually.

So another thing we could do is on selected on destination selected.

If you press Ctrl space you will see we have this value that returns something.

So this function will return the value that we have clicked.

So if you go over value you will see that this take an int.

So I say I will say int just to define the type right.

So it's more clear.

And after this inside the function we will use the print function that we know.

And we will print the value.

And I will put my semicolon.

And now let's save if we open the terminal or the debug console.

And I will restart the app.

All right.

If we click on home you see we have zero.

And if I click on profile we have one.

So now we have access of this index but it's still not updating.

We will do this later inside this course.

That's about it.

And I will just delete the print and we can move on to the next video.

I'll see you in the next one.

Bye.

4. Floating Action Button

Okay, so the next widget is the floating action button.

This one is pretty cool.

If you remember when we had our first app, we had a little button on the side where you were able to click on something like that.

So what we will do is we will recreate this button.

We will go inside the scaffold again, and inside the scaffold we will create the floating action button argument and use the floating action button four action button widget.

Great.

Now the Onpress you will delete and you will press Ctrl space and you will use the second one, the second function.

And now you can format, document and save.

You see the little button right there.

That's great.

Now we need to define an icon for this.

So we will go inside.

And in the argument we will say child.

And the child takes a widget.

If you go over so you will say a Nikon and inside icons dot add or anything.

So now you have a little plus sign a little button.

Again if you want to make sure that it does work, you can use a print function and print anything inside.

Put the semicolon and check inside the debug console.

So when I press on it, I need to save first.

When I press on it I have this print okay.

So this is how to use the floating action button.

Another thing I need to show you is what about you need two of them so you can wrap this floating action

button.

Because it does take a widget and it's not required to be exactly this.

So you can wrap it with a column.

So refactor wrap with a column and then inside you can just paste another floating action button and save.

And you can see that this is not working.

And it's because the column we need to change the main axis size to main axis size dot minimum.

So main axis size dot minimum and save.

Great.

We have two of them.

And we could also use a size box widget with the height of maybe 10.0.

And use this.

So this is how it could look.

All right.

So that is how to use the floating action button with flutter.

And I see you in the next video.

Bye.

5. Drawer

All right.

So let's talk about the dryer widget.

This one is pretty cool.

I don't really use it as often, but it's a it's there in multiple apps.

So I need to show you this.

Inside the scaffold you will go and you will right the dryer argument.

Inside the dryer you will use the dryer widget.

Pretty simple.

And if you save just by writing this little code now when you press on this you will see we have a dryer.

Pretty cool right after this.

Inside the dryer, what we will do is we will change.

Maybe the child.

The child is what is inside.

And what we'll do is we will use a list style.

Let's use this.

Let's use a list style and use a title with a text saying I don't know, logout and let's save.

So now if we reopen the thing, we have the list style right there, but you can see that it's still weird at the place.

It's a it's at a weird place.

So what we need to use is a dryer header.

So what I will do is inside the child, I will wrap this list style.

I will say refactor and I will wrap it with a column.

And inside the column I will use a dryer header like this.

All right.

And the child will be a simple text or anything saying dryer.

And I will put my commas save and format document.

And now you can see that it looks way much better.

All right.

So this is how you can create a dryer with flutter.

Pretty simple stuff.

So that's about it.

And I'll see you in the next one.

Bye.

6. Safe Area

All right, so let me show you one more widget, which is pretty cool.

So you remember when we opened the drawer and we had like the logout inside the the timer?

It was pretty weird.

So what we'll do is we will delete the dryer and I will save again.

And now you see we have the logout.

But one thing you can use with flutter is called the safe area.

So if I wrap the dryer like this and I say refactor and I wrap it with a widget, call the safe area and save.

Now you see that the dryer will not go on the place where the the timer or the notch camera is.

It will never go there.

All right.

This is pretty cool.

And another thing you can do is you can actually I will go over the safe area.

I will wrap it refactor and I will remove this widget.

And instead we will put it maybe over the scaffold.

Let's see what happens.

So I will say I will go over the scaffold.

I will right click refactor and I will wrap this widget with a safe area.

And let's save.

So now you can see that even the bottom and the top which have some stuff, is not usable in the app anymore.

So this is how you use it.

So I will re delete this.

And also the dryer.

We will delete it also.

Let's save.

We will remake our app look clean and we don't need the dryer anymore.

The floating action button.

We can delete this.

We will not use this neater.

So this is I will put this as zero and we will fix up the bottom navigation bar later also.

So this is our application.

So let's work and let's start building a real app from this.

That's about it.

See you in the next one.

Bye.

4_fixing-bugs-2-step-process

1. Type 1

Okay, so before we start building a real app, we need to make sure that you are able to fix bug by

yourself with flutter.

So when you build your own application, you are able to handle problems and build your own app yourself,

right?

So what we will do is we will talk about the two types of errors inside flutter.

And I will show you how to fix both of them.

So right now it's the first type.

And the first type will happen.

And you will have these things like that.

You will have like a red curly line under a text.

What you need to do is you go over and you will see expected to find you will find the error right there.

So it's expected to find a comma.

The reason is we cannot put like an argument and then a widget without separating another comma in order

to display another argument.

So this was the first type of error.

If you have this or if by example you you are missing one bracket.

So let's delete the bracket.

You will see that if we go there it's saying expected to find a bracket.

So you need to put it back.

All right you will see sometime it's less obvious.

Maybe if we do this you see we have a problem.

But the only thing you have to do when you have the red problem, you go over and flutter should usually

give you the answer.

Right.

So this is how to fix the first type of problems with flutter.

So I'll see you in the next video to show you the second type.

Bye.

2. Type 2

Okay, so to fix the second type of problems, usually you will have a red screen on your application and sometimes you will not have the screen.

But I will show you both ways are the same.

So this is the second type.

How to fix your errors with flutter.

So what we will do is first I will delete one navigation and I will save this query.

This create an error right.

If you don't have the red screen and you know that your application is not like responding the way it should be, what you can do is you can open the terminal and you will see inside the debug that you have some sort of problems, right.

So when it's not working, just check in your debug console and check out if you have errors.

Right now the problem is telling me like the first link is saying go to the navigation bar.

So usually when you have a problem it will give you a lot of links, but you can see that many of them is not even like project of mine.

Right?

It's like flutter or dart programming.

So what you will do is you will find something that you created by example, the navigation bar, the dart.

Let's find out what is inside.

So this is not ours, but you can see that we have a problem with the destination dot length which is less than two.

All right.

So right now if you go you see that the main.dart this is my file.

So usually it will be the first, second or third link.

Third link.

And you just click on it and it will tell you exactly where is the error inside your code.

So you can see that when I click on it boom, I'm redirected to the main and exactly to the place.

So I know that I need to add another navigator like this.

So let's do this again, but with another problem.

So let's say I replace the title with a container a container and I say height and I say double that infinity and I save.

Now let's restart the application.

And you see that we have some sort of errors but it's not visible in the application.

You see, we have an error about the size of the the widget because we have an infinity inside a navbar, which take like maybe just this size.

Okay.

So let's fix this.

Let's go at the top and let's find the link that bring us to the problem.

So you see that we have some sort of box constraints.

Let's see where it is.

It's not this one.

It's not this one.

So let's find, uh, we have toolbox.

Toolbox.

You see, sometimes it's less obvious, but it should be at some place.

Isolate.

Okay.

So it's not available.

Sometimes it can happen.

So when it's not available, then you have to find it yourself.

So what you will need to do is you will say box constraint force an infinite height.

So you need to figure out yourself like why you could either take this and go on, um, on the internet and search about this like flutter box constraint force, infinite height.

And it will tell you that you have like an infinite inside a something which is not infinite.

So we will delete this and save.

And now if I restart the application, you will see that we don't have the error anymore.

But let's create another error by simple colors and let's say colors dot red.

And now we will do another thing is a decoration box decoration.

And now let's save.

You see that we have the error.

But what we can do is we can restart the application and we can click on the first link.

So you see we have the container which is not this one, but we have the main.

If we click on it it will tell us exactly where is the problem.

So you see that sometime the link is not available.

But if it's not available you have all the errors.

And then you can search online on Stack Overflow.

This is the way to fix your errors with flutter.

All right.

So you need to take attention to the debug console, look at the screen and look at pretty much this place and find your answers.

That's about it.

See you in the next one.

Bye.

3. Null Safety

Okay.

So we need to talk about null safety right now.

What is null safety.

So first thing first we need to remove the error.

So I will put back the title like this.

I will do Ctrl Z.

And now we have our title back.

And the application is working perfect.

So we need to talk about null safety.

What is null safety.

Let's create a variable called string.

Title is equal to flatMap.

And I will put my semicolon.

And now I will replace the title for this variable.

And now let's save.

You see that everything works.

But what if we tell flutter this value can be null.

All right, nothing changed.

But flutter is telling us.

All right.

This value can be null.

So you need to tell me, like.

Are you sure that it's not null currently?

So what you can do is you can put this and you say to flutter, I'm sure that this value is not null.

So let's restart and you can see that nothing changed.

But this value is now nullable.

So what happened?

If you remove this and you restart the app, you will see that you have an error.

And this is because it's null.

And then you tell flutter I'm sure that this is not null.

So this is how you play with null safety.

You can define nullable value.

And if you want to say like okay, I'm sure that this value is not null, you define this.

Sometimes you will need to create null value.

So this is how you do it.

But most of the time if you are able then just create values like this.

And you should have the minimum amount of errors.

So I will delete this.

And this was how to use null safety.

Pretty pretty basic stuff.

So let's restart this.

We have our name.

Great.

So that's about it.

And I'll see you in the next one.

Bye.

5_states

1. Stateful

Let's talk about the states.

So there is three things we will talk about in the further video in this one.

And it will be the stateful widget.

So the stateful the state less and then it will be the set state.

So what are these.

The stateful widget is a way to tell flutter that the screen can refresh this.

So this can refresh.

This can't refresh the screen.

And this is to refresh the screen.

Whatever I'm tired.

Okay.

So you can you can refresh the screen with the stateful.

You cannot refresh the screen with the stateless.

And this is uh this is telling flutter I want to refresh this screen right now.

So let me show you a real example.

First thing first we will create a new class and this one will be called this.

Uh, we will use a stateful widget this thing.

So click on this and then you will write it maybe, um, my hap or my home page?

My home page.

Whatever.

So you will use this, and then we will replace this placeholder for the entire scaffold widget.

So we will take this scaffold and we will cut all of this.

And we will paste it right inside this.

So let's save.

We'll remove the constant and we will save.

Great.

And now we just need to put in the home the name of this.

So my home page.

And now we can save.

And you will see that nothing has changed yet.

But we can now use a stateful widget and we can reset this.

Or we can refresh the screen to change the navigation bar.

So that's pretty cool.

So what you will do next is you will go over the override.

So there is always two place where you can put your variables and stuff either over the override or after the build.

Okay.

So it's hair or hair.

And right now because each time we refresh the screen we rebuild the entire widget.

So if you put your variable there.

It will always rebuild, so you don't want to put it there right now.

So what we'll do is we'll say int and this will be the current index.

And we will start at zero like this.

And we'll use the current index inside this selected index like this.

So now if we refresh.

Still nothing changed.

But we will be able to use this pretty soon.

OOP I don't know why this is not profile profile.

And I will put this as a person.

All right.

So it looks better.

So what we'll do next is we will go inside the function and we will change this.

We will say the current index is now equal to the value that we receive from the destination selected.

So now if you refresh again nothing will change.

And the reason is, is because we are using a stateful but we are not resetting or refreshing the screen.

We need to use the set state.

So what we'll do is we will say set state and inside this estate we will show what we want to refresh.

So now if we use home or profile, we are now changing page, which is which is pretty cool.

And so this was it.

Another thing I want to show you, which could be interesting, is if you go inside the body and now you wonder, like, how do I display one or another widget?

You could say by example, current index is equal to zero.

If it's the case, then display I don't know uh center widget with the text hello or the text one.

And if it's not the case then.

And don't worry if you don't understand because we will do conditions a couple of times.

So it will make sense eventually.

This is just a conditions.

If the current index is equal to zero then display this.

Otherwise display the second one.

So now if we save you will see one.

And when you click on profile you see two.

So this is how you can create stuff.

All right.

So that's about it for this one.

And I'll see you in the next video.

Bye.

2. Recap

Okay, so I know this was a lot of things to learn to create this.

Just this little thing, right?

So what we will do is inside this video, I want you to relax and I want you to watch what I will do, okay?

Because I will redo everything.

I just want to make sure that you understand each step of the process.

And sometimes it's fun to have a a biggest view of what we have created.

So I will show you what we did in four simple steps.

So I will delete everything and what we will create.

First thing first is let's define the steps.

We need a material app.

Then what we need is we need a scaffold.

After this what we need is a nap bar.

With a title.

And then we need a bottom navigation bar whatever.

So it's just four steps.

So because we will need to use a set state inside the navigation bar, then we will need to use a state full widget.

All right.

So this is what we'll do.

We'll say stateful widget.

And then we will call this my app.

And then inside we will remove the container and use a material app.

And we will save.

Now you see that we have the debug banner.

So we want to remove this.

What we will do is we will say debug banner false.

And then we need to change the theme of the application.

Do you remember how to do this.

It's inside the material app.

You will say theme and then you will say um, you just go over.

You will see theme data.

Yeah.

Theme data.

And then you go inside and there is something called the color scheme.

And you use the color scheme dot from seed.

So color scheme dot form from seed.

So in this way you just have to give one color and flutter will do all the stuff for you, which is really cool.

So you will say colors dot teal or the color you want.

Then I will say comma, comma and save.

Great.

So now I will restart the app and you see that we still have nothing because we don't have any scaffold.

So inside the theme inside the material app, what we need to do is we will create the home argument and say scaffold.

So this is the skeleton of our app.

Next what we want is this to be dark.

So I will say brightness.

Brightness.

Dot.

Dark.

Comma.

Save.

And this is what we have.

Great.

Next we need a navbar.

So we'll say a bar argument.

And the widget is a simple app bar.

The argument title will require a widget which will be a text widget, and it will be flutter map.

Let's save.

Great.

The next thing we need is a simple bottom navigation bar.

So we'll go inside the scaffold and we'll see bottom navigation bar.

And then we will see navigation bar.

And the destination we will put a list and then we will save.

Now we need to have two navigation destinations.

So navigation destination.

The icon will be a simple icon.

And then the icon home.

Icons home.

And then the label will be let's put home.

Home.

All right.

It's terrible my typing home.

And now we need another one.

So I will just copy this copy paste.

And I will say profile.

And then I will say person.

Great.

So we just need to use this and change the index.

So we will go inside the bottom navigation bar or the navigation bar widget.

And we will change the selected index.

And this will be a selected index variable that we need to create right there before the build.

We don't want it after the build because otherwise it will be rebuilt every time we clicked.

So it will be selected.

Index is equal to zero.

Finally what we need is to onchange.

So on destination selected change.

We change the value and we give it a type just to make it more clear.

And then we say selected index is equal to the value.

And that's it.

So now we can refresh the app.

And we have it's not working because we don't have a set state.

We need to put a set state.

And then we put the selected value inside set and restart the app.

And now we have something working.

Great.

So you see how it's not that complicated.

We have only four things the material app, the scaffold, the app, the app bar and the bottom navigation bar at the end.

So that's about it and I'll see you in the next one.

Bye.

3. Split your Widgets

Okay, so with flutter, the goal is to create the smallest widget possible.

So you will need to split your widgets inside different files.

And I will show you how to do this.

And the reason is by example, when you click the set state and you refresh this little button to just refresh the screen, you don't have to refresh everything, because right now when you click on it,

this will go to the build and refresh the material app, refresh the scaffold and refresh everything.

And this will slow down your application.

So what you need to do is split into different widgets.

This is what we will do.

We will go inside the library and I will click and I will say new folder.

I will say widgets.

So this will be a splitted widget that goes inside this we will create another file and this will be the navbar widget.

So I like to call the name widget at the end.

So I always know like what it is right.

So you will go inside navbar widget and you will create this one.

We will see state full widget.

Great.

We will call this one the nav bar widget.

And then we need to use the import m.

So I will say import m.

If you don't have import m it's because you don't have the extension.

Uh it's just a shortcut.

So you will need to write the entire thing.

Otherwise great.

So now we have our nav bar widget.

We can delete the container and we will paste what we had in the main.

So we will go inside the main and we delete all of this.

I will cut it instead.

So I will cut this.

And I will say nav bar widget.

So nav bar widget.

So now this will redirect to the widget we created nav bar widget.

Great.

And you will paste this thing okay a couple of things.

First when you define the name of these files, use all lowercase letters with this little thing to separate them.

Great.

But when you create the class, use the capital letter and try to use the same name.

So nav bar Navbar widget.

Navbar widget.

Okay.

The next thing you need to do is you will need to put the Selectedindex.

So I will go inside the main and I will cut this thing and I will paste it inside navigation bar before the override.

So over the override remember to place.

You can put your stuff over the override or after the build.

After the build it will be rebuild so it will not work.

So let's refresh.

And now you see that our application is still working.

But we only we only reset or refresh the this part of the application instead of refreshing the entire my app, which is crazy great.

So that's about it.

And another thing you might ask is what about now?

If we want to have inside the scaffold a body, if we have a body and then we have some pages inside, how do we have the value which is inside the navigation bar widget?

How do we have access to this information.

And this is what I will show you in the next video.

See you in the next one.

Bye.

4. Widget tree

Okay, so now we have a problem.

We have the data inside the navigation bar widget.

But we need also to have the data inside the main.

So first thing first what we'll do is we will separate the scaffold and I will create another folder.

This one will be called views.

And inside views we will create another folder called pages.

Okay.

Inside the views also we will create maybe another file.

This one will be called the widget three.

Oops.

Widget three dot dart.

This will be like where you start your application and then you will go inside pages.

So let's create other pages.

We will say this one will be called the home page dot dart.

And then I will move this on the side so you can see better.

And then I will create another one new file.

This will be the profile page dot dart.

And you will see after we have created all the pages and the view, then I will show you how to use the data and so you can use it inside any pages you will see.

This is pretty cool.

So we have the widget tree.

This one will be a simple stateless widget.

And then we will need to import the material.

So import m import m.

Great.

We will need to change the name for widget three.

Okay.

And then we have inside what we will put.

So we'll go inside the main.dart.

And what we will remove is the entire scaffold.

So let me remove this.

I will cut this and I will say widget three.

Do we have it.

Widget three like this.

Perfect.

So this will redirect.

If you want you can press Ctrl and click on it.

This will redirect to the widget tree.

So now we can put the scaffold paste it inside.

And we need to have the import of the navbar navbar.

So I will say quick fix.

And then I will just say import this one.

All right.

So now we can go in the main again we can delete the comments and we can delete the unused reference.

Great.

So let's restart the application.

Everything should still work, but we still have the same thing on the screen.

Let's do the next step.

The next step will be to create another stateless widget.

So this one will be maybe a stateless widget.

We need to create another import again.

Import m.

Great.

And now we'll change the name for profile page.

Great.

Inside this one what we'll do is we'll create a center widget.

It's just to see something on the screen.

So I will say center.

And then I will use a child which will use a text widget.

All right.

This one will be profile page just like that.

Great.

So we can copy all of this.

I will say ctrl c and then I will paste it inside home page.

So let's go inside.

Let's paste it.

And then we will replace the name for home page.

Simple as that.

And we will replace this also for home page.

So we see the text home page.

Great.

So now we can close all things and we can reopen the widget.

Three.

So this is where we will handle the data.

All right.

So the next thing we need to do is we need to create a variable pages for example a list a list of widget.

So this is just the type inside is just the type of what will be inside the list.

So this will be the pages.

And this is equal to a list because we tell flutter it's a list.

And inside we put widgets.

And the widgets we will put are the two pages we created.

So the home page and then the profile page.

Great.

So now we have access to these two pages.

The next thing we need to do is to go inside the scaffold and in the body like this.

And we will say pages dot element at.

And with this we will be able to define either zero which will be the home page, or either the value one which will be the profile page.

All right.

So this is pretty much what we had to do for the view.

Now let's talk about how to create the data and pass the data.

All right.

So we will go inside the widget three.

And inside this one we need to access this information.

So I think we will do that in the next video.

Just to separate I will do this video maybe uh creating this structure of this application.

And in the next video I will show you how to handle the data.

So you have the two things separate.

That's it.

See you in the next one.

Bye.

5. Value Notifier

Okay, so how can you handle the data from anywhere inside your application?

So right now we only have the selected index available inside the navigation bar widget, but we need it inside the widget view in order to display either the first or the second element.

So let's do this first thing first we will go inside and the widget.

We should put it inside the view.

So let's put widget inside the view I will say move and everything should be handled automatically.

So that's good.

And next we will create another folder inside the library called the data.

Right.

So let's use data.

And then we will create another file.

This one will be the Notifiers dot dart.

This is where we'll put all our notifiers.

And there is two things you need to remember when you use Notifiers or when you want to handle data like this.

First you will need to use a value a value notifier.

The second one is you will need to use a value listenable Builder.

Okay, so the value notifier, this one will be the one that holds the data.

And then the value listenable builder, it will listen to the data.

So if the data change it will change.

And in this case you don't need the set state.

All right.

So pretty simple stuff.

Let's do this.

Now what we'll do is we'll create this value notifier a value value notifier I will press this will automatically import the material.

So we have the value notifier.

Let's call this one the selected page.

Selected page.

And I used to I like to call notifier at the end also.

So now I know that this will be a value notifier because it always ends with notifier.

And next we need to create a value notifier.

And inside we will put the value which will be an int either 1 or 0.

So we'll say zero as default.

And we put our semicolon.

So now we created our value notifier.

We can use this.

But you can also define the type.

Just to make it more clear, this is a int.

Okay.

Next we need to go back inside the widget tree or inside the navigation bar widget.

And we need to wrap our navigation bar with the value Listenable builder.

So I will delete all of this.

I will cut it instead.

And I will say value value listenable builder.

Inside the value listenable, you put what you created inside the notifier, which is the selected page notifier notifier.

Like this.

So I will use this.

It is the selected page notifier.

And then for the builder, what you will need to do is you delete this and you press control space.

Flutter will tell you what you need to put inside.

So we will use the second one, this one with the curly bracket.

Inside this one you will need to return what you had previously, which was the navigation bar at the end.

You need to put the semicolon, otherwise it will not work.

So semicolon.

And now let's save okay we have access to this selected page notifier Which is this value the this thing.

But we will rename it the selected page so it makes more sense.

Okay.

So we can delete the selected index.

And we can delete the set state because we will not use this anymore.

And the selected index is now the selected page which is this value coming from the value builder which

use the selected page notifier inside the notifier.

This thing.

All right.

So after this what can we do.

We can change the selected page.

And in order to to change this value the selected page notifier you need to call the real name.

So selected page notifier dot value.

So this will change the value which is inside this thing.

All right.

And next you can just put this value to the value that we receive from the on destination selected.

So you can say like this.

Put the semicolon.

And now uh we don't have the we don't have the value anymore.

So everything should be fine.

We can remove the stateful widget by putting a stateless widget instead.

So what you how you do this?

It would be nice if you just were able to right click and say refactor and put like, oh, apparently you can do it.

Oh, that's even better.

I didn't even know it.

So that's cool, because usually I was just doing like from the override to the override, and then you delete the stateful to stateless and it does the same thing, but you can just refactor.

That's cool.

And after this what you can do is uh, now it should work.

So let's restart the application and see if everything still works.

We have the profile.

It does work indeed, so that's fine.

But now we can access.

If you understand, we can access this selected page notifier from anywhere inside the app just by using this.

So we can go back over the widget three like this.

And we can wrap this thing, the body with the value builder.

So I will delete this and I will say Value Listenable Builder.

Inside this one we'll need to put the selected selected page notifier.

The thing we created in the notifier and the builder will delete this and say control space.

And we'll say flutter will give us the answer.

So we'll use the second one.

And after this you can say I return and you return what you had previously.

The page at element at.

Great.

So now we own we are only missing one step.

And it's to put the value inside this.

So you could directly put value like this.

You could rename it selected page.

Selected page.

It could be a better practice maybe.

So let's do this.

And that's it I think.

So let's, uh refresh this.

And now if we click on profile now we have the profile.

And now if you click on home you have home.

So this is how you handle the data with the value listenable builder and the value notifier.

So if you remember you use two things value notifier and Value Listenable Builder.

That's it.

This one will hold the data and this one will be able to listen to the data.

And you don't need a set state with this.

So that's pretty cool.

That's about it.

See you in the next one.

Bye.

6. Challenge

Okay, so I know this was a lot of things to remember.

So what we'll do is a little challenge okay.

If you want you can just watch and see if you understand.

Otherwise you can try it by yourself before.

So the goal of the challenge would be to create by example.

If you go in the widget three you see how we have is it it's inside the Main.dart you see how we have a brightness that dark, but we can change this for light and save.

And now we have a light mode.

So it could be cool to have some sort of boolean that will either say it's true or false, but you will need to use the notifier for this.

All right.

So let me just show you.

We will say by example is dark mode notifier notifier.

And we will set this as false when we start the application.

Or we can say true instead because it will always start like this.

And we need to change this for a boolean.

Okay.

So this is what we will need to do.

You will need to create some sort of logic inside the main in order to display either the brightness light or brightness dark by using the notifier.

Okay, so we can do this right now.

And there is maybe one thing that I haven't told you.

And it's about how to create buttons.

All right.

Because we will talk about this in the next module.

You will see it's all about user input and it's probably one of the best modules because this is where your applications start to take life.

Okay.

You have buttons, your screen change, and you create a real app.

So for now, I will just show you one widget which is called the icon button.

So we will go inside the widget three.

And inside the widget tree we will go inside the app bar and we will replace the trailing.

Or is it I think it's actions.

Yeah.

Inside the actions we will put one widget called the icon button.

Okay.

Inside the Onpress you will delete and you will press Control space and you will select the second one the empty function.

And then the icon will be icon.

And then you say icons dot light mode.

Maybe.

Let's try this.

So we have light mode.

We put all our commands and now we have this.

So your goal is when you press this little button it change for either the dark mode or the light mode.

And if you want to go even further than that, you can also change the icon for dark mode like this.

So every time you press this icon, change and also the entire screen, right?

If you don't know how to do this, then you can watch the next video.

I will give you the entire solution.

Otherwise you can try to do it by yourself.

Uh, anyway, it's up to you.

You decide and I'll see you in the next one for the solution.

Bye.

7. Solution

Okay, so this challenge might be pretty tough for you.

So if you were not able to do it, just watch the solution and then try to do it by yourself after.

And you will see at one, at one point you will be able to do it by yourself and you will see how you progress with flutter.

All right.

So let's start right now.

I will show you the solution for this.

So we were inside the notifier and we created the dark mode notifier.

Let's use this inside the widget three first.

So I'll go over the icon I will cut this and I will say Value Listenable Builder.

The value listenable is the is dark mode notifier.

And the builder will press Control Space and select the second one.

Then we will be able to return the thing we had previously, the icon dark mode.

And let's put the semicolon, refactor and save.

This is what you should have.

Okay, so nothing has changed yet, but we have access to the is dark mode value.

So I will copy this.

I will change the name.

And now we will use this information in order to change the icon.

So we can go inside the icon and we can say is dark mode.

If it's the case then display the icon.

Dark mode.

Otherwise display icons dot light mode.

Oops.

Light mode and let's save.

So now when I press on it, nothing happened because we need to also create some sort of button.

So I think it was the icon button.

On press on the on press.

What we'll do is we will say the is dark mode notifier dot value is now equal to the invert of what it was.

So you can say you can take this again and you can say this will be the invert like this.

So yeah, this was maybe not really easy to to see coming, but if you have already worked with some programming you were probably able to to figure it out anyway.

So let's do this.

And now if we refresh, you will see that when we click on it it does change the icon.

So let's do the same thing.

But for the light mode inside the main we will go over the team data.

I will select all of this.

I will cut it.

And what I will say I will say value.

I'm not even sure if I will be able to put this inside inside the team.

No, we absolutely require something else.

So what we'll do is we will wrap the entire material app.

So we'll say this I will cut it and we'll say value listenable Builder.

And then it will be is dark mode notifier.

Then the builder control space.

You select the second one and then you put a return return and you paste the material app.

Great.

Now we have access to the is dark mode.

I will copy this and rename this for is dark mode.

And then we can finally use this Inside and say is dark mode, then brightness dot dark otherwise.

Darkness dot brightness dot light.

So let's refresh the screen.

And now we have this.

So when we press on this we have the light mode and the dark mode.

So maybe we could reverse this logic.

We can go inside the widget three and we can say light mode.

And then we can say dark mode.

And maybe it will make more sense.

So when we press on the sunshine it goes to white mode and dark mode.

So this is how we use the is or the value notifier inside flutter.

So we have done a second exercise.

If you were not able to do it yourself it's really understandable because this was kind of tough was pretty tough even though.

So try to do it again.

Try to do it by yourself.

Once you're able to do it by yourself, you really get it, and then you can move to the next step.

That's about it.

See you in the next one.

Bye.

6_user-input

1. Text Field

Okay.

So now let's talk about user inputs.

This will be all the widgets that the user can interact with.

By example an icon button or a button or a text field where you enter your email by example.

So we will learn about this.

And the first one is the text field widget.

So let's go inside the profile page.

And we can close all the the other tabs.

And we can just go inside profile page.

Perfect.

For this one we will replace the center widget.

We will delete this and we will write a column widget instead.

After this inside you will press control space to see all the arguments.

You will use the children and inside the list of children we will write a text field widget.

So you can put your comma save.

And this is what we have.

So you can enter text and that's it.

So how do we use this information and how do we modify this to make it look good.

So first thing first, we'll make it look good.

We will say by example decoration and this will take an input decoration.

Input decoration.

Inside this one we will use the border okay.

And in the border you can see that we will need to put some input border.

But it's better to use instead the outlined input border okay.

Because you will see why in.

Once again if I put my comma and I save.

This is how it looks.

So you see it's a little bit better I think in my opinion.

But if you don't want this one you can put the the input border, the underline input border.

This one was the previous one anyway.

So we'll put the outline input border.

And then what you could do is you could put a padding around the column.

So refactor and then wrap with a padding okay.

So now maybe you can put 20 and save.

So this is your text field.

But how can you access the data and display it in the screen?

So for this, what we will do is we will say we will put a stateful widget.

So I will say refactor and I will say stateful widget.

Okay.

So we will use the set state inside.

We don't need the data anywhere else.

So that's fine.

So inside the text field right after we will put a text widget displaying the information.

So how do we display this.

First we will need to create a controller for the text field.

So for this I will go right over the override.

And I will say text text editing controller.

We'll call this one a simple controller.

And we will define it as a text editing controller.

Don't forget to put your semicolon and inside your text field you will put a controller argument this one.

And then you will put your controller inside the thing we just created.

All right.

So now we have access to the controller which can display the text inside like this.

All right, so now when we write the text and we press enter.

And then if we restart the this at reload we see the text inside.

Okay.

But how can we reload it from the screen directly.

So what you can do is you can say you have multiple options, but on one of them is, um, completed

on editing completed.

All right.

So this will return a function.

And inside when it's completed then what you can say you can say set state and you can say that the value uh, or you just can say set state.

Set state should work and do the job.

So let's try it.

I will say restart and I will go inside profile and I will write something and I will press enter.

And here you go.

You have the thing inside.

We have refresh the screen with the text controller the text.

So it does work.

So this is how you create a text field.

So that's about it.

And I'll see you in the next one.

Bye.

2. Checkbox

Okay, so the next thing I want to show you is how to create a checkbox.

So right after the text we will create a widget checkbox.

Inside this one we need to put a value which will be true or false.

So we will create one.

We will say Boolean is checked is equal to false.

Let's put false when we start.

And then what we can do is we can replace this for is checked.

Great for the Onchange you will delete and you will press control space and you will have two options.

So we will select the second one.

And then what you will say is first the value what it is.

It is a boolean question mark.

So let's say boolean question mark.

It's because it can be null.

And then what you will say is you will say we have the value.

Let's set state.

And then inside we'll say is checked is equal to this value right.

But now we have a problem because this value can be null.

So what we'll do is we will replace our boolean for a nullable boolean right.

So now this should work.

If we press on it we have a checkbox.

That's great.

So now let's create another checkbox, but this one will be a list tile.

So let's create a checkbox list tile.

Great.

The value will be the same, so is checked.

I will paste it and the onchange will be the same.

So I will copy the function this thing and paste it right there.

Great.

So it's the same thing.

But the list tile can have a title.

So the title you can put a text widget and right click click me.

All right.

So this was your list tile checkbox.

But also what you can do with the checkbox is you can put it as a tree state.

So we will say tree state and then we'll say true.

This means that it can have three value.

So let me show you.

If we click on it now oops we can see that we have some sort of error.

I will restart the screen.

And let's see I think it is because of the checkbox.

I need to put this one also as a tree state and say true.

All right.

So now if we go inside profile and we click on it we have clicked we have nothing and we have unselected.

So we have three state.

So that's about it.

For the checkbox I will put all my commands like this.

And I'm ready for the next video.

So that's about it.

See you in the next one.

Bye.

3. Arrow Function

Okay, so one thing we need to talk about is the arrow function.

Let me show you.

You see, inside the text field we have a set state.

So what we'll do is we will remove this and instead we will use the arrow function.

So let me show you.

Usually we use this one.

But this one allows us to put multiple function inside.

But the arrow function will tell us okay do only one thing after.

And it is this the set state.

So if you want you could create something like this.

And it will do the same thing.

Because when you complete and you press enter, you see that we have the screen, the text on the screen.

So it does work.

So the set state is the only thing that you want.

But if you want to print something also you cannot use this thing.

Instead you will need to come back to what you had previously.

And then you can put two things inside the print function.

Set state has been clicked or whatever.

You set it, and then you put your semicolon and it will work.

So that's the difference between the the empty function normal or the arrow function.

That's about it.

And I'll see you in the next one.

Bye.

4. Switch

So let's talk about the switch widget.

So it will be pretty much the same thing as the checkbox.

We will do it right after the checkbox list style and we will say switch.

So you see first we need a value.

So we'll go up and we will create a boolean.

But this one will not be nullable just a boolean and we will call it is.

It is equal to false when we start.

So let's use this boolean inside the value.

After this we will change the unchanged and we'll say control space.

And we will select this one.

Okay.

After inside what we need to do is we need to use a set state.

And we need to redefine the value is switch.

Weight is switch to the new value.

And this new value should be a simple boolean.

So it's better to just define the type.

Okay.

So we should be able to switch this thing which is really cool.

The next thing we will do is put a comma and put a switch list style like this.

And the value will be the same.

We will use the is switch and for the onchange we will copy the entire function, copy and paste it inside.

Put your comma format document and save.

Great.

The only thing we need is now a title, so I will copy the one from the checkbox style and I will paste it inside the switch list style.

So now we will write Switch Me instead.

All right so this is what we created.

You will see in the next video I will show you something pretty cool.

I don't want to talk about it right now.

I see you in the next one.

Bye.

5. Adaptive

Okay.

So let's talk about the dot adaptive.

This one is pretty cool because it will target the the the current phone.

And it will display the widget in an iOS style or Android style.

So right now this is Android style, but I am using a phone.

So it could be nice to have the visual of the iPhone.

So we will go over switch style and I will say dot adaptive.

And if we save, you will see that now the icon is automatically adapted to the current phone which is an iOS.

All right.

And you can do this for the switch also.

Dot adaptive.

And you can do this for the checkbox style and the checkbox.

So let's save.

And now you can see that all of these are now iOS looking which is pretty cool.

So it's up to you if you want to use the adaptive inside your app.

Um, I think it's cool.

And I just want to tell you that it's not all the widgets that are able to use the adaptive.

It's just a couple of them.

And, uh, yeah, that's about it.

If you don't see the dot adaptive after your widget, it's probably because it doesn't exist.

So that's about it.

See you in the next one.

Bye.

6. Slider

The next widget we will talk about is called the slider, and the slider will put it right after the switched style.

So don't forget to put your comma and we'll say slider slider like this.

And then the value will be a double.

So I will go on the top and I will go and write double.

Let's call this one.

The slider value is equal to.

Let's put zero that zero when we start.

Great.

So let's take this information and put it inside the value.

Now we need to change this I will press ctrl space and the value will do something okay.

So we will set state.

And inside we will say that the slider value is now equal to the value that we receive from this function.

We can also define the type.

We can also say this is a double just make just to make it like more visual.

And now if we go inside profile, you will see that we can use the slider and it is currently not using the dot adaptive.

So we can use the dot adaptive.

And now you see more iOS style.

But it's interesting to see the value.

So what we'll do is we will print the data.

So which would be the value.

Put the semicolon and open the terminal.

So now you see that when we slide we go from 0 to 1.

But we can also change this.

We can change the max to be 100 or 10.0.

Let's save.

And now you can see that it goes from 0 to 10.

But the thing is we don't have like fixed numbers.

What we can do is we can create a divider or division, and we can say that we will have ten divisions.

And if you save, you will see that now we have only chunks of like numbers like ten, six, five and stuff like that.

So this is how you use and create a slider.

That's about it.

And I'll see you in the next video.

Bye.

7. Single Child Scroll View

The next thing I want to show you is how to create a button out of any widget.

So for example, if we display an image, we will be able to make this image clickable.

But the thing is, before I'm able to show you this, I need to show you something else, which is a

widget that the user can scroll through.

So let's start by adding the image.

So if you remember it was using image dot asset.

And then inside you can put the asset of the image.

It was asset slash image slash background dot jpg in my case.

So you will need to find the path inside your asset image.

And this was your image.

All right so let's put this.

But now if I put like 2 or 3 you will see that we have a problem with the height of the screen.

We are not able to scroll.

So we need to create a widget which allows us to scroll through the page.

And this is called the single child scroll view.

So let's go over the padding.

I will right click and I will say refactor wrap with a widget.

And then you will use the single child scroll view and save.

Now your screen is now scrollable which is really cool.

All right.

So the next step for us will be to first delete one image.

And we need to make this image clickable and do something by example when we click on it maybe to change

this value or anything.

So I will show you how to do this in the next video by.

8. Gesture Detector

Okay, so how do we make anything clickable?

So for this you will need to wrap for example an image asset.

I will right click and say refactor.

And I will say wrap with a widget.

The widget I will use will be the gesture detector.

All right.

You will put your comma also at the end.

Save.

So now you see better.

Now this is clickable.

Let me show you inside the arguments you have something called the untap.

So this will trigger a function which you can use this one.

And what we'll do is we'll say print and we will say image selected or anything.

Put your semicolon save.

And now let's open the terminal control G like this and use the debug console.

And now when I click on it you can see image selected.

So every time I click I have another count.

Great.

So this is how you make anything clickable.

But I want to show you another way to do it.

And the other way will also allow you to make it look different.

So I will show you everything in the next video.

That's about it.

See you in the next one.

Bye.

9. Ink Well

Okay, so there is another way to make anything clickable.

And it's with the Inkwell widget.

And I need to show you both ways because there is a little difference between both of them.

First thing first we need to replace the image asset with a container.

So I will be able to show you what is the difference.

The height of the container.

Let's put 200 and let's put the width to double that infinity which is how the all the width of the screen.

The next thing we'll do is we'll change the color and we will use colors dot white.

Uh, white 12.

Just to create some sort of you can see if I go inside profile some sort of color different from the background.

All right.

So now when we click on it, we should see inside the terminal the image selected this thing.

Perfect.

So this container is clickable but nothing happened on the screen when you press on it.

So let's replace this with the ink well widget and save.

Now you will see that when you press on it you have some sort of splash effect, which is pretty cool.

So it depends of of what you need.

If you don't want any splash effect then use the gesture detector after this.

Inside the inkwell you can change the splash color for colors dot teal if you prefer and let's save.

Now you see that we have some sort of teal, and I think this widget is kind of cool.

So that's why I need to show you both different the inkwell and the gesture detector.

That's about it.

And I'll see you in the next one.

Bye bye.

10. Buttons

Okay, so all of this is pretty cool.

But what we want usually inside an app is a simple classic button.

And this is what we will talk about in this video.

So first for the Inkwell I will change the container for 50 just to make it smaller.

Perfect.

It's still clickable, but what we'll do after this in weld is we will use a elevated button.

Oops.

Elevated button.

Like this.

We will change the onpress for an empty function.

This.

And then the child.

What we will put is we will say a text and inside the text we will say click me, click me like this

and put all your commands save and format document.

There we go, here we go.

So this is what you should have after okay.

So we have a button which is a simple elevated button.

After this what do we want is we want to be able to change the style of any buttons.

So for this what you will need to do is you will say style and then you will say you will repeat the name of the the widget.

So elevated button.

Elevated button.

Dot style.

From.

And now you are able to change the style.

So if you want to change the first color it will either be primary or background color.

So let's try with color.

We have the background color.

So this should work.

Let's put colors dot teal.

Let's save.

So that's cool.

But we can change.

You can change also the color of the text by using the foreground color.

Probably foreground color.

Colors dot white.

So now you can see better the text.

But you don't always need to change the style because flutter offers like four different type of buttons.

All right.

So what we'll do is we will create another one or I will just copy this one and I will paste it right under.

So what you will need to do is I will remove the style so you can see just the button.

This is a normal elevated button.

This is the one we modified.

After this we can create another one under.

But this one will be the field button.

So now you see the difference between the elevated and the field button.

There is another one also, which is called the text button.

And this is the text button.

So maybe you wonder okay, but how do I change the style of one uh, fill button?

You do the same thing.

You use the style and then you say filled button dot style from like this.

And then you can change the style just like you did in this one.

All right, so we will delete this.

And I want to show you one last button which is this one.

We have the alt lined button which will create some sort of outline around it.

But there is also other kind of buttons.

Let's talk about them.

Right now we have the close button like this.

Let's save.

I wonder what this was.

Does nothing.

But you can create a function inside and I think we have the back button also maybe and the back button is just like this.

And you can change the what you do inside the onpressed.

So that's about it.

That's about buttons and uh.

Yeah.

See you in the next one.

Bye.

11. Dropdown Button

Okay, so one last widget I need to show you before we finish.

This module is called the drop down button.

You will see this one is pretty useful when you have a list of selectable items.

I will create this one right over the text field so we can see it at the top, and it will be the drop down button.

Great!

The items for this one will be a list of drop down menu items.

So we will create a list and we will use the drop down drop down menu item.

This one.

The child will be a simple text widget and we will call this the element one.

I will put all the commands like this format document.

We'll put more commands format document and the Onchange will be an empty function this great.

So this is what you should have for the drop down right now.

And you can see that it has one element inside and one element in selected.

So let's change this.

We will copy this drop down menu item.

We will create another one.

And we will call this the element two and let's do a third one.

Element three.

Element three.

Perfect.

So I will restart the application.

I will go in profile.

And now we have some sort of problem.

We will fix this right now.

And inside the drop down menu item you will select value.

And this one will be E1.

And it will create other value for dropdowns.

This one this one this will be E2 and this will be E3.

Great.

Next thing you need to do is we need to create a variable for the value of the drop down button.

This this one will be a string.

It will be from this one.

It will be a string nullable right.

So let's create one string nullable.

String nullable.

And let's call this the menu item and say that we can keep this null when we start.

Great.

So let's copy this and put it inside.

Let's save and restart the application.

So now if we go inside profile you see that we have a drop down which have inside the element one, element two and element three.

And to make them selectable, you will need to go inside the the Onchange.

The value will be a string like this.

String question mark.

So we will put the type.

And inside we will say Setstate.

And we will say that the menu item menu item is equal to the value.

So now when we click and select element two it's now element two.

Element three element one.

So that's pretty cool.

Another thing you can do is you can see that right now we don't have any default value.

If you want to have select element one selected by default, you can do it by putting this as E one because this is the same key as this value right.

So we can restart and you will see that the element one is already selected.

Great.

So this is how you create a drop down menu item.

You have all the tools in order to create pretty much any user input inside your application at this point.

So let's keep going to to build a real application in the next module.

That's it.

See you in the next one.

Bye.

7_navigation-routing

1. Push

The next thing you need to learn with flutter is how to navigate through different pages.

Currently we are only using the bottom navigation bar, but it could be fun to click this button and go through another page, right?

So what we will do is we will take all this garbage and we will put it inside another page, which we'll call settings.

We'll be able to access it from the button that we'll put in the app bar.

All right.

So let's do this.

We will first convert the profile page for a something called settings.

Settings page.

All right.

Next we will rename this.

We will say this is now called settings.

And we can create another profile page.

I will say new file.

Profile page dot dart.

We can copy the home page.

What is inside.

We can paste it inside profile page and just rename this for profile page and rename also this for profile page.

Great.

So now you have three of them renamed.

After this we will go back inside the widget three and we will say that this we need a quick fix and we want to import the profile page.

This thing we can delete the old settings reference.

Great.

So now if we refresh we should be able to see the profile in the home page.

Now we need to access this settings page.

But how do how do you do this.

Let's go inside the widget three.

And inside this one we have the actions.

So right under the icon button you will see another icon button.

This one will have a Onpress.

The Onpress will have this NMT function and the icon will have an icon.

Icons.

Dot settings.

Oops.

So things great.

Now let's put all the commas and save.

So now we have a button settings at the top corner.

We need to use this by going to the navigation or navigator dot push.

And we need to push another page from this button.

So we will push another page over this one.

What we'll do is we need a route for this.

You will always use the material page route.

And inside you will put the builder.

So the builder will be control space.

And it will be a context that returns something.

The something that we return will be the setting page.

All right.

So I know this was a lot of things.

Then you need to put your semicolon, your semicolon right there and your comma and save.

Great.

So there was a lot of things, but at the end it's pretty simple.

We have an icon button.

Then inside we have a non press.

When we press on it we use the navigator.

We use the navigator that push because we want to push another page over this one.

After this we use the material page route in order to display the setting page.

So if we restart the application now, we probably will have an error if we go inside the settings.

And yes, we have.

The reason is let's go inside the setting page.

The reason is we start with a single child scroll view.

But we need to have a scaffold.

We need to have some sort of skeleton for this page.

So we will take this entire thing, this entire single child scroll view.

I will cut this thing and put a scaffold inside the scaffold.

We will have the app bar, the app bar argument, and inside a title with a text saying that this is the settings page.

All right.

The next thing we need to do is to put a body for this scaffold, and the body will be what we have copied or cut, which is the single child scroll view.

All right.

So now let's restart this application and you will see that the home and profile are fine.

And you can go inside the settings to see this setting page with all the scrap inside.

That's pretty cool.

And you can also see that flutter will automatically give you a button to come back.

All right.

Inside the the the app bar.

That is pretty cool.

If you want to remove this, you can always use the uh automatically imply leading and say false.

All right.

If you do this, when you go inside the setting, you will not see the comeback button.

Okay.

So this was how to navigate through flutter.

Let's go back inside again.

It's very simple navigator material page route.

And then you put your page.

That's it.

Inside your page you put the scaffold and you have it.

So that's about it for this one.

See you in the next video.

Bye.

2. Pop

So in the last video we have talked about the navigator dot push.

But now it's time to talk about the navigator dot.

The pop will just remove the last page.

So let's go inside my example the setting page and inside the setting page, as you remember maybe is you have the automatically imply blah blah blah.

So we can set this as false let's say.

So now when we go inside the settings there is nothing.

So let's create this button by ourself.

What we'll do is we will say leading and we will use a back button.

The back button is just a logo of the back like this.

So that's why we use this one.

After this inside we will use a on pressed.

So in the on press we will put a function.

And now inside this one we will use the navigator dot pop just like this.

And inside you put the context.

You put your semicolon and you save.

So now you have created the button to pop back.

All right, just a heads up.

If you put this button inside this page, it will probably not work.

So let's do it just to break flutter, just to see what will happen.

Let's do it inside this app bar.

And now let's save you.

See you just break flutter.

All right.

So just make sure that you have a page previously if you use the pop button.

So I will restart the app.

And now everything still work okay.

So we can go back inside the settings.

And we can remove the automatically imply leading because obviously it will it will not be there because we put one ourself.

All right.

So that's it and I'll see you in the next one.

Bye.

3. Push Replacement

So let's talk about the push replacement instead.

So let's go back inside the.

So I will close all of them first.

And I will open only the widget tree okay.

So this is where we navigate inside the setting page.

Instead of using the push we will use the push replacement.

All right.

And now let's restart the application when we go inside.

This will not just push a page over the stack of pages.

It will replace the previous page.

So now because we still have the back button that will pop, this will create an error because we have removed the previous page which was the widget tree and everything.

So let's restart the app.

And now if you go inside settings, you can remove inside the leading the back button.

So we will cut this.

I will remove this and deleting also.

Now if you save and you go inside, you will see that flutter will not put automatically the back button because you are not able to go back.

There was no page before.

All right.

So at this point you have learned how to use the push replacement.

This can be interesting sometimes when you have like a login screen and the user login, once it's logged in, you need to navigate into the home page of the application.

It could be interesting in this place.

So that's about it.

What I will do is I will put back the push and that's about it I think cancel, let's save and restart the application just to see if everything works.

We still have the this and we can go back.

Perfect.

So this was the push replacement.

That's about it.

See you in the next one.

Bye.

4. Send Data Through Pages

So all the navigation is pretty cool.

But you might have a question like, how do I send data from one page to another?

Like, yes, you could use the notifiers, but sometimes you just want to send like a title to the other page, like a description page.

And that's it.

So I will show you how to do this.

So instead of going inside the setting page and getting the information from this, I will send the information from the previous page.

Right.

So for this you will need to create some sort of arguments to receive.

What you will receive instead is the title.

So we will say required will say that we require this information.

Okay we'll say this dot and it could be title.

So let's put the comma and that's it.

After this we will create the value.

So the final we'll say final.

This will be a string title.

So when we call this page we just give it a title.

And then we can use this title inside the setting page.

So I will use it right there I will say title And one thing you have to understand is if you are using a stateful widget, you will need to say widget that title.

All right.

But if you are using a simple stateless widget, you will not put widget before you just put title and it will work right.

Otherwise it will not.

So just keep this in mind.

Anyway, now we have an error inside the widget tree and it's because we need to put the argument title.

And this will be a string which you will put a simple settings and save.

So now or even we can put settings or something else just so you see the difference.

We will go inside the settings.

And now you see that the title is settings.

This thing.

Great.

So the information is coming from the widget three.

And then we send information inside the settings page.

We handle this information as a title and we use it.

Okay.

Again you use widget dot title for a stateful and you just title if you are inside a stateless.

Okay.

So that's about it.

See you in the next one.

Bye.

8_clean-ui-widgets

1. Clip R Rect

For this section.

What we will talk about is clean UI widget.

So you want to create an application that looks good at the end also, right.

So this is what we'll do.

We will merge them with the layout widget.

And we will create something that looks good at the end.

Let's start by creating a new file called the welcome page dot okay.

This page first we will import the material and then this page.

The goal will be to have a button click login or register and go to the the home page the widget three.

All right.

We will do this by creating a stateless widget first which we will call Welcome page.

And inside we will put a scaffold widget.

All right let's save this.

And now let's go inside the Main.dart.

And what we will do inside this we will remove the widget three and put the welcome page.

So now when we start the application let's restart everything.

When we start the application and we restart we You will see only a welcome page.

The goal is to be able to navigate through this to go inside the widget.

Three.

So let's go inside the welcome page first.

And what we'll do inside is we will create a body.

So the body will have a column.

And inside the column we will have children.

Great.

Inside this one we will use the widget I want to show you in this video which is called the clip or

rect.

This one.

This widget will allow you to create rounded borders for any widget.

So in this case we will use the child and inside we will put the image we had previously.

So the image asset and inside we will say it was a set slash image slash background dot jpg.

So you need to put the path of your image.

This one is set image background jpg.

Okay.

And now let's save.

It's right there.

We will put the column as the other main access alignment.

Main access alignment dot center.

So it goes in the center.

And now I want to show you the Clipper widget.

So you will say border radius and you will say border radius.

That's circular.

Then you will say 20 dot zero and save.

Now you see how the borders goes rounded.

And this looks pretty good.

So that's good.

And after this we will wrap the column, refactor wrap with a widget and it will be a padding because we want to create like maybe 20 padding and it'll look a little bit better.

This is where we will put like the login information.

We'll just like write login and we'll have the button login and register stuff like that.

All right.

So this was how to use the Clipper widget.

Before we finish this video I just want to add another button right under.

So we will use a filled button.

This will use the an empty function.

And the child will be a text saying um I don't know.

Login like this I will put all the commands.

Save.

And now the only thing we need to do is to add a navigator.

So I will go first inside the main.dart in order to remove the unused path.

Okay.

And next I will go inside the widget three to copy the navigator that push.

So let's copy all of this Ctrl c.

And now we can close all the tabs and just use.

So I will say save.

And now we will just use the welcome page okay.

Inside this one we will paste the navigator that push.

And we will change this for push replacement.

Right.

And inside we will return the widget three.

Widget three.

Like this.

So now let's restart the application entirely and click login.

You see that you go inside the widget tree where you have your entire application.

There is no way for you to log out right now, but we will do this later.

So let's refresh and this is what we will keep working on to make it look clean as a welcome page.

That's about it.

See you in the next one.

Bye.

2. Hero

Okay, so the next widget I want to show you is called the Hero Widget.

And for this we will need to create another file inside the widget.

And we'll call this the hero widget.

Okay.

Like this dot dart.

After this you need to import em.

So import material.

And then we will create a stateless widget which will be called hero widget.

So this is what you will need.

And next what we can do is we will replace the container with this entire image with the rounded corners.

So we can go back inside the welcome page.

We can cut this clip rect like this.

We cut this and we call instead the hero widget.

Okay.

You can go back inside the hero widget and we will return the Clipper rect.

So let's save.

Nothing should change right now, but we can wrap this clipper wrecked with the hero widget.

So let's say refactor and wrap with a widget.

We will use the hero.

One thing you need to put inside the hero is a tag.

You need to tag this with a unique value.

So we will just say for this example the hero one.

All right.

But you need to use always a different value for each hero widget.

The next thing you can do is we will plug this hero widget first inside this welcome page, but also inside the home page.

So let's replace the center with a column or first a padding.

The padding will be agent sets dot all and it will be 20 dot zero.

And then we put a child.

The child will be a column.

And then in the column you will put children.

So the widget and the widget will be the hero widget that we just created.

All right.

So this was a lot of things.

But this overall is what you should have.

So you can pause the video if you want.

What does the hero widget do.

You can see that the hero widget is inside this page but also inside the home page.

Have you seen the transition?

So the the hero jet will always create some sort of transition for the image or for the widget that is inside, which look pretty cool.

So what we could do is we could create inside the log the the profile page.

Let's go inside and replace this for a padding again with the agent sets dot all 20 dot zero.

And then put a child column.

And then children that will have a widget inside.

And the widget will be a list view a list style, sorry list style and the list style will have a title.

The text will be log out.

So now we will have at least a log out button.

And then what you can put is the untap.

Put an empty function and this will log out.

In order to log out we can go inside the uh welcome page.

We can copy this navigator, this thing.

I will copy this and I will put it inside the profile page, inside the list style.

Like this.

Now the only thing you need to do is replace the widget three with the welcome page.

And now if you restart the application, you will see that when you click login, it creates some sort of transition between the image and when you click logout then you go back.

That's it.

So this was oops.

So yeah it's not really visible like that.

So maybe it would be better another button.

But you get the point right.

When we do the logout we should do something else.

We should also change the value of the notifier.

So this notifier selected page index we should put it back to zero.

So we'll say selected notifier dot value oops.

Notifier dot value is equal to zero.

So we restart this.

So let's restart the application.

When we click login we click profile.

We click logout.

And then when we click login you will see the little animation.

So that's about it.

This is the hero widget.

Pretty cool stuff.

See you in the next one.

Bye.

3. Snack Bar

Another widget I want to show you is the snack bar widget.

This one is pretty cool, so we need to talk about it.

First thing first, what we'll do is we'll go inside the welcome page or inside the hero widget.

And I want to change it because it doesn't look like with the theming of our application.

So let's change the color of the image for colors the teal.

And after this we will need to change also the blend mode.

So the color blend mode, blend mode dot.

And you can use maybe blend mode dot modulate.

This one is pretty cool.

Or we could use darken.

Darken I think will make more sense for our application.

So you just see how it looks better like this.

And after this another thing we can do is we could change maybe the the title inside or whatever, but we will do this later.

Okay.

So another thing I want to do is to use this widget which is called the snack bar.

So for the snack bar we will go inside maybe the settings page, and we will use one button to trigger the snack bar.

So if we go inside settings you see that we have multiple buttons.

We will we will use this one.

The first one click me okay.

So I will replace this for open snack bar and we will save.

Oh you see how it is.

It's this button instead.

So I will instead use it inside the elevated button I will cut this elevated button.

I will put it at the top, at the top before the drop down button.

And now we can play with it.

I will remove the style and I will say this is called open Snack bar.

Okay so this is our button.

Now let's let's make it work.

I will say show snack bar.

As simple as that.

So is it show snack bar I hope.

Uh, no.

Show scaffold dot.

Yeah, it's scaffold messenger.

So scaffold messenger dot of context.

Dot show snack bar.

This is it.

All right.

So you need to remember this little thing.

It's scaffold messenger dot of context.

So you get the context of the current scaffold.

And then you will display something on the screen.

And the display is the snack bar.

So you need to create a widget snack bar like this.

And the content will be, I think, just a simple widget.

So you can say text and say snack bar.

Okay, after this you need to put your semicolon and your commas like this and you can save.

Great.

So now when you press snack bar, you see we have a snack bar coming from the bottom that will stay there for like two seconds.

You can change the duration if you want.

You can say duration and you say duration and maybe seconds five seconds.

All right.

So when you click on it you have the snack bar.

But another thing that is pretty cool is the position of the snack bar.

I don't think its behavior, but I'm not sure.

So we'll try this one behavior.

I will say snack bar behavior.

And I will say that floating it is exactly that.

So let's click on it.

Now you see how it is a floating box and it just looks better.

So this was how to create a snack bar with flutter.

That's about it.

And I'll see you in the next one.

Bye.

4. Alert

Okay, so the snack bar is pretty cool, but another thing that you could be using with flutter would be an alert dialog.

So I will show you how to do this also.

We'll copy paste the elevated button and we will call this instead the open dialog.

All right.

So it's pretty much the same concept.

But instead of using the scaffold messenger of context first you can right off the bat use the show dialog.

All right.

And inside you will put a builder.

So you will press control space and you will say context return.

And you can just put a alert dialog or about dialog.

They are pretty much all the same.

So we will just use an about dialog for now.

And it is like that.

So let's click on it.

You see how we open this.

So you see that the about dialog will automatically give you all the licenses of your app.

So this can be interesting if you need this.

Otherwise we will use the alert dialog.

The alert dialog will allow you to customize it more like you want.

So if I click on open dialog, you see that nothing appears right now.

This is because we need to put some stuff.

So first we need to put the content which for now will just be a text widget saying alert content.

And then we need to put also maybe a title which will be another text saying alert title.

And finally, if we try it, you see that we have the alert title and the alert content.

But we want to see to be able to close this thing.

So you can create a button and this will be inside the actions or is it inside actions?

Let's find out.

It is inside the UL.

It is action or I think it is inside the content action.

Yeah, exactly.

It's inside the alertdialog.

Sorry, I was at the wrong place.

So inside the action, you will say, for example, a close button or fill button and you will say oops,

filled button.

And inside you will put an empty function.

And after this the child will be a simple text saying close, right?

But you can put any other buttons if you prefer.

So this is just that.

And inside, in order to close the navigator, in order to close the dialog, you need to use the navigator,

because this is kind of another page over it, right?

So you use the navigator dot pop.

As simple as that.

And now if you click on open dialog, you have the alert dialog with the content and you can close it.

So this is pretty cool.

Um, that's about it.

And I'll see you in the next one.

Bye bye.

5. Divider

Another widget we need to talk about is called the divider widget.

So between these two elevated buttons I will create one called the divider.

All right like this.

And let's say so you see this is a little line that separates the two widget.

You can also change the color colors that teal.

And also you can change the thickness the thickness.

Let's put 5.0.

You can also change the the end indent.

So this is the space at the end that you will have without the divider.

So for example I put 200 dot zero.

Let's save.

You see that this is 200 space that is not used for the divider right.

So this is how to use the divider.

You can also use a vertical divider.

So I will create a container.

I will define a height of 50.0.

And then I will use a child and put a vertical vertical divider inside.

And then I will save.

So you can see that we have another divider.

But this one is vertical.

It can be used for example inside a row if you want to separate two items.

So that's about it.

What we'll do is we will delete the vertical divider.

And you can keep just this one with maybe a value of two for the thickness.

So it's not that disturbing inside this page.

All right.

So that's it.

And in the next video we will talk about the card widget.

6. Card

So now let's talk about one of my favorite widget called the card widget.

It's like a container, but you already have some sort of rounded corners.

And I will show you how it looks like.

So first thing first we will go inside the home page.

So right there we can close everything else.

So I will say close all and I will return inside the home page.

And what we'll do is first thing first we will create a padding widget.

And inside we will put an agent sets dot symmetric.

And the symmetry will be only vertical.

So it will be only vertical.

The padding I will put all the commands and save.

Okay.

Inside this padding we can put a child argument and use a card widget.

In this card widget we will use a single child again and we will put another padding and the padding will be again.

Agent sets dot all for this case.

And we will put maybe 20 dot zero.

I will put again all the commas and save.

Okay so you see that this is our little card.

But there is nothing inside.

So inside the padding I will say child and I will write a text.

The text will be by example.

Um.

Basic layout.

This is the first module module of this course.

All right.

So we have this.

The next thing we want to do is we want to add more things inside.

So we will need to put a column instead.

So for the child, I will right click and I will say refactor and wrap with a container.

A column where it is column.

Right.

And we need to do the same thing but right under.

And this will be by example a the the description of this.

All right.

The last thing we'll need to do is to put some sort of width double that infinity.

So can we do it inside the card.

No we can't.

So what we'll need to do is we will wrap the card.

We'll say refactor.

Or maybe we can even use the padding.

Replace this for a container because the container also have the padding, but the container have the width argument.

So we can say double dot infinity.

And now let's save.

So this is our card widget.

You can see that it starts to look kind of beautiful.

So that is why I use a card widget.

Next we will need to put everything on the left side.

So we will go inside the column maybe inside this one.

And we will say the main axis or cross axis alignment cross element dot start.

And now we have our text to the side.

But there is another thing we need to change.

And it's the style of the text.

And we will do this in the next video.

So that's about it.

See you in the next one.

Bye.

7. Text

So the next widget I want to show you is the text widget.

Let's play with this one and change how it looks.

So first thing first we'll go inside the text and we will use the style argument.

This one will take a text style and inside the text file you can put a color.

Colors dot teal.

So this will change the color of the text like this.

Next you can put the font size and you can change the size.

Let's put 18 and save.

We can also change the font weight.

So font weight bold like this.

But also you can put maybe normal if you prefer or W129.

So this is very thin and this is very bold.

All right.

But we will use the bold one instead.

And that's it.

You can also change many other arguments like the spacing later.

And it could be 10.0.

But it's up to you.

Okay, so let's keep it simple.

Let's save.

And now what we need to do is we need to put this inside a data.

What we will do another file and we'll say constants dot dart.

Because it could be interesting to have always the style of this saved at one place.

So we can reuse it everywhere inside the application.

All right.

So I will cut this textile.

And then what we will do is we will go inside the file content.

So I'll cut this.

We go inside constant and we create a a class.

Let's create a class called K textile.

And this class K textile will open the bracket inside.

What we'll do is we'll say static const.

And then we will put text style.

How do we call this one.

We will call this the normal blue or normal teal text Or even tidal.

Teal.

Text is equal to the text field and we need to create another one.

But this one will just be maybe.

Um.

Description.

Text.

Description.

Text.

For the description, we will remove the color.

We will remove the bold, and we will keep only 16 as the font size.

Okay.

So we have two type the type the title teal text and the description text.

And inside this we will change this for the K textile dot title text title teal text and the other one

we will do the description text.

Okay, so this is what we have done.

We have used the text widget inside.

We have used the text style widget in order to change it.

And then we have saved everything inside a constant class.

So we can reuse this anywhere inside our application.

So our application look the same everywhere.

So that's about it.

And I'll see you in the next one.

Bye.

9_package-management-and-modifications

1. Lotties

All right, first thing first, congratulations on being on this step of this course because a lot of

people will give up at the start.

So the next thing I want to tell you is we have talked about the basic flutter widget, some cool widget

like the cart and stuff like this.

But now let's go one step further and let's add animations inside our app, because at the end we don't

want a garbage application like this.

We want to have a great visual application.

So we will change all of this and make it look great by using Lottie Files or Lottie animations, you

can go on Lottiefiles comm and you can search for an animation you would like to have inside your welcome

page.

Okay, you can write home by example, just make sure to click there and select free.

And in this way you just have the free one.

Then you can click on for example this one.

It will fit with the application, have the same color teal and blue.

And you can just say download for me.

I will not be able because I have already downloaded ten files, but you will be able with the basic

basic free plan or whatever it is it's called.

So you will be able to download this.

Otherwise you can go on another website, Lottie files and stuff like this.

You get the point.

The next thing is I have already downloaded the JSON.

It's right there.

I can't download anymore but anyway.

And we will put this JSON inside the application.

But how do you do this right?

How do you put a JSON file inside flutter?

For this you will need to use the Lottie flutter package like this.

So you go on Pub.dev package Lottie and you will use this one.

So what I will do is I will go inside the installing and the only thing you need to do is either run this or either add the Lottie automatically inside.

So what I like to do the fastest way I think, is you just take the name of the package, Lottie, and you go inside the Visual Studio Code.

You press control shift P and then you say add dependency and you press enter.

Okay, in this way you can just put Lottie and you will see that inside my pubspec.yaml it will add the lorry right under this one.

Okay so let's do this Ctrl p add dependency lorry enter and you will see.

Boom we have the lorry.

If you want you can add it manually right there.

Or you can paste inside the terminal.

What I've shown you this little thing you can paste this.

All right.

So there are multiple ways but this is the way I do it.

All right.

So now we should have access to the lorry files.

Let me just click this flutter pub get this little button if you don't have it just right inside the terminal flutter pub get and press enter.

Okay.

It will do the same thing.

We will save the file.

I will close the pubspec.yaml.

And now we are ready to use our Lottie file inside our thing.

So what you will do is you will create a file like this.

I will delete mine.

It was created too early.

But you will go inside the asset.

You will right click and you will say new folder.

Inside this one you will say lorries.

And then you will put your lorry inside this one.

So it is right there.

I will put my welcome come inside the lorries.

Okay, so now we have access to this lorry, but we also need to use the package that we have added in the dependency.

This one.

So you will go inside by simple example and you will find or just inside readme.

I think they will give you an example.

Yeah.

You just need to use this lorry asset and then you put your JSON.

Pretty simple, simple stuff.

So I will copy this line.

I will close everything okay.

Because we don't need this anymore.

And now we need to use this.

So let's go inside the view inside the page inside the welcome page.

And I will replace the hero widget with this thing.

The Lodi dot asset.

I will say quick fix and I will say import the package Lodi Lodhi.

All right.

So now we have the import.

We can delete this unused import which which was the hero widget.

And then, uh, we will have an error.

The reason is you need to go inside the pubspec dot and inside the pubspec.yaml.

You will see that under assets, We need to give the asset path for the ladies.

So I will paste under and I will say ladies.

And don't forget to put this.

Otherwise it will not work.

So this will give you access to all the files inside the ladies.

Okay.

So we will save this flutter pub get and then I will close the pub spec.

All right.

So now we can just change the name for welcome and save.

And now let's restart the application and see what happened.

So you see that unable to load assets.

Welcome.

Yeah.

It's because I need to put loads slash welcome and save.

Great.

So this was the animation I downloaded earlier.

It's not the home but I think it's pretty cool.

You can find it on Lottie Files.

I'm pretty sure this is where I found it, but it's up to you.

Anyway, so this will be kind of our welcome page.

Okay, so what we need to do next is maybe put a text under saying you can put the name of your application.

I will just put flutter map.

Another thing I would like to do is change the style.

So we'll go inside the style and we'll say text style.

I will not use the textile inside the constant this thing, because this style will be a little bit,

a little bit weird.

It will not be like standard title or description.

So that's why I will not use this.

So I will say text style.

And inside the text style we'll say um font weight.

Font weight.

Bold.

Font size the font size.

Maybe we can put 50 dot zero.

And then the space letter spacing.

Letter space letter spacing 50 dot zero.

Again.

Let's see what happened.

Okay.

So right now we have a problem with the visual of the application.

I want to show you another widget that will help you fix things like this.

So we will just stop this video.

And in the next one I will show you the fitted box widget.

You will see this one is pretty cool.

So that's about it.

See you in the next one.

Bye.

2. Fitted Box

Okay.

So let's talk about the fitted box widget.

It's one of the most powerful widget that you can use.

And not a lot of people is talking about it.

So the fitted box widget, what it will do is it will take it will take the text and make it fit to the screen.

Right.

Simple as that.

You can just wrap the text widget, right click refactor and then you will say wrap with a widget and you use the fitted box.

And now when you will save, you will see that the code automatically put all the text in the line and align with the phone.

Right?

You get what I mean.

Anyway, so another thing I want to show you is you can change maybe the font size for 500 and you will see that it will automatically adapt also.

But when you do this it will remove a little bit of spacing.

So you need to adjust for something that you like.

I think 5050 seems pretty cool.

That's fine for me.

The next thing we'll do is we will add a simple sized box right under with a height of maybe 20.0 and comma save.

Great.

So now it looks better.

We will put another button.

This one will be a simple text button.

So I think I will just copy this.

Yep.

Copy and paste it right under replace field button with the text button.

And that's pretty cool.

Uh, both of them goes to the widget three.

But we will change this in further video for a login page or a register page.

And we can just change this for, uh, login and get started.

Get started.

Okay.

So this will be the register and this will be the login.

Another thing we can do to make it look better is we can expand this text button to be all the width.

And for this you will need to go inside the filled button or any buttons.

And you will need to change the style.

So we'll say style.

And if you remember to change the style of a button you reuse the name.

So fitted filled button this filled I will just remove this because it's kind of annoying.

Perfect fill button.

And you just need to fill it button.

Have some difficulty with my keyboard right now, so I will just copy paste fill button dot style from.

And then you just need to put size I think.

Or minimum size.

Yeah.

Minimum size.

And you use the if you go over size widget like this.

And the size widget just take a width and a height.

So the width will be double dot infinity.

And can we put null for the height.

No we can't.

So what we'll do is we'll put maybe 40 dot zero and comma save.

And now this should work.

So we'll restart the application.

Uh it's a maximum size.

It's a minimum size I was I want to use instead.

So minimum size.

And now let's save.

Great.

We will do the same thing for the login.

And that's about it for this video.

So I will copy the style and I will paste it right over the child of the text button.

Let's now save.

And you can see that the text button is a little bit invisible.

But when you press on it, you see it.

You have the get started, you have the name.

And this look pretty cool.

So what we can do in the next video is instead of going instead, instead of going straight in the app, we will go to a register page or a login page.

So that's about it.

See you in the next one.

Bye.

3. UI modification

Later.

Inside this module, I will show you how to use a package that will save data locally.

For example, when we restart the application, it's always going back to dark mode.

So it would be nice to save this information locally on the phone.

We will do this inside this module, but for now I don't want to show you like package after package after package because it's a little bit overwhelming right.

So what we will do is I will show you a package like the Lottie package, and then we will modify a little bit the application to make it look good.

And once in a while I will show you a new package, for example, the one that will save the data locally.

This one will be pretty cool, but that's that's it.

So let's start right now and create another page so you can see that when we log out and when we log

in, we go directly inside the application.

But it would be better to go inside a login page or register page okay.

So inside page I will say new file.

And I will say um register page or login page.

Let's put it at as login page okay.

And then I will say import material and then we'll say Statelesswidget.

This one will be called the login page.

So inside this video what we will do is we create a login page.

So when we click login instead of going inside this one we go in the login page.

And then we put some information.

And then we go inside the home right.

So this is what we tried to create right now.

I will put a scaffold first with a nav bar.

The app bar will just have a a bar inside.

Then I will put a body in order to be able to put a column widget.

Right.

So what I do is I know that I will put a padding over the column, but I prefer to right column first

because when I do refactor with the padding, it will automatically put the the information inside.

But if you just put the widget padding, let me show you.

If you just put padding, you will see that it doesn't fill everything automatically.

So that's why I prefer to put column first and refactor so it go faster after this.

So you have this thing.

We will put this as maybe ten.

And now let's go inside this page from the welcome page, when we go, instead of going inside the widget

three okay.

Which is the application, what we will do instead is we will replace this for the login page like this.

So for now both of them will go at the same place.

You can remove the unused uh link okay.

And then let's restart.

So we have this.

When we click on Get Started we go inside this login page.

So that's pretty good.

Next what we'll do is we'll go inside this login page and modify the visual.

So you can write children to put multiple widget inside.

And then which one will we put.

I think we will put an just an image.

We can put the hero widget that we had previously hero widget.

Like this.

But can we put something inside this one?

I will restart just to see what happened.

Get started.

Perfect.

We have this information.

And let me restart again just to see if we have an error.

No it's good.

So we could put the word login inside.

So let's go inside hero widget.

We will modify this.

We will say this will take a required um this dot title.

And we'll say final string title.

Oops.

String title.

So this just means that we will send the information via an argument.

And we will have this information available for our widget, which we will wrap the entire hero widget.

I will say refactor wrap with a column.

Why?

Because it will replace column with stack.

And now we are able to put some text over it so we can say, um, some text.

Let's put a simple text, let's put inside the title.

So title and let's save.

Now we just need to put the title inside the center.

So what we will do is we will say Alignment and what it is.

It is a wrap alignment.

Alignment.

Simple alignment.

Dot and center I think.

So let's try this.

We'll go inside the login page and we will add inside the hero widget.

We will add a title.

The title will be login login.

Perfect.

And we'll go inside the home page and do the same thing.

Title.

The title will be I don't know flutter map.

It doesn't really matter for now, but now you can see that we have the login next.

What we can do is we can make this look better.

So let's go inside title.

Let's go inside this.

Change the style again.

We will not use the text style from the the constant because it will be a weird one.

Right.

So let's say text style, the font weight.

Maybe we can put font weight bold and then the font size will be maybe 50 to 0.

Let's save.

So now we have this.

We can put the letter spacing.

Letter spacing maybe 50 to 0 or so and wrap all of this text.

Refactor.

Wrap inside a fitted box for box and let's save.

Perfect.

So that's kind of good.

It's a little bit weird.

So what we'll do is we'll change the color.

We'll say color colors dot.

Uh maybe let's try black okay.

That's not right.

Um let's try white colors dot.

White.

White 30.

Let's try this.

All right.

So we will go with something like this.

It doesn't really matter because it's like a little different.

So maybe 60.

Yeah.

That's good.

That's enough.

Good.

Uh, after this what we'll do is we will put the text field two text field.

Do we have some of them inside the settings.

We can just copy paste one of them.

So we have this this drop down text field.

So let's copy this text field.

All right.

So you had it inside your setting the page setting page dart.

And I will close all the page.

Close all.

Then we will go back inside the hero widget.

And what we want is the text field.

So instead of hero it will be a login page.

Now inside the login page we need the text field.

So I paste it.

And then we need a controller.

So this will need to put a controller text editing controller.

Controller is equal to a text editing controller.

And that's it.

Okay now we will need to change this for a stateful widget refactor because I need to show you something

else.

So convert to a stateful widget.

And now we need to use something called the dispose.

But what is the dispose.

I will explain you what it is inside the next video.

So that's about it.

See you in the next one.

Bye.

4. Overrides

Okay, so what is the dispose?

The dispose is part of the override that you can use inside a state, a stateful widget.

Don't make the confusion.

You are not able to use them inside the stateless widget.

They are not available only in the stateful.

So you have the dispose and you have also the init state.

Oops init state.

You have a couple other of them, but the most used are these two ones, so the dispose will dispose the thing when you don't need the login page anymore.

For example, when you change the page and you go in the home, you don't need the controller anymore.

So you can just say controller dot dispose.

All right.

So you don't need to save the information of the text forever inside the app.

You can just dispose it.

Next in the in the init state.

This will run when you start the the page.

For example, when you go inside the login page this will trigger something.

You can trigger some functions or stuff like this and then it will build the entire thing.

Right.

So what you could do is you could just say print and init state init state, put the semicolon.

And now if I restart the application and I open the terminal or the debug console, maybe let's say get started and you see init state.

So this trigger right away when you go inside the login page it will be triggered once it will run this information and then it will go inside the build and rebuild the code only if you say set state.

But when you say set state, it will not rebuild the init state, only the build.

Okay, so this was a lot of information.

If you want you can rewatch this section of the video okay.

Just to make sure you understand.

And that's about it about the overrides.

And just make sure that you have put the controller that this pause for your controller.

We can also put the controller, I will cut it and we can now put it inside this okay.

So it will be close to the other.

We can delete the init state because we don't need it only the dispose.

So that's about it.

And I'll see you in the next one.

Bye.

5. UI modification

Okay.

So let's keep going on this UI design.

So first thing first what we will do is we will change a little bit the padding for 20.

Like this I will save and we will add a sized box right there over the text field of height 20.0.

And now let's save this.

Okay.

So we will change also the outline input border.

And we'll use a border radius.

Border radius dot circular 20 maybe 15.

That's zero.

Like this.

And the comedian save.

Perfect.

So this looks better.

Another thing we can do is we don't know what is this text field.

So it would be nice to have a hint.

And this is what we'll do inside the input decoration maybe.

So let's try int in text.

Yes it is perfect.

So in text and we just say uh email email and save.

Great.

Another thing we can do is the controller.

We can copy this one, paste it on there and say email and say pass word and say same thing.

Controller.

Controller.

Email and password.

Because you guessed it, we will do two text fields and we need to separate the controller.

Otherwise it will be the same.

So we need to say controller email.

And do we need to.

Do we need anything else.

This seems good.

We can copy this one and copy the size box also.

Oops.

Size box and text field.

Copy and paste it under.

We will change the size box for ten instead.

And the text field we will put pass word.

And now let's save.

Okay so this seems pretty good.

Now I will just restart to see what happened.

Get started.

And now you have the email the password.

You can enter the information.

Maybe you will need some sort of button to say login.

So let's add this button.

We can put a field button.

We can also copy the one from the welcome page.

It would be faster, so just take the field button and copy this with the size box.

Why not copy this and go back inside the login page?

Inside the login page you just paste all of this and you have your size box and you have your field button.

Great.

So now you can just say login like this.

And this seem good.

This seems good.

The next thing you can do is for example, the email or one thing I want to do also is inside the hero widget.

I just want to change the visual look a little bit weird.

So I will say, um, black, black, 4554 instead.

Uh, it looks weird also.

So white white ten maybe I will also change the bold for normal.

You can play with it, but for now I think we will keep something like this.

And maybe 38.

Let's save this look already better, I think.

Okay, so that seems good.

I will change again for 30 instead.

And this seems even better.

Okay, the next thing is to write the email.

So you see both controller use the same thing.

So I will go inside the login page and inside the login we will say um controller email this one.

We change this for controller password.

Great.

So that is better.

The next thing we need is when we press on login we need to go inside the widget three right.

So for this you would change the login page for widget three.

Here we go.

And we can create even a logic to enter a real user or real email and a real password.

And only if you put those two things you can go inside the login page.

All right.

So for now it's not working.

But we will fix all of this inside the next video with a logic coding challenge.

That's it.

See you in the next one.

Bye.

6. UI modification

Okay, so let's create the logic to enter some information and click login.

First thing first we will change the fill button with the elevated button.

So I've changed both of them the style and the button.

And let's save.

I think this looks already better and we will do the email.

So how can we write this email?

We can say for example let's go right after the class login page.

Extend this right after the controllers you can write string.

I will click this and you can say um confirmed email is equal to one two, three.

And then the string confirmed password goes to I don't know 456 doesn't really matter.

Okay.

These will be just the information to confirm.

If you have put the good information to go in the login page.

Right.

So let's go inside the this.

So you can see that we have some sort of function that is triggered when you click the elevator button.

So we will take all this function and we'll put it inside another one.

So it would be like more visual of what happened when you click the elevator button.

So let's cut all of this.

And what I will do is I will say um, let's create a function.

So it's on login pressed like this okay.

And now let's create this function.

Usually what I'd like to do is right before the last one.

So right there you just click and you say um, how do we call this.

We call this a void on login.

Proceed.

We open the bracket.

We create a function.

So this is just a simple function that is used inside the elevator button okay.

And all of this is not inside the build.

So you can see the build end right here and right here.

So that is why we use it right here.

After this we will put the navigator and we'll create some sort of function.

We will say if the context or the confirm email is equal to the controller email dot text or is equal equal just to say, if these are equal, then we will do the navigator like this.

Oops.

Then we will do the navigator.

Can I just move it.

All right.

So that's good.

We can do another thing the same thing but with the password.

So we can say um, and if the confirmed password is equal to the confirmed uh, controller password dot txt.

Can we just put text?

Why is it working.

All right.

So I don't know why text doesn't work.

Get a text.

Define the type string.

Confirm password because I.

Controller.

Password.

Controller password.

Perfect.

All right.

So this was pretty long.

But right now if you click login nothing will happen.

So let's restart the application entirely.

I say get started and then I say login.

Nothing happened.

But if we press one, two, three and then four, five, six and say login.

Here you go.

You go inside the home page.

Pretty -- cool.

But later inside this course, or at the end of the course, you will understand that you can create your own, um, authentication system.

So you will either use your own database that you have or you can use Firebase and stuff like this.

But this is not like the good way to do it because it's just like static inside the app.

You will need to connect with your authentication system.

But don't worry, I will show you everything that you need in order to connect with APIs and stuff like this.

All right, so that's about it for this one.

Uh, let's log out and see if we can change one little thing.

This seems pretty good.

I will say maybe a button to come back.

That could be interesting.

So for the app bar, what we can say is, um, do we need a button to come back?

Actually.

So yeah, we will go inside the welcome page.

And what we'll do is you see we have a push replacement, but instead what we'll do is we will just do a push.

So I will take all of them, remove both of them and save.

So now when we go inside we have the back button automatically.

So the get started and you have the back button if you need right.

If you go and log in you have it also.

Great.

Another thing we can do is so we see that the login is only accessible if you put one, two, three, four, five, six and then login.

And that's pretty cool.

So as I told you, inside this section we will talk about how to save the data locally.

I will show you how to connect with uh, an API, stuff like this.

You will see this is a lot of information, but for now that's about it.

And I'll see you in the next video.

Bye.

7. Verify and fix your App

Okay, so when you build an application, you need to take in consideration all the little things that can break your application.

For example, if I turn the screen you can see that this is not scrollable.

So if someone has a very small screen this might create a problem.

So you need to wrap the padding of the welcome page, this entire body refactor and wrap it with a single child scroll view single oops single child scroll view.

And now let's save.

Perfect.

But another thing is this logo is pretty big, so we will put it maybe at 300 height.

Let's go inside the Lottie asset.

In the arguments I will say height and maybe 400 dot zero and let's save.

So this I think is already better.

It's not really good for the this orientation, but usually people take their phone like this.

Okay, so the goal is really to fix the big problems.

So by example, if I go inside, get started and I try to write something and I open the keyboard.

So control command shift K on Mac and I start to write something, you see that we have a yellow bar.

So we need to fix this also okay.

But first let's fix this one.

This one.

Now you see that it's not in the center anymore.

You could try to do the main access alignment main access sys main access sys that minimum.

I'm pretty sure this will not work.

Yeah.

So the reason is right now we are we are telling the column is the minimum size and it's in the center but it doesn't go in the center.

The reason is the single child scroll view will start everything from the top because this will be scrollable.

Right.

So you cannot do it like this.

What you can do instead I will remove the minimum.

What you can do instead is you can wrap the entire single child scroll view, Scrollview.

Refactor wrap with a center widget.

So now everything will be in the center and if it's too big, for example, if I add another asset,

you will see that it's still scrollable.

All right so that's good I will remove the two assets.

And that looks pretty good.

All right.

And it's also scrollable now.

So let's go inside to the next page.

Get started.

Inside the login page we need to do the same thing.

We need to make this scrollable first.

So if I restart this seems to be pretty high up.

Get started I think this was in the center also.

Main axis alignment.

Main axis in the center.

Okay, but the thing is, when you open the keyboard, you see that when you open the keyboard, can

I open it?

Keyboard okay.

When you open the keyboard you have this error.

So you need to make this over to the body.

Refactor with a single child scroll view.

Single child scroll view and refactor again.

So refactor with a what was it?

The center?

Yeah.

So let's save.

Now everything is in the center.

But when you start to write you don't have this error because it's scrollable.

All right.

So that's pretty good.

Another thing we can do is maybe we can remove the the size box.

Do we have a size box over.

We can put a size box at the bottom so it will look a little better.

So let's put sized box with the height of maybe 50.0 comma at the end and save.

So this looks a little bit better already.

So that's good.

So we have the get started.

We have the login.

And maybe we could have the register or something like this.

Yeah.

Let's create this right now.

So what we can do is in the login page we can receive the information the information that we put inside this this error widget.

So the login name.

So we can say this will be a required.

And it will be the how do we call this the title.

Yeah.

Let's just call it title.

And now we just need to create a final string title which we can use inside our login page.

But we have a problem we need to say required this dot title.

Great.

So now we have access to.

Title when we send it from the welcome page.

And we will just use it inside this if you remember.

When you are inside a state full widget you need to say widget the title.

Like this.

Okay, so that's good.

Another thing we can do is go inside the welcome page and inside this one we'll send the information.

So the title will be for when we click on this button which is the Get started.

This will be.

Register.

And when we click on the other one this will be login and save.

So let's restart the application.

Now when you click Get Started you have register and you put your email password and login.

We could change also this button name.

So let's go inside the login page and we can change this one for widget dot title.

And let's save.

Perfect.

So now we can register or we can login.

But as you can see both of them use the same confirmation email which is just static.

It's just the information stuck inside the the application.

So you will need to use some sort of server.

Later I will show you how to do all these API stuff and stuff like this.

So that's pretty good.

Uh, we can say login or we can say register.

Both of them go to the same place.

We have fixed the problem with the size.

If we turn the screen now this looks better.

So that's good.

That's good.

Register.

Okay.

Register.

We need to put the password.

123123 and 456.

And I want to show you another thing because this is pretty annoying to put every time.

So in the controller we will say um text.

And we will define this as one, two, three.

And we'll do the same thing for this one text and it will be four, five, six.

So now when you restart the application, every time you restart the app and you go inside, get started, you will see that both of them will already be filled.

So 123456.

Right.

So I can just click register and see if the application works.

So right now the application is not working but um overflowed by this okay.

So we need to go inside the welcome page.

Is it.

Well no it's a home page.

And we need to put this column Refactor.

Wrap with a single child.

Scroll view.

Single child.

Scroll.

View.

Save.

So now this is working.

We can scroll and the rest is good.

Done.

I need to remove this in the profile.

If we open, everything is fine.

Perfect.

So for now we don't have any errors like this.

So I will put back the application like it was.

I think it's like this.

Perfect.

So that's about it for this video.

I know this video was kind of boring, but at least you know how to handle things.

When the screen change and stuff like this, you create some sort of single child scroll view and then you put everything in the center, right?

So applications start to look pretty -- cool.

I kind of like it.

What we'll need to do, though, is inside this page, because right now this is pretty ugly.

So we will change this in further video.

That's about it for this one.

And I'll see you in the next video.

Bye bye.

8. UI modification

Okay, so we are soon ready to talk about how to save data locally on the phone.

You remember this little button every time it starts it's dark mode.

It would be cool to have some sort of logic that saved the data locally about if it's white or dark mode.

Okay, and we will do this in pretty soon because we just need to change one thing for the UI.

And the thing is, in this video I will show you what it is.

If when you click Get Started it goes and use a push.

Okay, so let's go inside.

Welcome.

You see that this will use a push.

The push means that you can come back to the welcome page.

But when we go inside the login page okay we are using again a push replacement.

So it will replace.

I have created some sort of visual for you.

So right now it goes inside the second page with a push.

All right.

So this is our application.

And then what we do is we say push Bush replacement with the widget.

Three.

And the only thing we do is we replace this with this.

But we still have a back a back image.

Uh, or back page called the welcome page.

So when we click register, you see that we still have a back button to go back inside this page which was the first page.

Okay.

So we will change this.

And we will create a logic to take all of this.

And whatever the number of page previously we will say delete all of the page and then replace just one.

So it will remove remove remove.

And this first page will be replaced for the thing you want.

Okay so let me show you how to do this.

Hopefully the little image was useful and we can go inside the on login pressed okay.

So this is when you press in the login page this button okay.

It's this button.

And we will replace this for push and remove until okay.

The push and remove until will require a third thing, which is the boolean.

Do we need to stop before.

Okay.

In simple words.

So I would say predictable I will say control space control space.

And I will use route with the arrow.

For now I will just say false because right now the only thing we will say is I want to delete everything and then replace this page with the welcome the widget three.

Okay.

But what we could do is we could say to the code, for example, if the route is equal to the route number five or whatever, okay.

If you create a condition and it's true, then it will stop before.

So by example, if we say okay, we have a stack of four things and we say if it's if if it's a page two, then stop.

So instead of doing this, remove everything and replace the page one for something.

It will stop.

I will put everything back.

Okay.

What it will do instead it will remove this.

And if we say like page two, it will remove this.

And then it will replace this one with the with the number four or something.

Right.

I hope you get the point.

So for now we will just say false.

And because we say false, it will remove the the one that is at the back okay.

It will remove everything and replace it.

Oof!

This was complicated to explain.

Uh, maybe I could have simplified things, but I hope you understand with the image and stuff like this.

Okay, so for now, if we just restart, you will see that when you click Get Started.

Register.

Now you don't have the back button because this is the only page available right now.

So we can say profile logout and everything seems to work fine.

So let's work on this little thing on the next video.

This is called Shared Preference and we will save the data with a package.

We will save it locally.

That's about it.

See you in the next one.

Bye.

9. Shared Pref

It's now time to learn how to save the data locally on the phone, and we will save it if it's dark or light mode, right?

For this we will use shared preference and you can go on this website pub.dev slash shared preference and we will take this one.

So we'll click on install and we take this shared preference I'll copy this.

Go inside the Pubspec.yaml and add it command shift P add dependency shared preference.

And you should see it right there.

All right.

You can add it manually if you want.

If you add it manually just put this click the little arrow or run flutter Pub get.

Great.

So now you should have access to this.

I will save the file.

Close it.

And what we need is to use shared preference.

But I want to show you something.

And this something is this little diagram.

So we will use shared preference.

We will save data using a boolean.

We will retrieve the data using the same boolean, but we will only retrieve it when we start the application,

and then we will use the value notifier to display if it's either dark or light mode.

All right.

So let's start with the shared preference.

How you do this.

You go inside Readme.

And if you scroll down a little bit you should find right there preference set as bool.

All right.

So you set the value.

You put a key to find the value back from the phone.

And then you can use also this the read data.

So preference get boolean and you put the same key.

This will return you the boolean right.

So anyway I can talk all day all night.

But let's do it right now.

So first thing first we need to get the instance right.

So once you have the instance you can use the pref.

So we will go inside the code.

And where do we need this.

We need this on the button.

So the button is did I add it inside the pubspec.

Yeah.

So it's added.

That's fine.

Inside the widget three we have the icon button and this is where we will put the code.

We will say final shared preference.

We need to import shared preference, quick fix and import shared preference.

You will see it's this one.

And after this we need to use the await.

Await is just a way to say okay, run this thing, but wait for this to finish before running the next line.

All right, so it's just a waiting for something.

And in order to use await you need to make the function this thing, the yellow thing.

You need to make this function know that you will wait for something.

And in order to do this you need to write async.

Okay, so it's always together.

Await async.

And when you wait for something you always put the async before the curly bracket.

Okay.

So now we have access to prep so we can save the data.

We can go back inside Chrome and we can say await set Boolean.

So we will save the value.

Okay.

And you can see that we are using a weight again.

So the code will wait for this to run before running this thing.

And we can use the same async.

It doesn't matter.

Great.

So what will be the key?

The key.

Let's create this one inside the constant.

Just to make sure we can create one constant one class called k constant.

And we open the curly bracket.

And this one will be a what do we call it a static const.

And it will be a string and it will be key.

Um, how do we call this one the set bool color or is dark key.

That's good.

That's perfect.

So we will call this one is dark key.

Doesn't really matter because it will always use the same thing inside the phone to recover the data.

Okay.

So anyway so we will use the k constant.

K constant.

That is our key.

Or it would be better to just put dark key.

It would make more sense.

Or light mode.

Key.

Light mode.

T mode.

Key.

Theme mode.

Key.

And that's a little bit better.

Perfect.

So we can save.

And now we can change this for is theme mode.

And we will set this to the value that we receive.

So which is this value.

Right.

So copy this I will paste this.

But we need to change the value first right.

Otherwise it will save the old value.

So you need to take this and put it right there.

So we change the value and then we save the new value into the phone.

So this should already work.

Now the only thing we need to do is to retrieve the data.

So we say final boolean this thing Copy this and we go at the place where the applications start, which is in the main.

In the main we have a stateful widget, which is good because we need to use the init state.

So we'll use init state and in the init state.

What I like to do is just put one named function.

Okay.

So I don't want to put a lot of code inside this.

I want to see what will init step by step.

Okay I want to keep it simple.

So for this I will create a void for now.

And it will be um setup or init init.

Uh team mode is equal or no is not equal is a function like this.

And now you can just put this inside.

You can just say init team mode.

All right.

So that's good.

Another thing we can do is we can say preference.

So because we are using pref again we need to reuse the init Like this.

This thing you get instance, I mean.

So we will reset get instance.

We will use the shared preference as a library in port.

Right.

The await.

We need a async.

Now let's save.

And now this seems pretty good.

So um when we retrieve the information now we need to retrieve with the same key.

So the key was k constant the team mode key.

So that's good.

We have the repeat.

We have the information okay.

And with this we can set the value.

Let's enable builder.

Or how does it call this one.

Let's go inside the notifier.

So dark mode notifier.

So we close this.

And then what we can say is is dark mode notifier.

The value is equal to um repeat.

And we have a problem.

This value can be assigned to the value bool.

Try changing it.

So we'll define a default value which will be false.

Okay so let's try this.

Let's try to figure out if it works.

So we have the dark mode.

Let's go inside.

Let's register.

And now let's press light mode.

Okay.

Restart the application.

And it does not work.

So what is the reason.

Let's figure this out.

So error and a little misconception flutter shared preference.

Okay so I think we need to reset shared preference.

So I will start by closing the application.

I will flutter pub get just to make sure everything works I will rerun the application cancel.

I will save again verify that we have shared preference.

I will go inside this and relaunch the application again from there and go from main and start.

So I will see you in the next video.

Hopefully it will work.

Otherwise we will fix the bug in the next video.

That's it.

See you in the next one.

Bye.

10. UI modification

Okay, so now when I restart the application, you can see that the light mode is by default, which is great.

So if we put dark mode you will see that it is also working.

All right.

That is pretty cool because this is getting the information from the phone saved on the phone.

And use this in order to display either the dark or the light mode.

Okay.

But one mistake, a little mistake you can say is when I restart the application in light mode, you will see that for half a second we have the application in dark mode okay.

It's not a big deal okay.

But I just want to explain you why it happened.

And then we will move and work a little bit on the UI just to make it look a little bit better.

All right.

So, uh, let's log out.

And the first thing I need to explain you why we have the black screen for half a second.

The reason is very simple.

The dark notifier value.

If we go inside it by default, it is true.

So it is dark by default.

and the init the init state will not wait to complete before running the build.

Okay, so how the code will run is it will run the init state, it will run this and it will go inside and run all of this.

But meanwhile it is running all of this.

It will still build and start building with the value.

False.

This was a lot of words, but let me show you.

I will put some breakpoints at some places and you will see what I mean.

If we restart the application, you will see that it goes inside the init state first.

If we keep going, we go inside the init team mode.

Then we go inside the init team mode.

But even though we have an await and we will wait for this to, uh, to retrieve the information before

going to the next line.

This doesn't mean that it will wait in order to continue and build the application.

So the this should go instead of this line.

We should go directly into this line, but hopefully.

Yeah.

Okay.

Imagine if it was not the case how stupid I will look like.

But anyway, so you have this and now it starts to build, but it doesn't.

The the dark notifier is not changed yet, so let's keep going and you will see that after once we have retrieved the information, then we will change the is dark mode to the good value, the repeat value.

So now when you press play you will see that it's light.

All right.

So this was a lot of explanation for a little thing.

But at least now you understand how init work init state work and build work a little bit more.

Now let's change a little bit the visual.

If we go inside the application we register a couple of things that we need to change.

Okay.

First thing first, um, this page looks pretty cool.

I like this, uh, visual.

This would be interesting to create a button instead.

So we click inside and we go inside the page.

Then we could create many, many things like this okay.

So it looks a little bit better a little bit more full.

And then inside the profile we could create some sort of email, um, an image and stuff like this just to make it look better, right?

And while we are doing this, you can pick up how how I do it.

Like how I how my brain works.

So we can copy this and create your own app faster, right?

I'm not sure if this makes sense, but anyway.

So we'll go inside the home and first thing first, uh, let's log out instead and change this button.

So this button is kind of ugly in the white mode.

So if we go in white mode you will see what I mean.

If we log out and register, you see that this elevator button is kind of weird.

This thing is kind of weird also.

So let's change both of them.

So let's go inside the welcome page.

Let's remove this annoying.

Uh, I think it's this one.

No, it's inside the login page.

So inside the login page what we can do is we can remove first the hero widget like this.

Let's see what happens if I do it.

Uh okay.

And now we can add a Lottie asset.

You remember a while ago we were talking about the home that Jason from the Lottie Files.

Well, I've downloaded because I think it looks cool.

So we need to use it now in the Lottie asset.

I will just copy this line and I will paste it right inside the login page.

Inside the login page first we can remove the unused import.

And I know it's unused because we can see the yellow color and now we don't see it.

And next inside the scaffold we need the column.

And we need to put the Lottie file.

And now we need to import the Lottie package.

So quick fix import package Lottie which is this one.

And now it's working.

But we need to change the welcome for the home because this was the one I liked.

All right, I think it's I think it's a little bit better.

Right.

So the color doesn't really match but it's not a big deal.

So we go inside the register.

I will change it.

Maybe for a text button like this one.

So it will be like less visible.

Or we can use a field button.

I think we will go with the field button.

So field button.

Let's save.

Now I think this looks better.

Better.

So you can see right now we have the get started and we go both of them goes inside the same login page.

What we could do is we could create some sort of page when we get started, just to see an onboarding instead of just going inside the same login page.

So in the in the next video I think we can do this and, or we can do it right now.

Let's do it super fast okay.

So what I will do is I will copy the login and instead I will create a new page.

I will say new page onboarding dot dart.

And now let's copy the login page completely and I will paste it inside the onboarding dart.

Great.

Now let's change just the name.

Or we can even replace uh can we replace this.

Yeah we can replace this for a stateless widget instead.

So right click refactor I'm not sure if it will work.

No it doesn't work.

So we need to remove the overrides.

So what you do is you take from the override to the next override.

You delete all of this and then you change this for statelesswidget okay.

So that's good.

Now we are missing the controllers.

But this doesn't matter because we don't need the textfield anymore.

For this page, we can delete all of them and we can just replace the widget for title instead, because now we are not using the stateful widget, but the stateless.

So the stateless you just put the same name title.

After this the login pressed, we can delete this, and we can go inside the Onpress function and create some sort of navigation to go inside the login page this time.

So let's go inside the welcome page and I will scroll down.

I will copy this navigation dot push.

So let's take all of this copy.

But instead of going inside the login page I will go inside the onboarding page.

I think I need to change the name.

Yeah we need to change the name of onboarding the dart for.

uh, on my ding page.

Oops.

Oh my God.

On board the ING page.

Great.

Let's change this also for onboarding page.

Okay.

So we can remove the unused import.

And finally I think we have something pretty good.

If we go inside the welcome page now we can say onboarding page and the title we require a title but

I will delete this.

I will go inside onboarding page and I will remove the title required.

And the final.

So we don't we don't need this.

We can change this for anything else by simple default just to see what happened in the screen.

So when we click login we go inside the login which is good.

But when we go and get started we go up inside the same page.

So let's restart the application to see if it works.

Get started.

Okay, it's not working.

So in the welcome page let's change something.

We have the login.

Okay, so I, I did a mistake.

Instead of using it inside this we need to control Z.

Keep the login page because this is the login.

We need to change the register instead.

Okay.

So we need to change this for the oops the onboarding page.

Okay.

The the only thing I'm doing right now is I'm adding a page between the register and this welcome page.

Okay.

It's the only thing I do.

So let's restart and click Get Started.

Now you can see that we have an onboarding page first which is cool.

Okay.

Uh we can change this icon for something else.

I will do this in the next video.

Okay.

But for now we can keep it.

So we have this and we can just say, I don't know, Skip for now.

You just get the point of what I mean.

So let's go inside the onboarding page.

Replace the default for next.

And now let's save.

We can add also some text text.

Um Let's remove this for, um.

Flutter map is the way to learn flutter.

Period.

That's it.

Right?

Simple like that.

You put some sort of description, you can change the style for maybe a textile dot title.

Maybe we can try.

Yeah.

Okay.

Maybe description instead.

And let's save.

Okay.

So something like this.

We can also change the alignment if I remember text align.

Text align.

Is it this one justify I think justify is the good one.

Yeah.

Let's try this.

So this justify take all the width of the screen which is pretty good okay.

So we have flutter map.

Get started.

Login.

When you get started you have this little onboarding screen.

You can change all of this if you want.

It doesn't really matter.

But what matters is when you press the fill button you need to go inside the navigator that push of the login page.

Okay so quick fix and then I will say import the library.

And I replace this for register.

And now let's save.

So when we click on next or we'll restart everything when we click on next you see that we go inside the login the register page.

All right.

And when we go in the login we go straight into the login page okay.

These are the same page.

It's just the name that change okay.

So that's about it.

We have created an onboarding page.

Very simple stuff.

You can modify this to make it look better because we didn't we didn't pass a lot of time on this one.

But anyway this video has been already ten minutes which is crazy long.

So what we will do in the next video is we will edit a little bit of this thing.

And I really want you to to understand how I work with flutter, because this is how you will be able to build your own application or build build things by yourself.

Okay.

You have to.

Yeah, whatever.

You understand what I mean.

That's it.

See you in the next one.

Bye bye.

11. UI modification

Okay, so for this video what we'll do is inside this onboarding page we will just change this for another JSON.

So I have downloaded hide the JSON.

It's just a little welcome.

So you don't have the same image like twice because it's kind of weird.

So let's go inside the onboarding page and change this asset for hide JSON.

And the height for the JSON was just a little welcome.

Like this I think it's pretty pretty cool.

All right.

So welcome.

Maybe we can put some sort of size between both of them.

So let's cut this and put it right there a size box.

So now we have some sort of description and we have the welcome.

And we can say next pretty basics but that's cool.

So let's click on Get Started.

We have the welcome.

And you say next I think it's a better transition um color wise right.

So that's good.

Uh, we can now uh, maybe we can put some sort of text or anything.

Whatever.

So that seems good.

Another thing we can do is modify this.

All right.

So we need to add more buttons like this.

So I will go inside the welcome page and we can it's not welcome page.

First I will close all of them close all tabs so it's more organized.

And then we go inside the home page.

So let's go home page.

And inside the home page what I will do is I will add another container like this one.

So this container let's create a widget a custom widget for this one.

So inside widget new file and we will say container widget widget dot dart or something like this.

Uh yeah it seems good.

Container widget dart and press enter okay.

So now we can say import M and say stateless widget and call this the container widget dot dart uh container

widget.

Just like this.

Let's go back inside the home page dot dart.

And then what we do is we take this entire thing and we cut it and we say the container widget instead.

Container widget.

Great.

And when you go inside container widget you replace container for what we just cut, which was this

entire thing.

We just need to import the K textile.

So quick fix and import this one.

And inside this one we need to remove the unused import.

Great.

So if we restart you will see that nothing really changed.

Um so get started next and register.

Nothing really changed.

But now we have this as a reusable widget.

So now we can just go inside container, maybe change something.

The title we can put some sort of arguments we say required this dot title and remove this dot description.

Now we just need to create the finals strings.

So strings title and final string description.

Great.

And one tip I can give you in life is if you like coding and stuff like this, you should highly consider

learning how to touch type, right?

It's just like uh, you just right touch typing online.

You practice.

This is how I'm able to write.

I don't write really quick, but it's not bad.

You know, if you write and you have to look the keyboard, it's terrible.

It will take a lot of time.

So it's worth learning.

It will take one month and that's it.

Anyway, so let's put the title.

Let's put a description and now let's save.

Let's go inside the home page okay.

And add the two required arguments.

Title will be I forgot what it was but it was basic layout.

And then the description was, uh, this this uh, this is a description or whatever.

Right.

So now we have a reusable widget which is the container widget.

We can copy and paste right under to see what happened.

The space is a little bit too much.

So what we can do is we can go inside the container widget.

And uh, what we'll do is we'll change the padding of the container and we'll do only the top.

Okay.

So I will say I will replace the C matrix for only and I will say top like this.

And now it does look better.

Let's use the dark mode to see if everything looks fine also.

Maybe I can log out and see this homepage.

This looks really cool.

I really like it in in dark mode.

Anyway, so that's good Register.

And now what we need is, um, we have two of them.

Maybe we can add a third one.

So let's go inside the home page and add a third one.

And four and five doesn't really matter anyway I need to show you another widget because you will see this is pretty cool.

And another thing we can see that, uh, right now it is going behind.

So we should have some sort of padding around the entire thing, but we don't need the padding, uh, in the vertical.

So let's change this.

We have the edge insets that all.

Exactly.

So we will say symmetric instead.

And we use the horizontal 20.0.

Great.

So now you see it looks better okay.

Let me put white mode.

You can see better.

So that's good.

Uh let's talk about the other widget I want to show you which is the list dot.

Uh let's talk about it next time.

See you in next one.

Bye bye.

12. UI modification

Okay, so I was not supposed to talk about this one, but I think it's a good moment to talk about it, so let's just do it.

One thing you can do with flutter is you can say list, dot, generate, and this will generate a list of multiple same thing right.

So let me show you what, what I mean.

I will click this.

Now what I realize is I need to delete all of these.

Delete this one.

Cut it.

And now maybe it will work.

Great, great.

So the length is just how many items you want to generate.

We will say five for now.

And the generator you can delete and control space.

You will see that this require a function which have an index right.

So you'll say this will return the container widget and you put your semicolon.

Just make sure everything is the same.

And now you can see that we have some sort of error right.

And the reason is I will put my comma.

The reason is the list that generate is a list of widget okay.

And right now the column is not required.

It does require a list of widget, but not for one widget.

So it's like you try to put one widget.

One widget.

And then you try to put five widget in one go.

It doesn't work.

In order to fix this, you can either say column like this and use children, and then replace the list for the list that generate.

Okay, this could be one way to solve this.

So let's cut and paste inside and save.

So this will work.

So if I restart you will see that we have five items inside the list 12345.

Okay.

We can put three.

So you see the difference right.

Three items.

That's cool.

But the other way you can do it is you can just say three dots.

And this is a little shortcut with flutter.

If you don't remember it doesn't really matter.

But um you can use dot dot dot.

And this will say to flutter that okay I want to add many widget in one.

Go inside this list.

So I want to add a list inside a list.

But.

Yeah.

Uh, my explanation is kind of garbage, but you understand what I mean?

So now it does work.

You are telling flutter.

I want to add many widget inside this list.

Okay, so it's a shortcut.

If you remember, you remember.

Otherwise, it doesn't really matter because the column does the job really well.

The next thing you can do is we could create some sort of, uh, value that will have basic layout and other stuff.

So let's go inside the constant.

We can create another class and this one will be K.

Um I don't know, k value or k value is equal to or let's open the bracket.

Oops.

Let's open the bracket.

Can I open it.

Come on.

All right.

Perfect.

So k value.

The next thing you need to do is static.

I completely forget what was the purpose.

So let's go back inside the homepage.

And yeah the goal is to write some text.

so basic layout will be the first one.

Static const string and basically out.

Basic layout is equal to basic layout.

So you get the point of what I'm doing right now.

I will do just three of them, or maybe four.

Let's do four.

And I will just say, uh, what we have learned inside this course, maybe we could do that.

So let's say for example, basic layout and then let's say clean UI and then let's say fix bug and stuff like this, stuff like this.

And then we can say key concepts.

And replace all of them key said.

And then clean.

Oh my goodness.

Terrible clean UI.

And the last one fix Bugs.

All right, that seems good.

Enough is enough.

So let's go back inside the home page, and we can use these values instead.

So you should do this for every, uh, every data that you have.

You should put all of them, like flutter map, basic layout.

This is the exception.

You should put all of them inside a file.

Constant or everything like this.

It's not good to put random magic word and random magic number inside the code, but just because we are, uh, in a teaching style, I just want to go fast and don't waste your time.

So that's about it.

Anyway, so let's go and remove this and we'll say k uh value if I remember.

And it was anyway k concept.

Okay.

What we will do at this point is we will create a list.

I will say um list.

We can even create it inside.

It doesn't really matter.

So I can say, uh, list of strings.

List of strings I have difficulty with the keyboard right now.

So list of string and this will be called just a simple list.

Let's create this list.

And inside we will just put all the constants.

So it was k value dot key concept.

Then k value dot.

Uh let's use all of them 1234 replace boom clean UI and then fix bug and then basically out okay.

And we can use the list instead of using three we will say list dot length.

And now we will have four items.

But we need to change the name also for the name that is inside the list.

And you do this by using.

Instead of writing this you will say list dot element at and you put the index from the list generate and this will automatically target the one, two, three, four and display the value.

So that's about it.

Uh, you could create some sort of button.

You could wrap this with a gesture detector if you would like, in order to make it clickable and go into the page.

But for this course, we will not touch these things anymore, I think because it's all redundant, it's always the same thing.

You have learned what it needs.

All right.

So what we'll do instead is we will work on this one.

We will create a gesture detector for this little button, the hero button, because we need to at least use the hero button.

Otherwise it's completely useless.

So let's go inside the hero widget.

We will change this for a gesture detector.

So over the stack.

Refactor gesture detector wrap with a widget gesture detector.

Great.

And now I will say on tap.

And on tap will trigger a function.

And the function will navigate into another page.

So which page.

Which page are we navigating to.

It would be We can say, let's create something new.

I will say this, that, uh, can I.

Yeah, we can do it inside this video.

We are at seven minutes.

Three minutes left.

Let's do it fast.

So this dot, uh, next page.

Okay.

And I don't put required because this value will be able to be null if needed.

So I'll say this will be, uh, for now we'll just say string, but I'm pretty sure it will not be a string.

I think it will be a widget.

So widget next page like this.

Okay.

And now we need to tell flutter that this can be null.

And because it can be I need to change this for the comma.

And because this can be null now then it works.

We are not required to put the information because if we don't put it, it will be null, right?

So the next thing we need to do is to navigate.

We don't want to we we don't really want to rewrite everything navigation blah blah blah.

It's super boring.

So what I like to do is just copy one from the Wellcome.

Work fast, not order.

So copy this and then you just replace this.

Save.

And now the login page.

What we will do is we will return the next page like this.

But the thing is we need to make this available only if we have a next page value.

Right.

So what we can say is we can say if the next next page, if the next page is equal to.

Is not equal to.

Null is not equal to null.

Question mark.

If it's not equal to null, then do the function.

Otherwise do nothing null okay.

So this.

Since we know that this will not be null.

We can say that I'm sure that this will not be null and it will work so great.

Now we are able to use this as a button.

Let's just do it in one minute.

So when we click on it, what we want is let's go inside the home page.

We are using the hero widget.

And we can replace the.

And we can put the argument next page and create a page.

Um, for now what we could do is we could just go inside the onboarding page, um, and just save.

So when you click on it, you go in the onboarding page.

But that is not the best way to do it.

So instead what I will do is I will just copy a new page.

I will say new page.

I will say course dot dart, course page, dot dart.

Like this I will import material.

And then what we can do is we can just copy a page.

I will copy the home page.

So I go in home page copy all of this Ctrl C go in the course page copy all of this okay.

So now we can remove a couple of things.

The list will not be useful anymore.

Then the laser generate useless.

We can delete this.

The hero widget.

We can remove the next page.

So we just see the top of the widget.

We just see the hero widget and we delete all of this.

Let's save replace the name for course page.

This video is very long.

Sorry for that.

And now this seems pretty good.

So we can say this is a course instead and save.

Let's delete this home page.

In the home page what we'll do is instead we will redirect to the course page.

So now inside this course page we should have the hero widget but with the text course.

So if I click on it it does work.

But we have one problem and is that we don't have any scaffold okay.

So inside the first we remove the on unusable uh import.

Let's delete this.

And next let's go inside.

Um which widget are we course page.

And we can wrap all of this inside a simple scaffold.

So cut.

Scaffold.

App bar.

App bar.

Empty.

The only reason I do this is to have the back button body and we paste the body.

Save.

Now you can see that we have the back button.

And when we click on it you see the little animation.

This widget doesn't move and doesn't even move.

And I think it's pretty cool.

It's because of the hero widget.

All right.

So that's good.

What we can do next is um, let's use the dark mode to see if this looks cool.

I can click on it.

Pretty cool.

And yeah, that's pretty much about it.

In, uh, in the further video, we will be able to work a little bit on the profile page, but I think

for now this is pretty, pretty good.

I hope you you like this so far and I'll see you in the next video.

You will see this one is -- pretty cool.

That's it.

See you in the next one.

Bye.

13. Launcher Icon

Okay, so I told you I will show you something pretty cool.

And this thing is how to change your flutter app icon.

Okay?

Because currently it's the it's the flutter logo.

But we would like to have our own logo.

So for this what you will need to do is you will go on Pub.dev and you will search for flutter launcher icon.

First thing first let's go in installing and I will take this dependency and add it inside my code.

So I will say uh command shift p add dependency.

And then you will paste it and press enter okay.

So you will see it will appear right there in a while.

So save again.

Let's save.

It didn't work.

So let's try again.

Add dependency paste and press enter.

So this should be inside perfect.

So we have the flutter launcher icon.

If you don't have the add dependency you can just write the flutter launcher icon as usual.

Next you will run Flutter Pub get just to update the file and everything.

So let's go back inside the code.

And now the next thing you need to do is go inside Readme.

And in readme, you can scroll down and you will see that you have two options.

You can either set up with a config file, which it's.

I prefer this one because it's faster, but if it doesn't work for you, you can either use this so

you will be able to put all of this information inside your Pubspec.yaml.

So this file.

Right.

But we will generate the thing inside this video because it's cooler and faster.

So run the following command to create new config.

So I will just say copy and boom we go inside the terminal.

And where is the terminal.

Terminal.

We can delete everything.

Oh no this is okay.

So we have the terminal.

And now we can paste dart run flutter launcher icon generate.

So when we press enter you should see that.

Now you have this file the flutter launcher icon okay.

But if it doesn't work for you, you can just enter all the information in the pubspec.yaml like I told you.

All right, so let's modify this thing.

First thing first Mac, macOS, windows and all this stuff.

We don't need this I will delete.

Okay.

If you need them, keep them.

But for now we will just use Android and iOS.

Okay.

If you want you can create uh, an image path Android and an image path iOS.

But for now, what we'll do is we will just keep one image.

So we keep it simple.

The image won't be inside this file.

Instead it will be inside image and it will be.

Let's add it right now.

It will be logo dot png this one.

Okay, I add it inside my project.

And now, uh, again, you just have to remember that if you create another file, you will need to

go inside your pubspec.yaml to add one.

Okay, this is working because we already have the asset image.

Just keep this in mind.

After this inside the flutter launcher icon, what we will do is we will change the path and it will be image and then it will be logo.png.

Great.

So for the rest, we can keep everything.

Remove alpha channel.

This is mandatory.

You will see later.

It's because Android iOS doesn't have any transparent logo.

You can see there is no transparent logo and this one is transparent.

So it will automatically fill the the transparent.

Okay, so that's good.

Next we will save this file Ctrl s.

We can close it and then we can go inside the flutter.

Um we can go inside the terminal.

Actually we will go back inside the website if I'm able to find it.

Perfect.

And what we will do next is you will run flutter pub get and dart run flutter launcher icon.

So let's copy this.

Go inside the terminal and paste and press enter.

Great.

So this will resolve everything.

Uh, flutter pub, get launcher icon create default icon warning you can see the warning icon with Alpha channel are not allowed in Apple Store.

So that's why we have the.

If we go back inside flutter launcher icon.

That's why we have this thing remove alpha channel.

True okay.

So that's good.

Everything seems pretty good.

So what we'll do is uh, currently my application is closed.

So I can close this.

Also, my application is closed, so I don't have the application running in the background of the phone.

So is it.

So I will kill it if if it was running.

But anyway we will relaunch the application.

Because if you keep the application launched and you do all of this and then nothing happens, it's because you have to close the application and relaunch it.

Okay.

So I will say I will go inside Main.dart and I will click this little thing.

Boom.

And I'm launching the application okay.

So if you go back inside the flutter launcher icon, meanwhile I want to show you a couple of things.

So first thing first the minimum Android um, the minimum SDK Android.

This is inside if by if by example, you need to have this information and what it is inside your, your your project, you can go inside Android and then Gradle and then it is Build.gradle.

Is it this one.

Is it Build.gradle.

Gradle.

Gradle.

Gradle.

No it's inside app and Build.gradle.

This one because there is two Build.gradle.

There's one inside app and there is another one.

So let's go inside app Build.gradle.

And you can see that we have the minimum SDK.

And if you want you could change this for 21 or something else okay.

If later inside your flutter journey you are required to do this, at least you will know where it is.

Great.

So that's good.

Now let's close this and see our application.

And now you can see that we have the flutter launcher icon.

That is pretty cool.

But uh, what we'll do anyway it's for the next video.

That's it.

See you in the next one.

Bye bye.

14. Circle Avatar

All right.

So let's talk about how to create a circle avatar inside your app.

And also what we will do is we'll change the UI.

Let's let's change the UI first because there's a couple of things that is pretty ugly.

So for example the space between each item is pretty big and this is pretty small.

So let's let's fix things.

And then I will show you the circle avatar, which is a pretty cool widget.

So what we'll do first thing first is you see how this little image is going straight into the app bar when we scroll as soon as we scroll.

So it's really so close.

So what we need to do is create some sort of space between both of them.

We will go inside the home page and inside this one we have the hero widget, which we will add a sized box over.

And maybe you wonder okay, but why don't I use the padding for this?

So let's use it.

If we if we put a vertical padding of 10.0 right, what will happen when I scroll?

You can see that we have some sort of, uh.

Yeah, it's called a padding, but it's like, um, a black bar, so it's kind of weird.

And it does the same thing with the bottom.

You can see right there.

And we don't want this.

So what we'll do instead is we won't use the padding because the padding reduces the available space.

Instead of using the padding we'll remove this and I will save.

We will use a size box.

So over the hero widget we'll say size box height of maybe 10.0.

And let's put all the commas like this and save.

Great.

So this is already better.

When we start to scroll we don't have the image right into the app bar okay.

The next thing we need to do is to reduce the space between each item.

And we can do this inside the container widget.

So we'll press command and I will click this.

So you can press on Mac.

On windows you can press Ctrl and click next.

We will go inside the padding.

So the container padding.

and instead of using vertical, I think it was 20.

Is it 20?

Padding.

Okay.

Sorry.

It's this one.

So instead of using ten, we will use five.

Okay.

This is the 20.

All right.

Perfect.

And now we will use five.

So this will be already better.

Next what we can do is we can go back inside the home page.

And we will add a little padding right there.

Because right now the space between both of them is not really the space between the image and the first one.

So we will add another size box right under and save.

And this looks already better.

Maybe what we can do I will restart the application just to see if everything is fine.

Let's go inside register.

That's good, but I think it's still a little bit too much.

So instead I will use five.

I will go inside the container widget and what I will do is for the vertical let's put zero and save.

Let's see what happened.

So when we put zero you can see that we have the cart which creates some sort of automatic padding.

So the cart have a padding by default.

So that's why what we'll do is maybe we can just say, um, padding inside the cart.

So elevation margin surface ship ship ship ship margin.

Let's use margin edge and sits and sits dot zero.

Let's try this.

Sorry.

And now you can see that all of them are now closed or very close to the other.

So you can, if you want, change this because the card will automatically create some sort of padding of margin.

Right.

So what we'll do is I will remove this, I will just put 2.0 and save.

And right now with this it looks pretty pretty much better of what it was previously.

All right.

So we have this next.

We need to go inside the profile widget.

And what we'll do is we will add a circle avatar.

Like I told you at the first at the beginning of the video inside the profile page, and we use inside the children circle avatar.

And now let's put the comma in save.

This is the circle avatar.

You can change the size by saying radius and maybe 50 dot zero.

And you can change what is inside.

So we will say background image.

And you probably think like okay, I need to say image that assets and then write the path.

You are right.

But with the circle avatar there is something a little bit weird that I will show you after.

So instead of yeah, we will put the path first.

So I'll say asset slash, image slash I will use the this image, the background dot jpg okay.

Because I want you to I want to show you how the circle avatar will automatically put this image as a circle.

So the background image we use this and we say bg dot jpg.

Great.

But now we have a problem.

We have this thing saying blah blah blah, we can't use this thing.

Instead, we will need to use the reverse.

I know it's pretty weird, but you will have to just remember somehow.

So it's a set image, but you will use this widget pretty much only inside the circle avatar.

So that's it.

So when we save now we have our circle avatar.

So this could be your profile image or something like this right.

So that's pretty that's pretty good.

Maybe you could write some sort of you could create more stuff to make your application look better.

But all of this is all redundant.

So it will be always the same thing.

So I don't need to show you again and again and again because this will just waste your time, right?

So for now, we have talked about pretty much everything inside the package section.

And I will see you in the next module.

But the next module I will say nothing, but the next module is pretty cool.

So see you in the next one.

Bye.

10_responsive-ui

1. Fractionally Sized Box

Okay, so you will see this.

This module is really cool.

Okay.

We will talk.

Um, first we will talk about the fractionally sized box.

But in the next video we'll talk about the layout builder.

And this is where you will see that it's going from beginner to advanced stuff.

Okay.

What is the layout builder.

First thing first, the layout builder that we will use in the next video is just a widget that allows you to have the information about the width and the height of the screen, and by having this information you are able to by example, I opened an iPad, I opened the entire project on an iPad, and you can see that this looks pretty stretched out and it looks super weird.

So what we will do later is we will say if the screen is bigger than X amount, then I want to change the the visual.

And for this you can use the layout builder.

But before I show you the layout builder, I need to show you another widget.

And it's called the fractionally sized box.

The fractionally sized box.

What we'll do is we'll go inside the login page, which is this page, and we will need to make it look better because on an iPad it looks really strange.

So for this you can go and wrap the entire column.

We'll say refactor and I will use a wrap with widget.

We'll say fractionally sized box.

So the fractionally sized box have some sort of a width factor and the width factor.

If I say 0.3 this will take only 30% of the screen for the width.

All right you get it.

And if we put maybe 70 this will take 70% of the screen.

So let's put 50%.

And now it look just better.

Okay.

We could do this for all the other widgets, but, uh, it's just you understand the point.

And if you understand the point, that's good.

All right, so we don't need to waste time, uh, doing the same thing 500 times, so that's good.

But in the next video, I want to show you how you how you can create this, um, more responsive,

like, more a responsive app, which, uh, let's go.

Let's jump into the next video and not waste time.

That's it.

See you in the next one.

Bye.

2. Layout Builder

Okay, so in the previous video I've tried to explain to you at the end something, but it's just sometimes simpler to just show instead of telling.

And this is what we'll do.

We'll use the layout builder and you will understand right away.

Let's wrap the entire fractionally sized box with the layout builder.

So because the layout builder is a weird widget, it won't work with the refactor, blah blah blah.

So I will cut the entire fractionally sized box.

So from there till this thing, I cut all of this.

Okay, cut.

And then I use the layout builder.

Great.

So the builder will be the context constraints which will return what we had previously.

Oops sorry for that.

So it will return what we had previously.

So I will just paste back what we had put the semicolon and save.

Okay.

So we can see that right now we have the layout builder and we have the fractional size box.

And this is the builder.

Okay.

Next what we'll do is we have access to this constraints.

You can define the type if you want.

So it looks more like professional somehow.

And you can use the constraints in order to say if you want to, uh, split the size of the screen by 50% or not.

And you can say constraints The max width and if the screen is bigger.

Oops.

If the screen is bigger than 500, then it will be 50% of the screen for the width.

Otherwise it will be 100%.

It will be the entire screen and it will act normally.

So let's save.

And now, because the screen is bigger than 500, let's find out what is the size of the screen.

I will add a red point.

I will restart.

And now if we go inside next next you can see that the width of the screen currently it's 990 992.

Okay.

So we will say now if the screen is bigger than 1000 which is false.

So this should go inside one and it should act normally.

So if I restart the application you see that.

Now this is going to act normally.

So if the screen is bigger than 1000 then it will act as 50%.

Otherwise it will be one.

But this doesn't make sense right now.

So we will put back this to maybe 500.

And we will see later if it makes sense with the, uh, the iPhone.

So we can, uh, what I will do, uh, behind the scenes is I will close this thing and I will reopen the iPhone so you don't have to to just watch me doing it.

That's about it.

And I'll see you in the next video.

Bye.

3. Media Query

Okay, so the layout builder is a pretty cool widget, but do you really need to use this inside every application you create?

Not really, because this is more if you want to create an iPad in an iPhone application that will work for both of them, but usually you just want to start with Android and iPhone, so you don't really need to use the layout builder.

But what if you need to have the size of the screen?

You can do this without using the layout builder and it's a little bit faster.

So okay.

A lot of words, but not a lot of the, uh, not a lot of code.

So let's code right now.

Media query dot of context.

So of context dot size dot width okay.

And this what it is it is a double.

So you can say double width uh with screen by example is equal to media query, blah blah blah blah blah.

So this will give you the information of the width of the screen.

You can put height if you want, but for now we need the width.

So we can just use this and put it inside instead of this instead of using the constraint max width okay, we can just put this and it will do just the same thing.

So you have still the screen of the width.

But yeah that's pretty much it.

But the thing is why do I show you two things.

Usually I like to just show you the best practices so you don't have to like, learn a lot of things for nothing.

But for now, this you kind of need to learn both of them.

The layout builder will be more useful if you want to create some sort of iPad, iPad, and iPhone application.

But if you just need to have the information about the width of the screen, then you can use just the media query.

So that's about it.

I wanted to show you both ways to do it because these are pretty interesting.

That's about it.

See you in the next one.

Bye bye.

4. Aspect Ratio

Okay, so this one might blow your mind and it's pretty cool.

So you can see how currently the application when we go inside oops.

When we go inside this thing is pretty big and it doesn't really look good.

Okay.

So when we go inside it's still the same thing.

But it's it's weird.

Maybe it would be better if the height of the thing will be smaller.

So for this we can use an aspect ratio.

The aspect ratio will create some sort of, uh, by example, a resolution.

You know, when we go, uh, a video, a video online is usually this resolution 1920 by 1080.

Okay.

Screen size whatever.

You understand what I mean.

So you can use an aspect ratio and give it this information in order to display the size of this thing.

And this is what we'll do.

So we will remove this and we will go inside the home page.

In the home page, we have the hero widget.

So let's go inside the hero widget.

And inside this one we should have the image.

So we have the image the Clipper rect.

We will do it over the clipper.

So let's refactor, refactor and wrap with a widget which will be the um how the aspect ratio.

Great.

The aspect ratio will require the aspect ratio.

And this, this thing, you can give it exactly that.

So it can be 1920 by 1080.

So let's do this 1920.

But you will need to divide by 1080.

So now if we save you will see that this is really like a basic screen 1920 by 1080 it's the same like format okay.

So next what we need to do is inside the image set we'll say fit fit, box fit, dot cover, box fit, cover discover and Save.

So now you can see that this is now an aspect ratio of 1920 divided by 1080.

But you can use something else.

Maybe you can put two and save.

What will happen if you put two.

Just mean that for each um, for each height it will be the double for the width.

So for example, if I say 0.5, you will see that it will be completely the invert.

Now we have for uh, we need two width in order to be the same size of the height.

So this is how you can use the aspect ratio.

I will put back the 1920 divided by 1080, because I think it's pretty cool just to use this and be able to display a widget that will, uh, mimic a screen resolution.

I think it's just cool.

And I wanted to show you this so you can click on it, and you can see that both widget will now have this aspect ratio, a pretty cool widget to know in order to create some sort of responsive layout.

That's it for this one and I'll see you in the next video.

Bye bye.

5. Expanded & Flexible

Okay.

So let's talk about the expanded and flexible widget.

These are pretty cool.

So let's first think first.

Go inside the setting page.

We will need to create another page.

So I'm able to show you pretty much everything about these widgets.

Okay so in the setting page we have uh some elevated button.

Some fill button.

We will change one of them and navigate into another page.

So I will say um show um full and expanded and save okay.

So when we will click on this button it will go to another page.

And we can do this with a navigator.

Do we have one available maybe in the welcome page.

Yeah we can use this thing.

We will copy this navigator and close and go in the setting page.

Paste the navigator inside the button we created save and we'll go inside another page.

So let's create another page.

And then we'll call this the expanded flexible test dot.

And enter.

Great.

So now we can import material and say state less widget and it will be expanded flexible test.

Great.

So we should instead say page.

It will make more sense to reuse always the same logic because each of the each page have the page at the end.

So it will make more sense.

And I will change this also by saying page.

Great.

So now we can use a scaffold widget a scaffold widget.

And yeah that's about it.

We will use a nav bar, a bar and then a body.

Oops.

Body column like this.

And now let's save.

Okay.

So now we can navigate inside the expanded flexible page.

And I will remove this login page.

I am inside the setting.

I just jump into the setting and I will say expanded flexible expanded flexible page.

Great.

So now when we click on show flexible expanded we are inside this page.

That's good.

Now I can show you how to use the expanded and the flexible.

So inside the column we will say children.

And inside this one I will say expanded.

Inside the expanded we will use a container.

And the container will have the color colors dot teal.

Let's use teal for now.

And let's save.

Okay.

So the expanded will take as much space as possible.

That's the only goal of the expanded.

And because we are inside a column the expanded will take as much space as possible vertically.

Right.

But if you were inside a row, it will take as much space Is horizontally.

You have to take this into consideration.

And also remember that when you use flexible and expanded, you need to use them inside either a column

or a row, usually okay.

Otherwise it can create some sort of bugs.

So that's good.

What I want to show you next is inside a column.

What you could do is you could create another expanded and call this one arrange.

So now let's save.

Both of them will take as much space as possible, but they will share the space.

If you don't want to share the space, you can tell that you can say that the flex of this expanded

is twice as big, so it will take twice the space of the teal one.

And you can see on the screen right now.

Next, if for example you say flex four, this one will take twice the space as this one because it's

two times bigger.

So let's save it will do just the reverse.

That's good.

Now let's remove the flex I want to show you other stuff.

So this was how to create a the expanded.

But, um, let's let's remove them and let's use a row instead.

Okay.

Inside the column I will use a row and I will say children.

And I will say X, I can just copy what I had previously, uh, copy and boom boom boom boom, boom row.

And I will just paste them like this.

Okay.

Now we have an expanded inside a row.

But the thing is, we don't see them.

And the reason is the expanded will take as much space as possible.

But within what?

Within the parent.

So the parent is a row.

So it will take as much space as possible.

Uh, horizontally but vertically nothing really happened.

So in order to show something you will need to put a height and the height.

Let's put 20.0 for both of them like this.

And now you can see that the flexible is working once again.

And it takes as much space as possible.

And it shares the space.

It does the same thing.

Okay.

That's good.

Now let's do something else.

The something else is we will use a flexible widget.

So flexible.

Flexible widget.

The flexible will do the same thing.

So you can see that it takes it share the space.

It will share the space with the expanded like it does.

But the flexible have something different and it will automatically shrink the child widget.

So if we go inside the container right now, we don't have any child in the container.

But if we put a child with a text saying hello and save and put both of them both child hello, you see that the expanded will still, uh, will still show the entire thing, but the flexible for the arrange color.

It will stop right away for the child.

But if the child go bigger like this and save, it will take all the space.

Okay, this is the difference between the flexible and expanded.

They are pretty similar, but sometimes you will see some difference.

You will have to practice with them in order to really understand them.

But I want to show you a little bit more examples.

So if we take another row, I will copy this one and paste it right under.

If we take another row and we change the order, let's put the flexible first and expand it after.

So cut this one and paste it right before and save.

Okay I will also add a divider so we can see better the divider like this and save okay so what happened in this case.

In this case you can see that the the second expanded even though it's sharing the space with the flexible the second expanded will go straight next to the the first flexible.

And all the available space right there is kind of used by the flexible widget.

This one the the flexible widget.

Okay, so I know this is weird, but what if you put in the flexible?

You say, I want this to be the flex.

I want it to be four times bigger than the expanded.

You will see that the expanded will shrink if I save.

Now the expanded is smaller because all the empty space is like kind of used by the flexible.

Even though we don't see it, it's kind of used anyway.

So this is how you use the flexible and the expanded within column and rows.

Just keep in mind when you use them, try to use them inside a row in a column.

And keep in mind that when you use expanded or flexible, it will go to the same direction of the parent widget.

If it's a row, it will go horizontal and it's if it if it's a column, it will go vertical.

One last thing I need to show you is what will happen if you wrap the column.

Refactor with a sized box widget.

So Sized, uh, not a size box, but, um, single child scroll view.

So now if I save you see that in the the code?

We don't have any bugs.

But what if I put an expanded widget inside this?

So expanded widget and the child, uh, let's put the same thing.

So I will just copy this expanded copy and paste.

Perfect.

So now you will see that this creates an error.

And the reason is the expanded will take as much place as possible vertically because it's inside a column.

But we are inside a single child scroll view which means this is infinite.

So yes, this will create a bug.

All right.

So you need to be careful when you use expanded and flexible widget because this can create bugs inside your application.

All right.

So that's about it.

Um if you want you can keep the page.

It's up to you.

It doesn't really matter.

So we will keep it for now.

And I'll see you in the next one.

Bye.

11_networking-api

1. Intro

In this module.

What we'll do is we'll talk about networking.

So networking is used in order to get data from the internet and display inside your phone.

So by example you go you get the data from an API and you display this inside the phone.

What we'll do is we'll really go step by step, okay.

Because I don't want to create a 35 minute video.

I want to create little videos that are easy to understand and digestible.

Okay.

So I just need to explain you first inside this video what we will do.

Okay.

The big picture and then each video will do one little step.

So the first thing is we will use the board API that app telecom okay.

You can use any other API if you want.

It doesn't really matter.

But we will just use this one because it's pretty uh it's pretty simple right?

Simple stuff.

After this you will need to go on http.

Um.

It's not this website, it's doc.dev cookbook networking fetch data.

Okay.

You will be able to find this page.

And next you need to use some sort of assistant ie if you want.

If you don't like this, don't use it.

But I highly recommend you to use this, uh, because otherwise you will be lost in the sauce later

in your life anyway.

So first thing first we will need to put the HTTP package inside our application.

So we will do this for this video at least what you can do is inside this page you will be able to find

the HTTP package.

So you can open the link in a new tab.

And this is just the simple HTTP package provided by Pub.dev.

So let's install this inside the Pubspec.yaml we will go inside Pubspec.yaml and I will say command shift p, add dependency and then boom http.

And you should have something like this inside your pubspec.yaml after this we will save and we will flutter pub get just like that.

And now you should be able to use the HTTP package inside your flutter app.

And this is what we will do inside the next video.

So I'll see you in the next one.

Bye.

2. Get

Before we start with the HTTP package, I need to tell you something and it's inside the Fetchdata.

If you scroll down, you will see that you need to use permission sometime.

Okay.

The the application with Android, you will need to add the permission if you want to release in production.

If you are developing or debugging, it will be fine.

The internet and the API will all works, but if you are publishing on the App Store then you will need to add this.

All right so I will show you where to add this.

And it's only for Android and Mac OS.

So if you deploy on Mac then you will need to do this also.

But for iOS it's fine.

So let's add this inside the Androidmanifest.

So where is this file?

First you will need to go inside your project.

You can minimize everything and you will find inside Android you should be able to find inside application and inside source and inside main android manifest.

So you double click on this and this is where you need to add the used permission if you are developing deploying on a production.

Okay.

So I will copy this and I will paste it just before the application like this.

Boom.

And you just make sure that is aligned with application.

So in my case it is not.

So I will do this.

And now everything is fine.

If you don't remember how to add it you can always go inside the Java I think.

Is it Java or.

Yeah.

Debug.

Inside Android manifest you will see that this debug engine manifest already have the use permission internet.

So it's the same thing.

We just add it inside the production one.

All right.

So that's good.

We can close both manifest and we can start building our HTTP request.

So we have added the installing.

In the previous video I've shown you how to add the HTTP inside your dependencies.

Now let's use it.

Let's go inside example and inside example.

We will find um, just this little code and we will add this code inside our project.

We will go inside our application.

I will close the pubspec.yaml and open, uh minimize everything open inside the pages.

So inside view pages we should have course dot page course page dot dart.

Let's use this.

And inside this one we'll modify the stateless refactor.

And we'll use a stateful widget like this.

The reason we need to do this is because we will use a init state.

If you remember, the init state will trigger as soon as the page starts, it will trigger this and then it will start building the page.

So we need to get the information get data at this point.

So we'll create a function, get data void get data like this.

And then we can open the bracket and put some code inside.

So now we will be able to see the response from the API inside the inside the terminal.

That's the goal at the end of this video.

After this we will work towards an application where you will see inside the application.

But for now, let's keep it simple.

So what we need to do.

First thing first we can actually just copy this entire thing and we will just copy this instead.

So let's copy this and go back inside the the code.

Let's paste this.

And now let's remove the useless comments.

And that seems pretty good.

Okay.

The next thing we need to do is we need to change the URL because we will use the board API documentation.

So this is the website.

Let's copy the website.

The website URL.

And again if you are using another API which is a good idea because this is how you really learn.

It's by like modifying the things we do and just try to do other stuff.

Anyway, I'm going out of topic.

Let's just copy this and paste it inside this little link.

Okay, so you don't need to put HTTP with the HTTP.

So we will delete this because this will automatically add it inside.

Okay.

So if you go over you will see that if you put an example the URI will return HTTP and everything.

So this will handle this stuff.

After this you will need to remove the random the path.

So we will remove this and paste it right there.

Because if you go over HTTP you will see that you require the authority which is the website board api.com.

And you need the unencoded path which is the random.

Okay.

Next if you go over you will see that you can put some query parameters.

These are some parameters by example.

You know when you go on some website I don't know if we will have some example.

No, not right now.

But sometime you go on a website and you have some sort of a question mark and some some query.

This is where you can use them.

Okay.

That's the goal.

So we will delete them because we don't need it right now.

And after this you see that we have an await.

What we do when we have an await, we create an async in order to tell flutter that we can use an await.

After this we have the HTTP.

So we need to add the package.

If we go back inside the documentation you will see that we need to.

Inside this one the example you see that we need to add the convert and the HTTP.

So let's take both of them.

Copy and paste the import inside our course page.

So you can see that we have two import.

And we use the as convert and and also the as HTTP.

This just means that in order to use it you will need to say http get.

Let me give you an example.

If I change this for HTTP.

Uh, hgh ghg AGT.

I will need to change this one also for http h g h h.

You get the point.

Okay, let's put back the HTTP.

And now let's keep going.

So we have this information.

Now what the code is doing is the response will say I will wait for the the information from the internet.

So it will get the information from this URI which is this information.

This will return some sort of information like this.

And after this we will say if the response status is a code 200 which means it was successful.

Then what we'll do is we'll say the JSON response is equal to all of this.

And all of this just means that we will use the response body, which is this information, and we will convert it inside.

We will convert it into a JSON response which we can use in order to display the information.

I know this is a lot of words, but you will understand once we play with it.

So let's first thing first just run the code.

I think if we run the code and go inside the terminal, everything should already work.

First we need to change the JSON response for the item activity by example.

So I will just replace this for activity and this should return a simple activity.

Let's go inside the application I will go navigate Next Generation next and click on this.

So if we go inside the terminal you see that.

Now we have a return number of books about HTTP null okay.

So the item count let's just verify activity.

If I have activity activity okay.

That's good.

We have the information random print number of books I will delete this and just print the item count like this Okay.

And let's try again just to see if we have some sort of value.

Get started next, next.

And we click on it.

And now you see that we have clean out your refrigerator.

And this information is coming from the internet.

And we get this information and we are able to use it inside our application.

I just think this is this is incredible and fabulous to know that whatever I'm going out of the pig.

But I just think it's it's so cool to know how to do these things.

Anyway.

So that's about it for this video.

As for now, what we have done is we have copy the code, but later in future video, we will modify everything.

You will be able to understand how it works, and we will be able to display this information inside the visual of the application.

If you already have an idea how to do this, try to do it by yourself.

Otherwise in the future.

Video.

In the next video I will show you all everything step by step.

So that's it.

See you in the next one and bye.

3. Explanation

In the previous video.

I've talked really fast about what does the code mean?

And I want to explain you a little bit more how it works.

First thing first, the convert dot JSON decode.

Okay.

What does the convert mean.

It's just because we use a has for the import.

So you need to use convert in order to use the import.

But if you want you can just remove this and it will do just the same.

Okay.

So right now because I have used the API like more than 100 times, uh, I should be declined at this moment.

But later inside this course we will be able to reuse it.

Okay, so let's just refresh the application and see get started next and say register click on it.

And now we should yeah we have it okay.

So I'm still not declined.

But later if you if you see some sort of code like 400 something it's because you have used the the API too much time in 15 minutes.

Okay.

So for now this has reset.

So that's a great thing.

So first thing first we have removed this and we don't really need it.

The reason we need to use it as sometimes is because two package or two import will be in conflict, one with each other, so if two of them have the same function, it will enter in conflict.

So that's why you need to use this.

But for now we can keep it like this.

The next thing I want to talk about is the JSON decode.

Like why do we need to use this actually.

And the reason is if we don't use the JSON decode, the response dot body is just a big string.

So it will be a long, long string with all the information inside, but everything split with some sort of maps.

So that is the reason we need to use this.

Otherwise it won't work.

So we can try and it can give you an example.

If I try to print another thing, another thing instead, I will use the log for this one.

So log developer make sure that you use the dart developer.

The log is the same thing as the print, but it will print in orange instead, so we will see the difference.

So instead of printing the item count.

So which should this which we should call this activity.

Just to make it a little bit more sense if we try to print the JSON Response dot body like this, you will see that this is a string.

So, um, yeah, we will try it right now.

Get started, click on it and click Register.

Oh I need to add a little something.

Also we will say body but we will add this thing also.

If I add this you will see right away that we cannot do this because it's a string.

The body is a string.

All right.

So let's remove this and click on it.

Now you can see that this is what is printed.

This is an entire string.

So we need to convert this information into a map of string and dynamic.

So we need to convert this into a map.

That's why we use the JSON decode.

And the JSON decode is only available because we use the import dart convert.

All right.

So now you understand a little bit more after this.

Uh I don't think we need to talk a little bit more about this because, uh, yeah, that's pretty much all we need to talk about.

Um, yeah, we have you understand this line and all those lines, uh, if anything, just put a comment inside the the video and that's it.

So I'll see you in the next video.

Bye.

4. Class

For this video.

What we will do is we will start to convert this information into something that we can use inside the phone and display right there.

All right.

So for this, we will go inside the internet and we will follow this little tutorial together step by step.

But we will use instead the board API documentation.

So let's go down and you should be able to find this section.

Create a class album.

So instead of creating this class and what I will do is I will just copy this, I will go inside ChatGPT and I will just say, you can use any personal AI.

It doesn't really matter.

It's just super fast.

So I will say, um, use this class as reference reference for the next prompt.

Okay.

Like this.

And boom.

So the the next thing will be to put this information inside and I will just say to, to GPT, I will say, um, use this to create a class and boom!

Like this.

Just by doing this, the dart, not dart, but GPT or any AI assistants will convert everything and will make it look just like what we had previously.

So factory everything should be the same thing.

Yeah, it's pretty much the same thing, but with what we want.

So we can copy the code and we can create a new class.

So we'll go inside data and inside data we can say new file.

Let's call this one activity class dot dart.

Or even we can create a folder for this.

It will make more sense because if you have more classes inside your code you will need a folder for this.

So activity inside the class in move perfect.

Now we can paste the activity inside and we have all the code that we need, which is pretty cool.

And let me explain quickly.

The code is just a simple class.

You put the information inside by using the from JSON and you just put the JSON inside.

This will convert everything and create an activity.

It will return an activity and if there is an error, it will throw a format exception error.

Okay.

Simple stuff.

So we can go inside the course page.

And now we can use this.

So let's go back inside the tutorial.

The next thing they tell us to do is to use the HTTP response.

So we are doing this already.

So await get this information.

That's good.

But now we need to say return album from JSON blah blah blah.

So we will come we will change.

We'll copy this and we will change what we have currently.

All of this, we will change this.

So maybe we can use the 200 or just paste this.

It doesn't doesn't really matter for now.

I will delete this.

I will delete this.

And we will need to modify a couple of things.

So album is not a class that we have created.

Instead it is called the activity.

I will use the activity class so I can say activity and this will import the package automatically.

You will see at the top we have the package where it is.

It is activity class.

We can delete the log package which is the developer.

We don't use it anymore.

And now we have one error.

And the error is probably because we don't return anything.

Yeah.

The void is not expected to return anything.

So what we can do instead is we can store this information inside the page.

We can do this by creating another variable.

We can say activity.

Activity.

Activity.

Just like that.

And boom.

And you can see that the activity should not be can't be null.

So what we'll do is we'll say this will be a late value.

Okay.

A late value just means that later we will set this value.

So it's okay if it's null for now.

So we will set this activity value.

Activity is equal to this.

Great.

So now we have access to this activity and the next goal.

So let me prove it to you first.

I will say print activity dot activity because it's activity is the class.

And activity is this little thing this thing.

Okay.

So we should this should return one activity.

Let's do this.

I will restart the application.

Go inside the course and click on it.

And now if we open the terminal you see play a game of monopoly.

So everything still work.

But we are using a class instead.

So now we have access to an activity which we can use inside our application to display the information.

But there will be one big problem.

The thing is, this thing will not be finished once this start.

So it will start.

It will start to await the information.

But because we are waiting and we don't, uh, we don't wait anything for the init state, it will start building the application right away.

Okay, but you can't build this application if you don't have the data.

So how can you handle this information?

For this you will need to use a future builder.

The future builder will just be a little widget that you can use, and it will wait for the data before it will just wait for the data.

In order to display this, I will show you everything in the next video.

See you in the next one.

Bye.

5. Future Builder

Okay, so how do you use the future builder?

Again, just keep in mind that if the return in your terminal is like an error 400 something, it's because you have used the limit too many times.

Okay?

So just keep this in mind.

Anyway, so now we will use the future builder.

I will cut the entire padding widget.

Let's cut this and say Future Builder.

And this widget is pretty cool.

You will see first the future we will say get data like this.

I think it was the name.

Yeah, it will not work for now because we will need to convert this into a future or we can do it right now.

Future.

But after this, inside the builder, what we'll do is we will say comment space or control space.

And this, this will have access to a snapshot.

What is a snapshot is an async snapshot.

The first thing first we will return what we had previously which was the the padding okay.

And now for the Snapchat.

Let's put the type if you want.

It's up to you.

If you don't like to put the type, you don't have to.

But I like to do it, so that's good.

After this we can use the Snapchat saying is this loading?

If it's loading then display a loading widget.

And if it's finished, if the data is finished to load then give me the data.

All right.

So what we'll do is we'll say if and there is a couple of people that don't like to put many returns.

So I will try to just put one return.

But uh yeah.

So we'll create a widget.

Widget like this.

Oops.

Widget widget.

Uh okay.

Never mind.

We will do this later.

I don't want to confuse you right now okay.

So I will say if the Snapchat dot connection state is equal to connection state dot waiting.

So if we are still waiting for the data, then I want to return the circular progress indicator.

This widget is just, um, a loading widget.

So you will see it's loading on the screen.

After this, we can say if the snapshot connection state that has not connection state but has data, uh, we can say error maybe or has that it's pretty much the same thing.

So if it has data then we can display the data.

We can say return the padding, this entire thing.

And finally what we can say is we can say, uh, else if all of this doesn't work, then return a center widget which will have inside the child a text widget saying, uh, error.

Okay.

You can modify this to make it look better, but for now it's just simple stuff.

Okay.

So now you will see that we have error.

And the reason is the, um, the getdata isn't returning anything for now.

So if we go back inside, you will see that we still have the error.

And this is because the the getdata is not returning nothing.

So that's the problem.

And another thing we need to fix later is you see how many return we have.

I know some people like to put only one return inside their function, and I don't think it's a bad way to handle things.

So we will change this later in a future video.

So first thing first, let's fix the getdata in order to see something.

At least for this, we will need to.

Instead of define an activity, we will return the activity and the return activity.

Now we can just delete this first and we can remove the print.

And that should be good.

So now what is this.

Not now.

So if we save you see that now it's working and we don't have the error.

So now flutter understand that we have data.

But how do we use this data.

So what we'll do is we will say snapshot dot dot data.

Now we have access to the data.

And we can say um this is an activity.

So let's say Activity.

Activity.

Activity is equal to this.

Great.

So this should still work if I hot reload.

Yeah.

Everything still works.

And now we can use this information.

So instead of displaying course we will display activity dot activity.

Oops.

Activity.

And now let's save.

And now you see that we have the activity.

It's not really visible right now.

But if we go closer we can see, um listen to something new, music, blah blah blah.

I will just make this a little bit more visible so you can see better.

I will put another text under the widget saying exactly the same thing like this.

I will put all the commas and save.

Now you see.

Create a meal plan for the coming week.

So it is working.

And just the fact that we have built a future builder, that we request the information from the internet, and we display the information into the application by using a class.

I think it's just super cool.

So that's about it for this one.

Yeah.

Another thing I wanted to show you was how to remove the multiple returns.

Because some people don't like this.

And we can do this in a couple of seconds.

So what I will do is I will say widget.

Widget.

And the only thing we'll do is we will go through all the if condition.

And at the end we'll say return the widget.

Okay.

And we will remove all returns and replace them for.

So let me select return.

I will select all of them the three of them.

And I will say widget is equal to this information.

So now each of them will be defined.

So if the Snapchat connection is waiting then it will define this as the widget.

If the Snapchat has data then it will define the widget as this.

And otherwise it will define the widget as center and then display this.

So now if we just reload, you see that everything is still working.

But we only have one return statement which some people prefer this one one return widget.

So that's about it.

And I'll see you in the next video.

Bye bye.

6. Personal AI

All right.

So now we need to talk about one thing.

And then I want to show you one other tips and tricks that you can use in order to code like ten times faster.

So the first thing is if you are still going through the tutorial, you will see that later this thing will tell you like okay, you can display it with a futurebuilder like we did, but they do it in a different way.

They used the builder and they say if there is data then display the data.

Else if if there is like an error, then display the error and otherwise display the circular progress indicator.

So there are many ways to to code with flutter.

So this is another uh, another way to do it.

If you prefer this one you can take this one.

It's up to you after this.

Another thing I want to show you is how to use ChatGPT or any AI.

It doesn't really matter in order to generate all of this code without even, like wasting time.

So what we will do is we will first say, um, we have this information.

Um, yeah, we will just go in ChatGPT right now and we'll say, okay, based on this information,

I want a flutter page code.

I want a flutter code to display this, uh, this API.

That's it.

And maybe we will need to put some sort of information.

We will need to put this.

And let's scroll down to see what flutter gave us.

Okay.

That's good.

Uh, but now we'll need to change the URL.

So, um, the URL is, is this boom.

Okay.

So now, uh, you will be able to just copy paste really fast.

So this already created the class.

After this we have a main you will need.

You will not need to copy this, but it will create a stateless widget for you which you can just copy

and you will save an incredible, incredible amount of time just by doing this.

So let's wait for this to finish and I will copy all the code inside this page, and I will show you

how fast we can code with I like this.

So let's copy the code.

Let's go inside the course page and I will replace this boom and paste everything.

Next we will need to modify a couple of things.

First thing first the URL launcher.

Um we don't really need this after this.

And yeah, you have to keep in mind that with, uh, personal AI and stuff like this, you will need

to tweak a couple of things.

So it's not perfect yet, but it will save you a lot of time.

So we can delete all of this, delete this.

Also remove this and just keep the the activity screen.

So we will change the name of activity screen for course page.

Oops.

Course page.

And that's good.

And now the activity we need to import quick fix and import the activity that we have inside the classes that we have created earlier.

And next you will see some errors.

So you can say quick fix and say um yeah.

The error is just that we have some sort of a line before.

So we will remove all those lines.

I will select all of them.

This one, this one and this one and delete them.

Now it's better.

And another thing is saying constructor for public widget should have name key.

So we can fix this by going over quick fix and add key constructor.

Okay.

So you will see that personally I uh are not really like perfect yet, but if you have one that is better than use this one.

That's it.

Next, this is all the page that is created for us.

So we haven't code that much.

We can remove the launch URL.

We don't need this sometime.

They they do a little bit too much, but it doesn't matter.

We can remove the elevated button and boom!

This entire code is like generated by by GPT.

And I just think it's so cool.

And right now we have an error obviously.

So I will restart the application.

I will go inside get started next and click on Register and click on this.

And now this is loading and boom display everything.

Isn't it like absolutely crazy.

You click on Fetch data and it will fetch another one.

So you can you can really code your entire application by using a personal AI.

And just if you are able to plug and play the things, then you are like a machine.

You you are unstoppable.

Anyway, I will stop with that.

I'm going to another topic.

It's about like AI and stuff like this.

But I just think it's crazy how the the future is going.

Uh, I will, I will stop right there.

That's it.

See you in the next one and.

Bye.

12_next-steps

1. Let's talk about State Management

This course is coming to an end.

So you have done all the hard work and now it's time to fly by yourself.

Okay.

So I want to talk about a couple of things.

Just so you get to the right track for your own path, okay.

And I want to show you like the mistake I've done.

So you don't have to do them.

You just you just go out and start building apps.

The first thing to talk about is about state management.

Okay.

You might learn you might, uh, heard a lot of people talk about this.

Like, you need to use this.

You need to use provider.

You need to use river pod.

Let's be honest.

We don't care.

Okay.

We don't really care.

Inside this course, I've talked about the value notifier.

And if you release an MVP application or just a simple app, you can use this.

If your job is requiring you to use the river pod or provider, you can learn this.

It's super simple okay.

But don't waste too much time.

If it's too complex and you don't like this one, try another one and at some point you will have one that you kind of like, stay with it and it just doesn't really matter.

Okay.

Don't waste too much time.

Like learning a lot of things for nothing.

Just just build apps.

Okay.

Anyway, you will be able to do pretty much everything with the value net fire like we have talked about in this course.

But if you want to use a state management and you just want my opinion, I think River pod is pretty simple.

River pod is like the newest version of the provider, so why not use the newest version?

And after this, uh, another one, which is pretty cool, is called The Block.

I don't see it right there, but, um, the block is pretty cool, but it's a little bit complex, so it's complex.

And also you have a lot of boilerplate code.

So if you don't want to to go inside this route, just try with River pod.

It's pretty simple stuff.

And otherwise just keep the simple value notifier and everything should be fine.

So that's it.

I just wanted to talk about this.

Don't waste too much time on state management.

Start building apps.

That's it.

See you in the next one.

Bye.

2. Internationalizing

The next thing is about how to create an application that have multiple language.

If you want English, French and other languages, then you you need to use like internet internationalization,

flutter apps.

Pretty, pretty -- hard to say this word, but anyway, you need to use this so you can find this information online and just follow the tutorial.

We will not talk about it inside this course because this is like, uh, I don't know, it's not really useful when you create your first application.

So that's why it's just I just want to give you a little things that at least you will have in the back

of your hand, at the back of your head if you need them later inside your flutter journey.

So at least you will know that you just need to use this flutter localization and just follow this simple tutorial and you will be able to use English, Spanish, and other languages inside your app.

Uh, how you do this, you just need to create some sort of dictionary of all the words in English and all the words in French, and then the application will define if it's currently in French or English and stuff like that.

So if you need to have multiple language, uh, you can redirect to this information.

That's it.

I'll see you in the next one.

Bye.

3. Let's talk about Firebase

Now let's talk about Firebase.

Okay.

If you don't know or you don't have any server in order to authenticate user or create a database or stuff like this, you can use Firebase.

Firebase is pretty simple to connect with flutter because both of them are built by Google.

So pretty simple stuff.

Um, we will not talk about this inside this course because this could this is a course in its own.

Okay.

It's really a lot of things you can do with this.

For example, you can authenticate your user if you want.

You can create some sort of database, uh, live database or machine learning anything.

You can create a lot of things with this.

If you have your own server or you prefer your own server, keep your own server and use like HTTP requests and stuff like that.

But Firebase is a good way to authenticate your user when you start.

This is what I use.

Uh, very, very simple to do if you want.

I have some tutorial on YouTube and you can find them.

So you will be able to use Firebase authenticate user and use a database.

So that's about it.

And um yeah that's about it with Firebase if you need authenticate user and a database that's it.

See you in the next one.

Bye.

4. Publish

The next thing we want to talk about is this, um, how to publish an application on the App Store or the Android Store or the Play Store?

For this, you will need to follow this little tutorial on flutter dev.

Uh, how to release application change quite often so I cannot really do a video.

That will be because it will be outdated in like, uh, three months.

It's crazy.

So the only thing you have to do is go on this website, click on deployment, build on build and release on Android app, and build and release on iOS app.

And this will tell you everything you have to do in order to publish your application.

You will see this.

Uh, this is a lot of pain in the --.

Just keep it.

Keep this in mind.

Uh, it's pretty annoying to do, but once you know, and once once you follow the tutorial, uh, everything should be fine.

Great.

So that's about it for how to publish your application.

And in the next video, I want to show you a little bit how to create some sort of animation with your application.

So we will talk about this in the next one.

See you in the next video.

Bye.

5. Animations

Now I want to talk about animations.

So if you want to learn more about ammunition first thing first you can always go on YouTube.

I have a video about every single widget with flutter, so you will find even all the animation available with flutter and you can pick the one you like.

But I also want to show you one inside this video.

Just why not, right?

Because it's pretty simple to use and it can make a lot of difference inside your application to make it look a lot more cooler.

So what I will do is first we'll go inside, we'll wrap the entire column.

So I will cut this and I will use the animated crossfade.

So this animation will animate a transition between two widget.

And in order to change widget we will need to have some sort of button to trigger this animation.

So the first child is just what the application will display.

If the value is true and if it's false, then it will display another widget.

So the first widget we will paste is what we had previously the column.

And the second child.

What we'll do is I will just put a center widget and inside a child and inside I will say image that oops, I will say image dot asset and I will use the path of the image.

We think it was asset slash image image slash.

Uh I think it was background jpg.

And yeah so the crossfade state, if you go over you will see that you need to use a crossfade state widget.

And we'll say crossfade state dot show first for now okay.

We will modify this later.

After this the duration if you go over you need to use a duration widget.

So we'll say duration duration.

And the duration will be in uh milli.

Uh yeah millisecond.

And it will be 300 milliseconds.

It's a good it's a good uh it's a good number for animations usually.

So if I save you see that we still have the first widget which is the column.

But now we need to change this for the second widget to show second.

Now you have seen the little animation.

If I do it again and put the first and save.

You see how you have some sort of animation?

Okay, this doesn't look that much great.

But if you have two widgets that look alike, for example, if you have another center widget which I will show you, I will delete this the column, I will paste the center widget and I will use the other image which was icon dot png if I remember.

And no, it was not that.

So let's go inside asset image.

It was Logo.png Logo.png and save.

Now you have this.

And now let's change for show us show second and save.

You see the little animation the transition.

This is what I mean.

So we will change this for 500.

And in order to use it what you will need to do is first thing first I will put back the the padding the column.

And I will change this for 1000.

So it will be very slow.

What you need to do is to create some sort of buttons that when you press on it, it change.

So you will go for example, in the app bar.

In the app bar you should have an argument called actions and actions is just right there.

All the buttons right there.

I will use a Nikon button and this one will have a function empty for now, and a Nikon which have icons that let's let's say switch switch, axis shortcut.

It doesn't really matter.

For now.

I just want to show you how to use the the animation.

So the this will use a.

I don't think we need to use a set state for the animation.

Let's try it without.

First we will say we will create a variable.

So we will say right there or right there.

Boolean is first is equal to true.

And then we can change this value.

Inside the icon button we can say is first is equal to the invert of is first.

And that's it.

So now when we press on this we should see the animation and it doesn't work.

So I will refresh the app click Get Started.

Next register click on it and let's try again.

You see that this doesn't work.

So I think.

Yeah.

And the reason is we don't even use this information inside the inside the crossfade state.

We need to use this.

We need to say if it's first then crossfade state dot show first.

Otherwise if it's not first then we will say crossfade state dot show second.

And now it should work.

So every time you press the button now it should translate and convert into the other one.

So let me just reload and do it again.

And now it doesn't work because we don't use the set state.

So yes we need to use the set state.

I would say set state and put this information inside.

And now let's save.

I will just hot reload and try it again.

And now you see that it is now working.

We have some sort of animation.

It's not the best animation you will see in the world, but at least you know how to use them.

And if you want more animation, I have an animation.

I have a video that explains every single animation on YouTube, so you can go there and look at it.

So that's about it and I'll see you in the next video.

Bye.

6. Recap

First thing first, I want to personally congratulate you because you have finished the course.

Not a lot of people finish things and you are one of them, which is one of the best quality you can have.

I I've heard once, um, lazy people don't start after this.

It was, um, loser don't finish.

And at least you know you are not a loser because you have finish.

That's great.

And third, legend never stops.

And this has, uh, this has a big impact in my life, so hopefully it has with you also.

But yeah, congratulations for finishing this course.

Um, not a lot of people do this.

After this, I want to show you a little bit what we have done inside this course.

So you can look back and see.

Oh, --.

That's that's the big picture.

We have learned all of this.

And now I'm able to, like, build application with flutter by myself.

Because with all this structure, with all this framework, you are able now to build application by yourself.

You have all the tools available.

Okay.

Let's talk about them.

We have talked a little bit about the key concept like a widget inside widget, stuff like this.

Uh, a widget is a capital letter, and our argument is a small letter and stuff like this.

We have talked about the basic layout, like all the basic widget, column row, container, size box and stuff like this.

Okay.

After this we went into master layout, more like the scaffold, the material app and stuff like this.

If you understand what I am talking about right now, it's because you understand flutter.

Okay.

You have learned things that you didn't know before, and that's that's that's great.

Um, next, the two step formula to debug, uh, to fix your bug.

That was pretty simple.

Uh, the stateful stateless widget, if you want to refresh the screen or not, the user input, how the user can interact with the application buttons, text fields and things like that.

Gesture detector.

Next we have talked about basic routing.

So how to navigate inside your application.

So I think this should be right there instead navigator.

And this should be right there.

Yep.

My mistake.

So let's save.

And after this we have talked about the clean UI.

So how to create an application that looks a little bit better.

For example you could you could use the widget clip or rect.

Or you can use the example circle avatar stuff like this.

Just widget that are not the layout but make the application look good.

And next we went into package management.

So how to use package in your application to create something better.

For example I think it was a which which package have we talked about.

If we go inside the Pubspec.yaml we have talked about the Lottie package, how to add animations, visual animations, which is pretty cool.

Shared preference how to save data locally on your phone.

This is super important to know.

So we have talked about this and how to add the flutter launcher icon HTTP.

But there are so many dependencies or packages that you can use inside flutter.

We have just talked about like the most important ones.

So let's go back inside our little thing.

It's not inside this.

It's inside the other internet.

After this, um, we have talked about the responsive UI.

So how to create an application that is responsive either for an iPad or a mac or anything?

That's pretty cool.

And for this you use usually the layout builder and stuff like this.

Next we went into networking in the previous module and networking was the HTTP request.

How to get the data from an API.

And with this you can you can build any application.

You have all the tools to build any application.

You can get the data from the internet.

You can get the data from a server.

You can create an authentication with Firebase.

We haven't talked about Firebase yet, but later inside this video I will show you the the step you need to take.

If you want to go this route it's up to you.

But with the networking you are able to do everything with the internet.

And if you you merge this with state uh, with package management which we were using, the shared preference.

If you merge this with this, you can save the data into the phone also.

So this is really powerful.

And now right now we are inside this module.

Um, are you ready to fly high?

Which is just like the next step you can take in order to make your journey even, like bigger.

So that's about it.

With this, you are now able to build your mobile app, which is really great.

So if you want to go to the Firebase route, it's up to you.

If you have another server that you prefer to authenticate user and have a database, use your own.

Okay, I will not force you.

I will not, uh, force you to do anything.

So, uh, I've started with Firebase.

I think it's pretty simple when you start and I think it's it's cool.

So what you will need to learn is how to set up Firebase, how to authenticate your user, how to create a database and how to create a real time database and data management okay.

By example, how do you manage the fact that you receive the data from the internet?

And how do you save this on the phone?

And how when you start the application, where do you get your information about?

Do you get it from the internet first or in the phone?

So many questions that we can answer.

Maybe inside another course about Firebase.

I'm not sure if I will do this course any time soon or anytime in the in the life.

Because yeah.

Anyway, um, that's about it.

So again big big big congratulations for you.

You have finished your course and that's great.

Thank you for watching.

Thank you for taking this course.

Thank you for taking the time to learn flutter and to learn with this course.

You could have learned with any course.

And you, you choose this one.

Um.

And my, my English is terrible.

I won't lie, my English is terrible.

And you, you have still take this course.

Hopefully.

It's because it's simple to understand.

And I just want to say thank you for that.

Thank you for taking the time to learn everything and taking this course.

And congratulations.

See you on another video, maybe on YouTube or see you somewhere else.

You can always send me a message pretty much everywhere online and I should be able to answer you.

Thank you very much.

Bye.