

Flutter Fundamentals Routing



Peter Kassenaar –
info@kassenaar.com



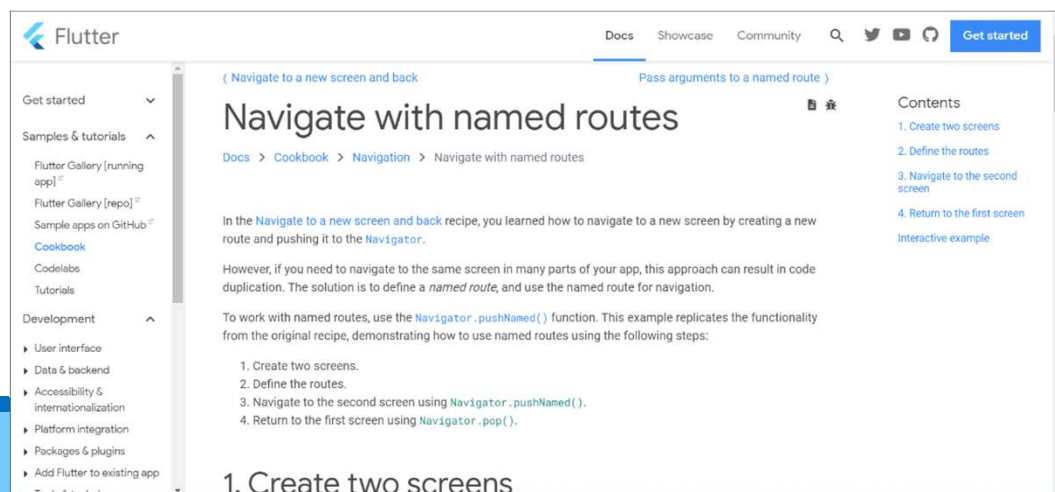
Adding Routing

Switching between screens in your application

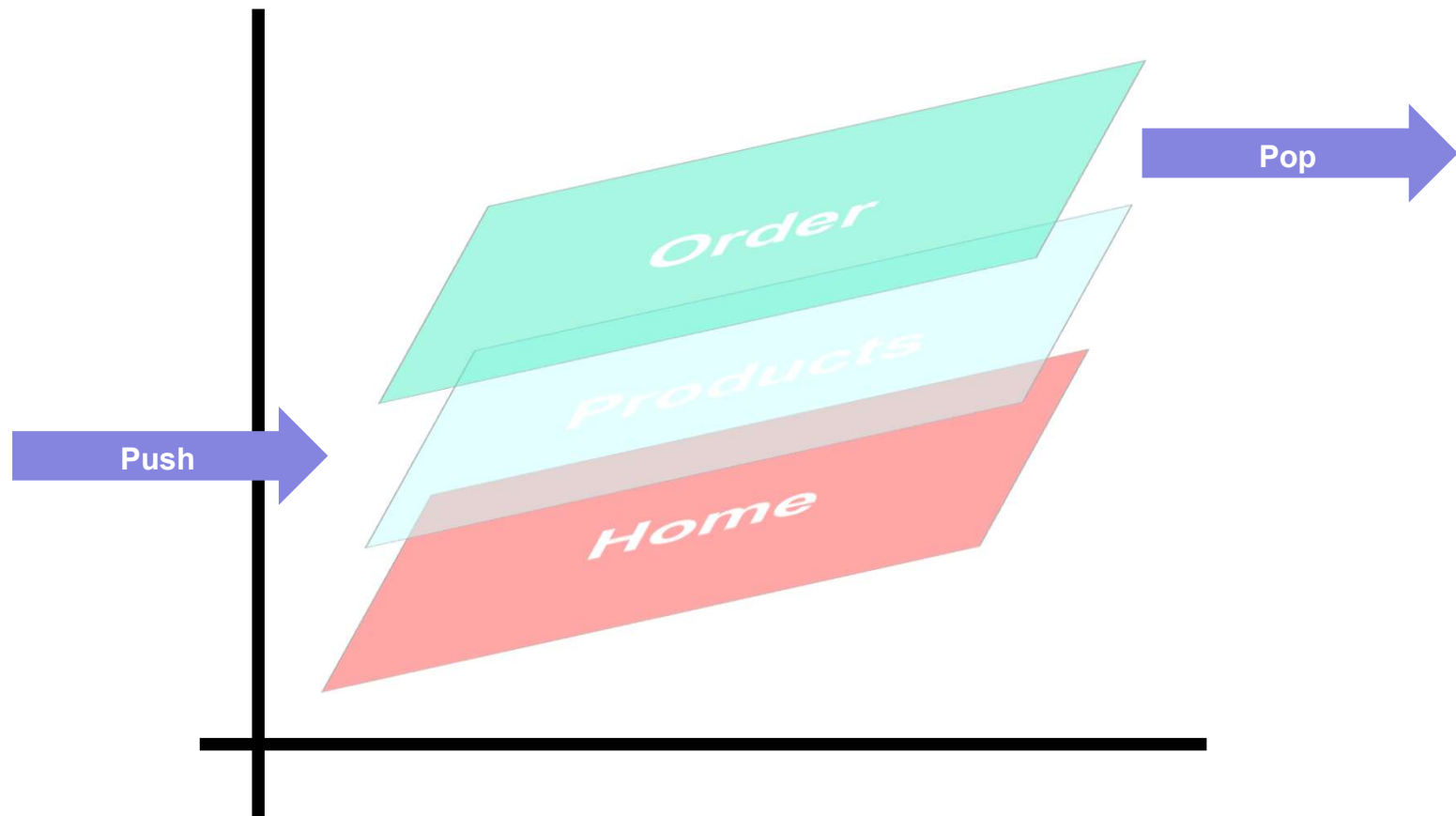
Adding routing



- Routing in Flutter is **different** from routing in web apps
- You **stack screens on top of each other** while routing
- Use the `Navigator` object to push new (named) routes to the stack
- <https://flutter.dev/docs/cookbook/navigation/named-routes>



Routing stack





Adding Routing

- The `MaterialApp()` widget has *built-in routing*
- Use `initialRoute` to set the start screen
 - then don't use the `home:` property!
- Use `routes:` to define a `Map` with routes
- A Dart `Map` is like a JavaScript object
 - key/value-pairs
- Use `Navigator.pushNamed()` to push a new screen to the stack

AppBar()



- When using an `AppBar()` widget, it shows a `Back-button` automatically
- Use `Navigator.pop()` to navigate from code to the previous screen
- Always provide `context` as an argument, to let Flutter know 'where you are in the app'.

Yeah. It's magic.

Steps. 1: Create additional screens



- Create a screen showing details of the country
 - Name: `screens/country_detail.dart`
 - For now it's a simple `Stateful` widget
 - We'll pass arguments with the specific country next

```
import 'package:flutter/material.dart';

class CountryDetail extends StatefulWidget {
  @override
  _CountryDetailState createState() => _CountryDetailState();
}

class _CountryDetailState extends State<CountryDetail> {
  // getting the country details later...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Country details'),
        ...,
      ),
      body: Column(...),
    );
  }
}
```

2. Build the routing table



```
import 'package:flutter/material.dart';

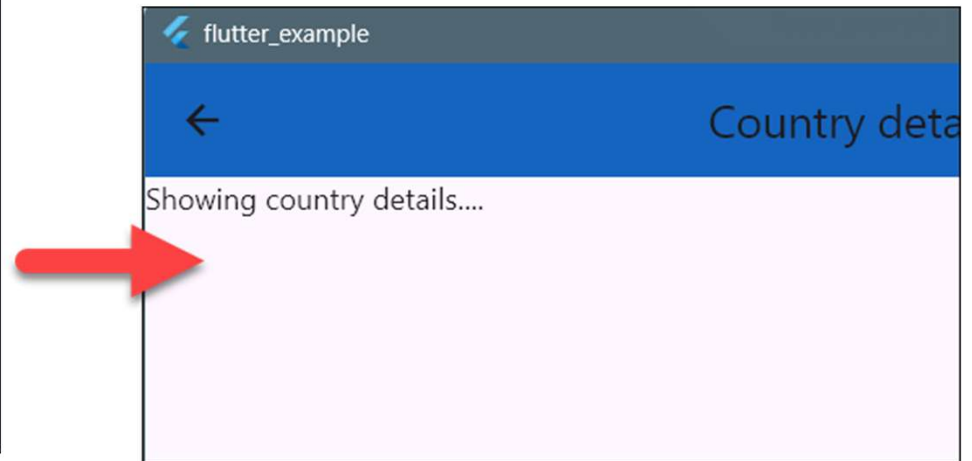
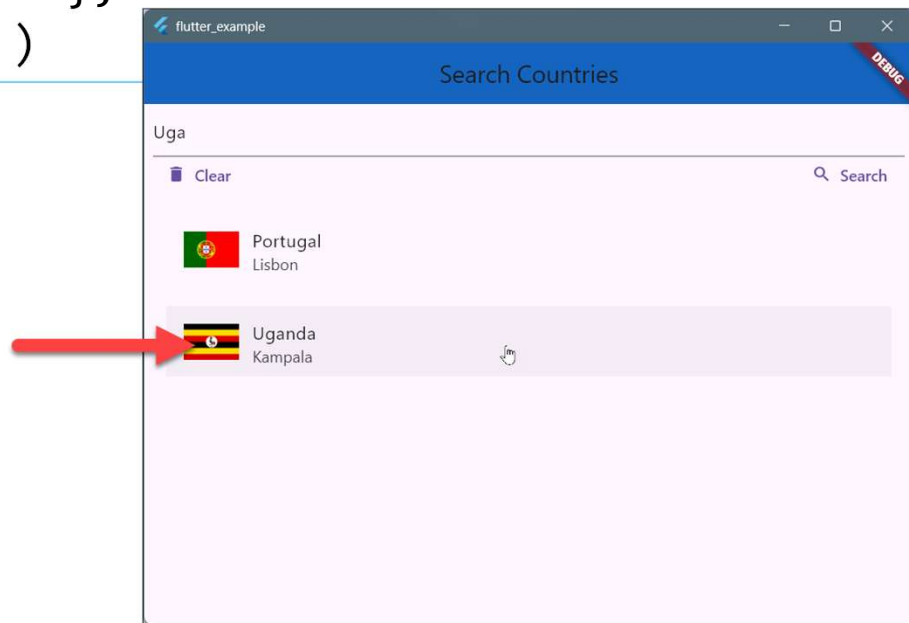
// Import the screens
import 'package:first_flutter_app/screens/homeCountries.dart';
import 'package:first_flutter_app/screens/countryDetail.dart';

void main() => runApp(MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => CountriesHome(),
    '/detail': (context) => CountryDetail(),
  },
));
```


3. Navigating to detail route



```
ListTile(  
  ...  
  onTap: () {  
    Navigator.pushNamed(context, '/detail');  
  },  
)
```





4. Sending arguments/parameters

- Use the `arguments:` property
- Send (in this case) the `name` of the country
- In the detail route: do an `additional http-request` to lookup the country, based on the name

```
onTap: () {  
  // navigate to detail page  
  Navigator.pushNamed(context, '/detail',  
    arguments: {'name': countries[index]['name']['common']});  
},
```



5. Retrieving parameters

- In the detail component
 - **Import** `http, convert` package
 - Create **variables** to hold the routing data & found country
 - **Get** the `countryData` in the build function
 - Create **additional UI** to show the found country

Getting the country



```
// 0. Variables in this widget
String url = 'https://restcountries.com/v3.1/name/';
Map routingData = {};
String countryName = '';
List countries = [];

// 1. Get the requested country
void getCountry(name) async { ...}

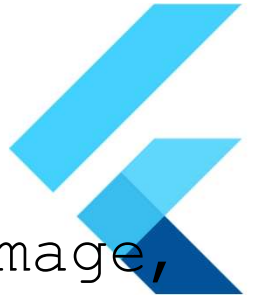
//***** UI
@override
Widget build(BuildContext context) {

  // Get the parameters from the router
  routingData = ModalRoute.of(context)?.settings.arguments as ...;

  // Check if there already is a country. If not, fetch it.
  countries.length == 0 ? getCountry(routingData['name']) : null;
  ...
}
```



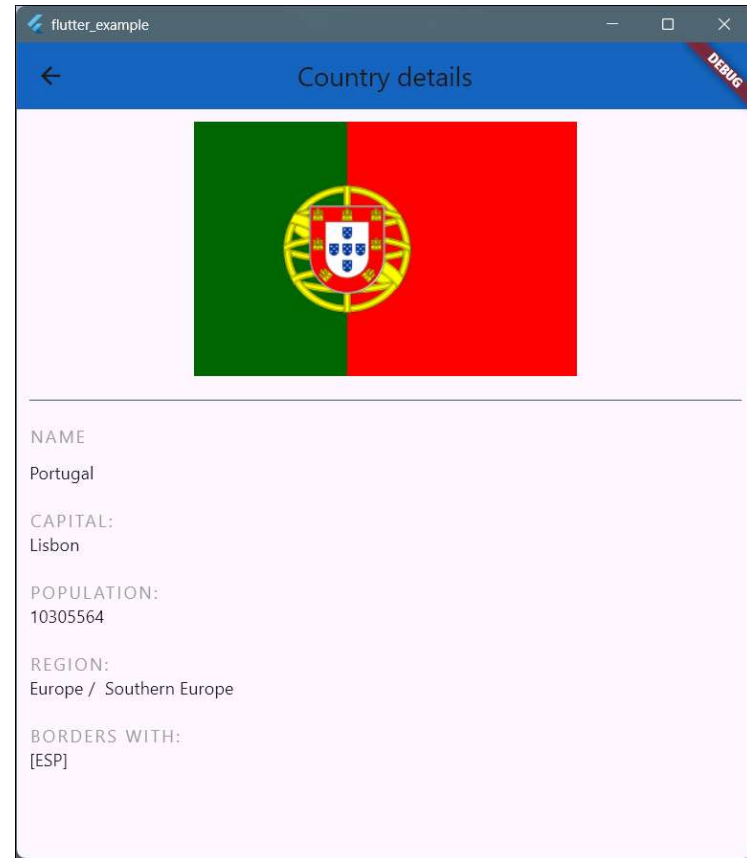
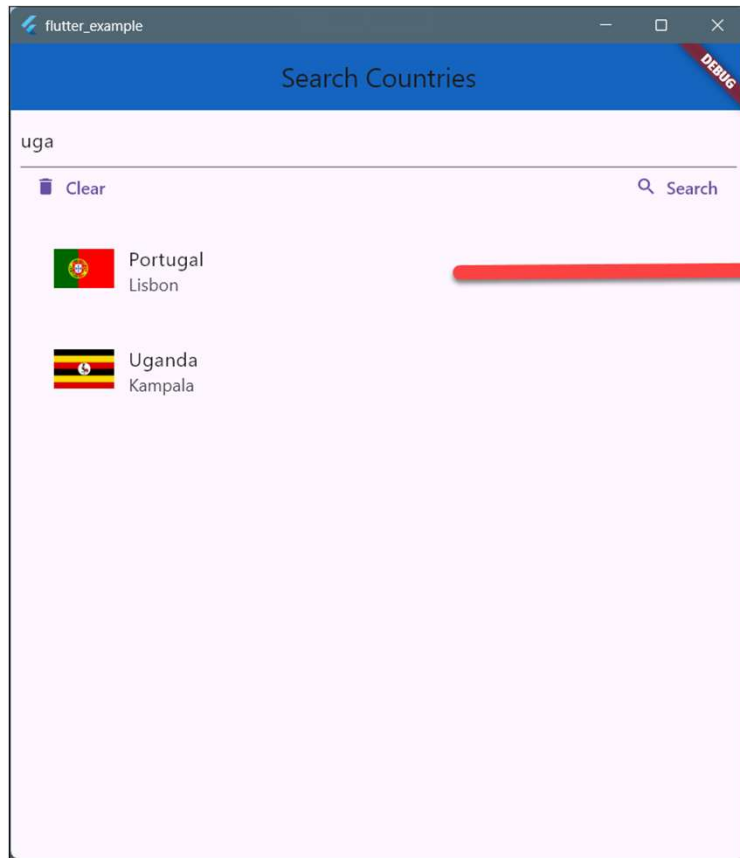
Create UI



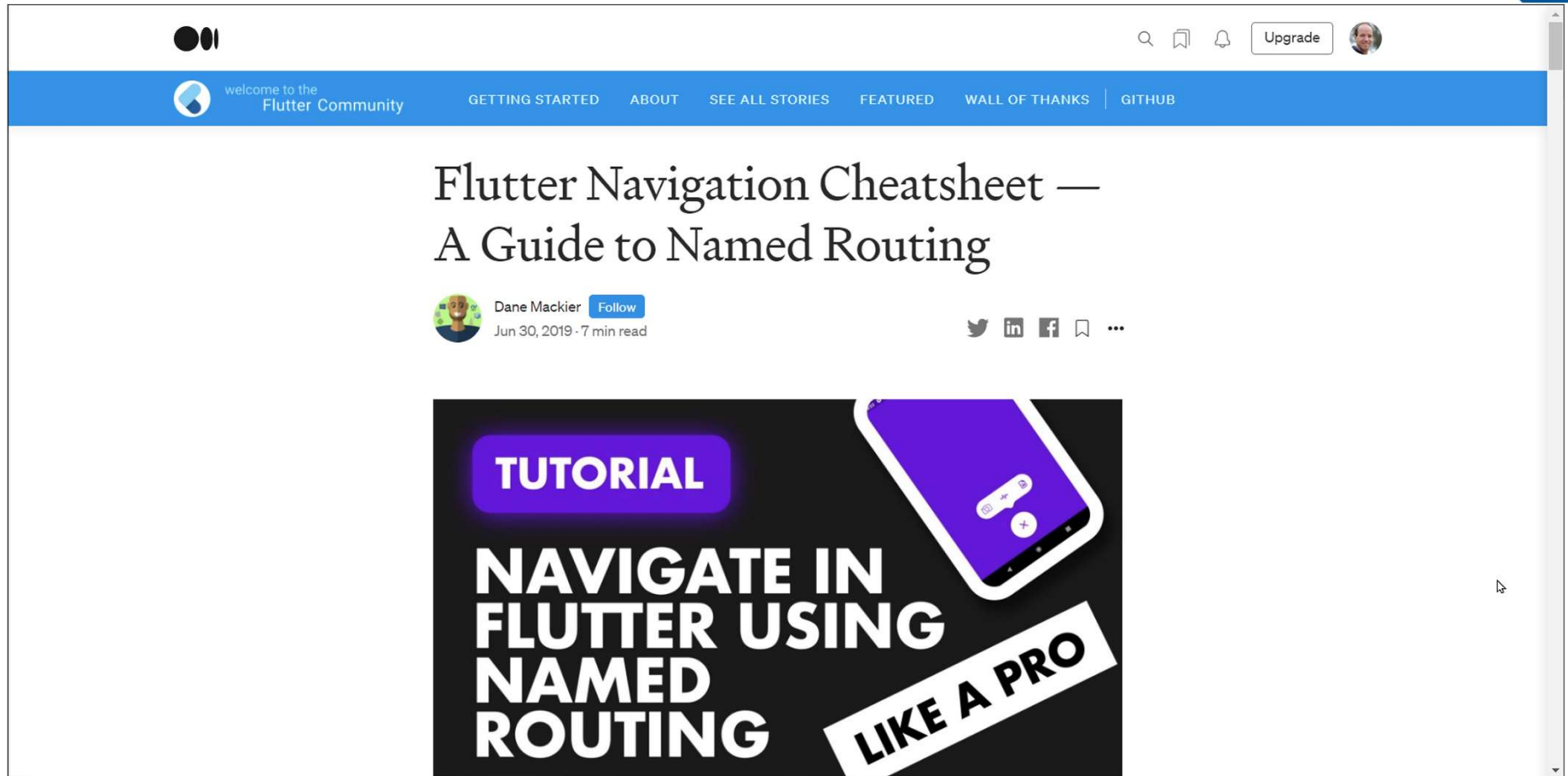
Use widgets of your choice (`ListView`, `Column`, `Image`, `Text`, `Sizedbox`, etc) to build the UI.

```
body: ListView.builder(  
  itemCount: countries.length,  
  itemBuilder: (BuildContext context, int index) {  
    return Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: <Widget>[  
        Center(  
          child: SvgPicture.network(  
            countries[index]['flags']['png'],  
          ),  
        ),  
        ...  
        Text(  
          'NAME:',  
          style: TextStyle(color: Colors.grey, letterSpacing: 2),  
        ),  
        ...  
      ],  
    );  
  },  
);
```

Result



More information on routing



<https://medium.com/flutter-community/flutter-navigation-cheatsheet-a-guide-to-named-routing-dc642702b98c>

Navigation cookbook

A screenshot of the Flutter Docs website. The left sidebar contains a list of navigation items: Introduction, Intro to Dart, Widgets, Layout, State management, Handling user input, Networking and data, Local data and caching, From another platform?, Codelabs, Cookbook (highlighted with a red box and an arrow pointing to the main content), Demos and samples, Stay up to date, App solutions, User interface, and Introduction. The main content area shows a list of topics under the 'Navigation' section, which is also highlighted with a red rounded rectangle. The topics include: Create a grid list, Create a horizontal list, Create lists with different types of items, Place a floating app bar above a list, Use lists, Work with long lists, Create a list with spaced items, Maintenance (with a sub-item: Report errors to a service), and Navigation (with sub-items: Animate a widget across screens, Navigate to a new screen and back, Navigate with named routes, Pass arguments to a named route, Set up app links for Android, Set up universal links for iOS, Return data from a screen, and Send data to a new screen).

Flutter Docs

Homepage Community Packages

- Introduction
- Intro to Dart
- Widgets
- Layout
- State management
- Handling user input
- Networking and data
- Local data and caching
- From another platform? ▾
- Codelabs
- Cookbook**
- Demos and samples
- Stay up to date ▾
- App solutions ▾
- User interface
- Introduction

- Create a grid list
- Create a horizontal list
- Create lists with different types of items
- Place a floating app bar above a list
- Use lists
- Work with long lists
- Create a list with spaced items

Maintenance

- Report errors to a service

Navigation

- Animate a widget across screens
- Navigate to a new screen and back
- Navigate with named routes
- Pass arguments to a named route
- Set up app links for Android
- Set up universal links for iOS
- Return data from a screen
- Send data to a new screen

<https://flutter.dev/docs/cookbook/navigation>

Workshop



- **Continue** with the typicode (users) API from the previous workshop
- Implement routing:
 - show a list of users on the homepage
 - When clicked on a user, show its details in the next page
- **Optional:** create a button to navigate back to the homepage (instead of using the built-in `AppBar()`)
- **Optional:** implement a `TabBar()` that a user can navigate from
 - <https://api.flutter.dev/flutter/material/TabBarView-class.html>
- examples: `../_300-routing`, `../_310-routing-detail`

