Βιοϊατρική Πληροφορική

Λευτέρης Κουμάκης,
Μαρία Χατζημηνά

Εργαστήριο 5

Futures

- Υπάρχουν δύο τρόποι για να εκτελέσετε ένα “Future" και να χρησιμοποιήσετε την τιμή που επιστρέφει (Εάν επιστρέψει οτιδήποτε). Ο πιο συνηθισμένος τρόπος είναι να περιμένουμε το “Future" να επιστρέψει δεδομένα. Για να λειτουργήσει αυτό η συνάρτηση που καλεί τον κώδικα πρέπει να επισημανθεί με “async".
  - Παράδειγμα:

```
FlatButton(
  child: Text('Run Future'),
  onPressed: () async {
    var value = await myTypedFuture();
  },
)
```

- Ο άλλος τρόπος για να χειριστείτε ένα “future” είναι να χρησιμοποιήσετε τη συνάρτηση .then() . Παίρνει μια συνάρτηση που θα κληθεί με τον τύπο τιμής του future.
  - Παράδειγμα:

```
void runMyFuture() {
  myTypedFuture().then((value) {
    // Run the code here using the value
  });
}
```

Forms

- GlobalKey<FormState>: Δημιουργούμε ένα global key το οποίο χαρακτηρίζει μοναδικά την φόρμα και επιτρέπει να την κάνουμε validate.
- TextFormField
  - validator: Συνάρτηση που ελέγχει την τιμή του πεδίου.
  - controller: Χρησιμοποιείται για να μπορούμε να διαχειριστούμε τις τιμές εκτός φόρμας
  - Η αρχική τιμή του πεδίου μπορεί να δοθεί όταν δημιουργείται ο controller.
  - initialValue: δηλώνουμε αρχική τιμή του πεδίου
    - Δεν μπορεί να χρησιμοποιηθεί ταυτόχρονα με τον controller.
  - onTap: μπορεί να χρησιμοποιηθεί όταν θέλουμε να προσθέσουμε κάποια ενέργεια όταν πατιέται το πεδίο (όπως το να εμφανίσουμε ένα datepicker)
  - keyboardType: τύπος πληκτρολογίου που εμφανίζεται

```dart
final _formKey = GlobalKey<FormState>();

TextEditingController textController = TextEditingController(text:"Some text");
TextEditingController emailController = TextEditingController(); //για να μπορου

@override
Widget build(BuildContext context) {
  // Build a Form widget using the _formKey created above.
  return Form(
    key: _formKey,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        TextFormField(
          controller: textController, //χρησιμοποιουμε τον controller ωστε να μπ
          //initialValue: "Some text", το initial value δεν μπορει να χρησιμοποι
          // The validator receives the text that the user has entered.
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter some text';
            }
            return null;
          },
        ), // TextFormField

        TextFormField(
          controller: emailController,
          // The validator receives the text that the user has entered.
          validator: (value) {
            if (value == null || !value.contains("@")) { //ελεγχουμε αν υπαρχει
              return 'Please enter a valid email';
            }
            return null;
          },
```

Forms

- Όταν καλείται η validate() τρέχει τις συναρτήσεις για κάθε validator: των πεδίων. Αν δεν υπάρχουν λάθη επιστρέφει true. Διαφορετικά τυπώνει τα μηνύματα λάθους και επιστρέφει false.

```
Padding(
  padding: const EdgeInsets.symmetric(vertical: 16.0),
  child: ElevatedButton(
    onPressed: () {
      // Validate returns true if the form is valid, or fals
      if (_formKey.currentState!.validate()) {
        print(emailController.text);  //απλα τυπωνουμε τις τ
        print(textController.text);
        // Αν η φορμα ειναι valid τυπωσε ενα snackbar (στο κ
        // συνηθως εδω σε πραγματικες εφαρμογες σωνονται οι
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Processing Data')),
        );
      }
    },
    child: const Text('Submit'),
  ), // ElevatedButton
```

Εργαστηριακή Άσκηση - Form Validation (20 λεπτά)

- Task 1: Προσθήκη πεδίου Username
  - Προσθέστε ένα νέο TextFormField για το όνομα χρήστη. Το πεδίο δεν πρέπει να είναι κενό.
- Task 2: Προσθήκη πεδίου Password
  - Προσθέστε ένα πεδίο κωδικού. Ο κωδικός πρέπει να έχει τουλάχιστον 6 χαρακτήρες.
- Task 3: Προσθήκη πεδίου Confirm Password
  - Προσθέστε πεδίο επιβεβαίωσης κωδικού. Η τιμή πρέπει να είναι ίδια με το password.
- Task 4: Βελτίωση Email Validation
  - Ελέγξτε αν το email έχει σωστή μορφή, π.χ. περιέχει @ και τελεία.

Exercise - Form Validation

TASK 1

```
if (value == null || value.isEmpty) {
  return 'Please enter username';
}
```

TASK 2

```
if (value!.length < 6) {
  return 'Password too short';
}
```

TASK 4

```
if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value!)) {
  return 'Enter valid email';
}
```

TASK 3

```
final passwordController = TextEditingController();

TextFormField(
  controller: passwordController,
  obscureText: true,
  decoration: InputDecoration(labelText: 'Password'),
)

TextFormField(
  obscureText: true,
  decoration: InputDecoration(labelText: 'Confirm Password'),
  validator: (value) {
    if (value != passwordController.text) {
      return 'Passwords do not match';
    }
    return null;
  },
)
```

Constructors

- Warning: Use key in widget constructors.dart
- A Key is an identifier for Widgets, Elements
- Όταν ξανασχεδιάζεται το widget tree το flutter πρέπει να γνωρίζει:
  - Ποια widgets είναι τελείως καινούρια και ποια έχουν ήδη state και σε αυτό βοηθάει το key
- Δεν είναι υποχρεωτικό αλλά θεωρείται καλή πρακτική

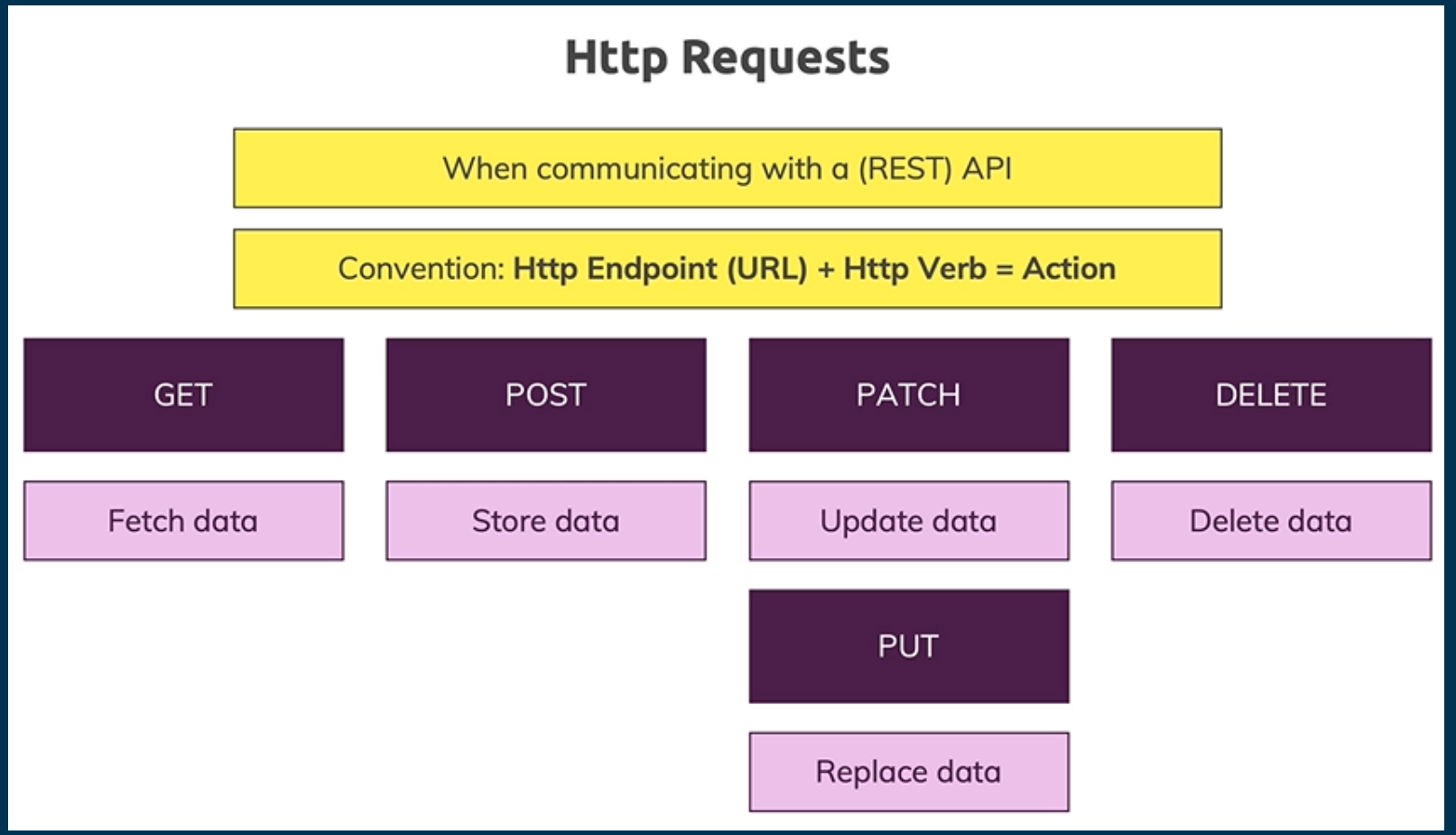
```dart
import 'package:flutter/material.dart';

// ignore: must_be_immutable
class WelcomePage extends StatelessWidget {
  String name,surname;
  WelcomePage({Key? key,required this.name,required this.surname}) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return Scaffold(
   body: Center(
     child: Center(
       child: Column(
         crossAxisAlignment: CrossAxisAlignment.start,
         children: [
           Text('Name: $name'),
           Text('Surname: $surname'),
         ],
       ), // Column
     ), // Center
   ), // Center
   ); // Scaffold
 }
}
```

```dart
Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => WelcomePage(
                                                  name: _name,
                                                  surname: _surname,
                                                ) // WelcomePage
                        ) // MaterialPageRoute
                        );
```

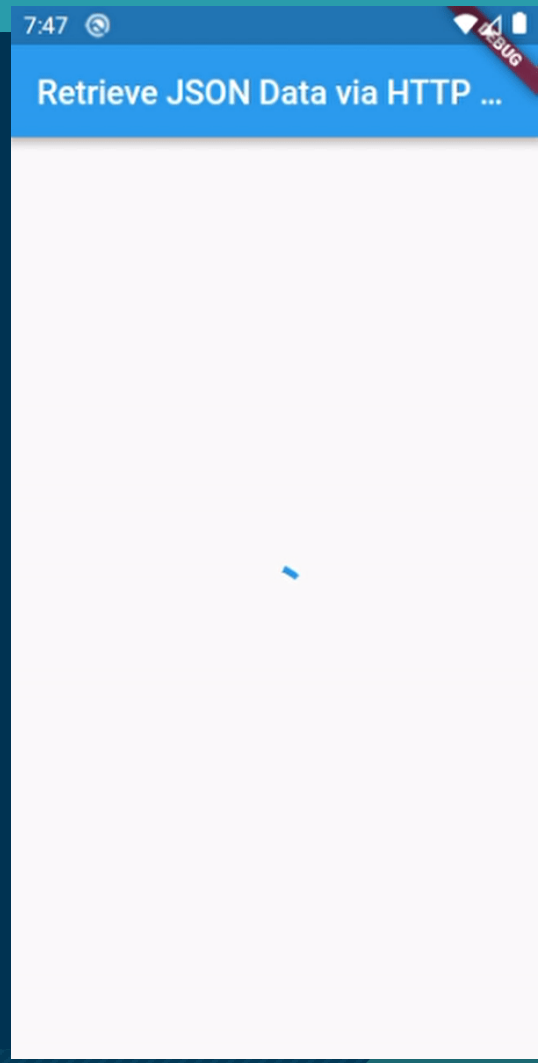
Http requests



HTTP Request

- Η http.get επιστρέφει ένα αντικείμενο future
- Χρησιμοποιούμε την μέθοδο .then((value){}) σε ένα future όταν θέλουμε να εκτελέσουμε κώδικα αφού εκχωρηθούν στο future δεδομένα.
- Στο παράδειγμα χρησιμοποιούμε την async μεθοδο για να δείξουμε ένα loader ο οποίος εμφανίζεται στην οθόνη μέχρι την αλλαγή της τιμής της μεταβλητής isLoading από true σε false

```dart
class MyGetHttpDataState extends State<MyGetHttpData> {
  // API endpoint
  final String url = "https://jsonplaceholder.typicode.com/users";
  // List to store the fetched data
  List data = [];
  // Loading indicator flag
  var isLoading = true;
  @override
  void initState() {
    super.initState();
    // Call the method to fetch data when the app starts
    getJSONData();
  }
  // Function to fetch JSON data from the API
  Future<void> getJSONData() async {
    try {
      final response = await http.get(Uri.parse(url));
      // Print status and response body for debugging
      if (response.statusCode == 200) {
        // Convert JSON string into Dart object (List/Map)
        final decodedData = json.decode(response.body);
        // Update the state with the fetched data
        setState(() {
          data = decodedData; // Here the API returns a list directly
          isLoading = false;  // Stop showing the loader
        });
      } else {
        // Handle request failure
        setState(() {
          isLoading = false;
        });
      }
    } catch (error) {
      // Handle any errors (e.g., network issues)
      setState(() {
        isLoading = false;
      });
    }
  }
```



GestureDetector vs InkWell

- Και τα δύο παρέχουν πολλές κοινές λειτουργίες όπως το onTap, onLongPress κ.λπ. Η κύρια διαφορά είναι ότι το GestureDetector παρέχει περισσότερους ελέγχους όπως dragging κ.λ.π. ενώ δεν περιλαμβάνει το ripple effect, το οποίο κάνει το InkWell.
- Μπορείτε να χρησιμοποιήσετε οποιοδήποτε από τα δυο ανάλογα με τις ανάγκες σας,
  - αν θέλετε το εφέ κυματισμού (ripple effect) επιλέγετε το InkWell,
  - αν χρειάζεστε περισσότερα στοιχεία ελέγχου επιλέγετε το GestureDetector
  - ή μπορείτε να συνδυάσετε και τα δύο.

StaggeredGridView

- Package: https://pub.dev/packages/flutter_staggered_grid_view
  - Οδηγίες εγκατάστασης στο tab “Installing”
- Εισάγουμε στο pubspec.yaml :
  - flutter_staggered_grid_view:
- Αν δεν εγκατασταθεί αυτόματα όταν σώσουμε το αρχείο τρέχουμε:
  - flutter pub get
- Το κάνουμε import στο αρχείο που θέλουμε να το χρησιμοποιήσουμε:
  - import 'package:flutter_staggered_grid_view/flutter_staggered_grid_view.dart';

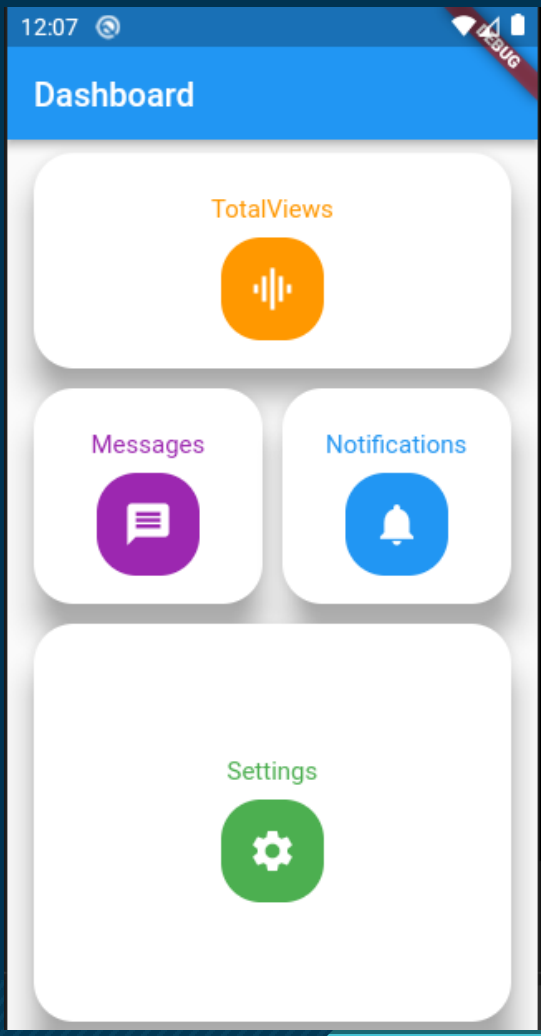
StaggeredGridView

- StaggeredGridView.count ο constructor που χρησιμοποιείται πιο συχνά
- crossAxisCount:
  - Ο αριθμός των στηλών που θέλουμε να καλύπτει το grid view
- crossAxisSpacing:
  - Το κενό αναμεσά στα κουτιά/tiles στον οριζόντιο άξονα
- mainAxisSpacing:
  - Το κενό αναμεσά στο κουτιά/tiles στον κάθετο άξονα

StaggeredGridView

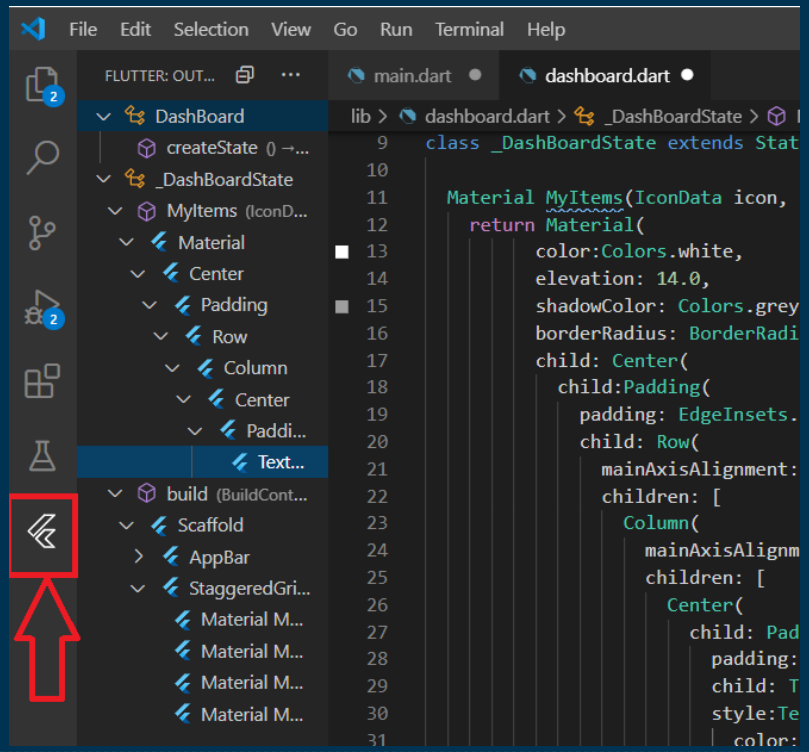
- Το TotalViews καλύπτει και τις 2 στήλες που έχουμε δηλώσει στο crossAxisCount και έχει μικρότερο ύψος από το Settings γιατί στο mainAxisCellCount έχουμε την τιμή 1
- Το messages και notifications δηλώνουμε ότι καλύπτει το καθένα από μια στήλη για να δημιουργηθούν διπλά - διπλά

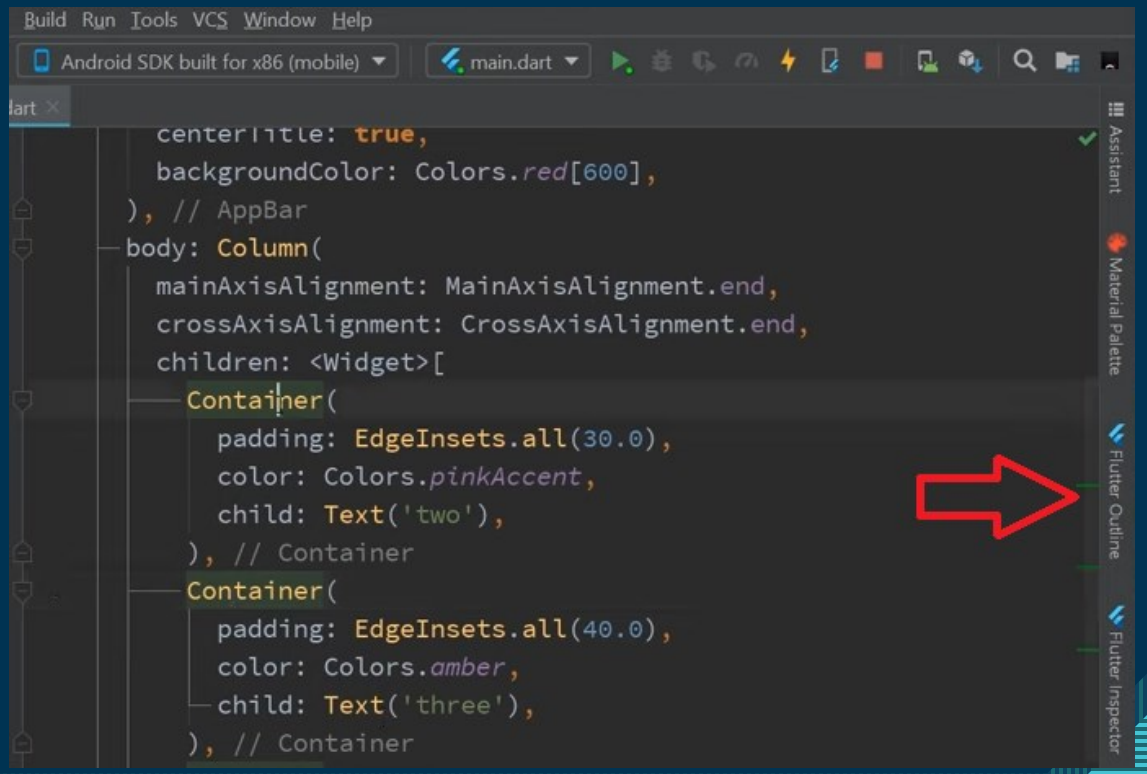
```
body: SingleChildScrollView(
  child: StaggeredGrid.count(
    crossAxisCount:
        2, //ο αριθμος των στηλων που θελουμε να καλυπτει το grid
    crossAxisSpacing:
        12.0, //η αποσταση μεταξυ των κουτιων στον οριζοντιο αξονα
    mainAxisSpacing:
        12.0, //η αποσταση μεταξυ των κουτιων στον καθετο αξονα
    children: [
      StaggeredGridTile.count(
        crossAxisCellCount: 2,//Ο αριθμός των κελιών που παίρνει αυτό το πλακί
        mainAxisCellCount: 1,//Ο αριθμός των κελιών που παίρνει αυτό το πλακίδ
        child: myItems(Icons.graphic_eq, "TotalViews", Colors.orange),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 1,
        mainAxisCellCount: 1,
        child: myItems(Icons.message, "Messages", Colors.purple),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 1,
        mainAxisCellCount: 1,
        child:
            myItems(Icons.notifications, "Notifications", Colors.blue),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 2,
        mainAxisCellCount: 2,
        child: myItems(Icons.settings, "Settings", Colors.green),
      ), // StaggeredGridTile.count
    ],
  ), // StaggeredGrid.count
)); // SingleChildScrollView // Scaffold
```



Flutter outline

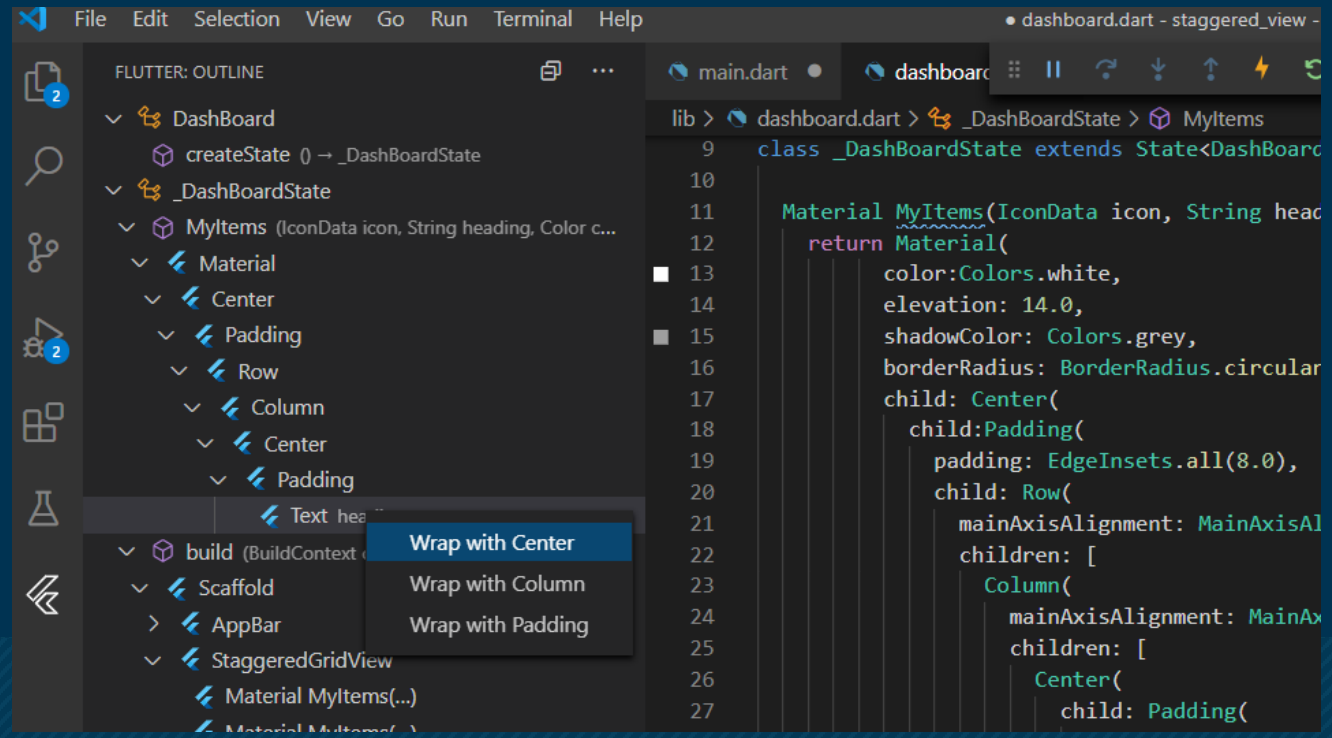
- Command Pallete → γράφουμε outline view → επιλέγουμε outline view





Flutter outline

- Επιλέγουμε το widget που θέλουμε (γίνεται highlighted στον κώδικα)
- Στο widget tree του outline view πατάμε δεξί κλικ στο widget που θέλουμε και μπορούμε να το κάνουμε wrap με αλλά widgets όπως padding, center κτλ.



Ερωτήσεις

- Πώς λειτουργεί η αποστολή αιτημάτων HTTP σε ένα (REST) API;
  - Η εφαρμογή Flutter στέλνει αιτήματα με συγκεκριμένα Http Verbs και endpoints (paths) στον web server(= το API).
- Τι είναι το “future" στην Dart;
  - Ένα αντικείμενο που περιμένει την ολοκλήρωση μιας ασύγχρονης λειτουργίας και στη συνέχεια (πιθανώς) χρησιμοποιεί τα δεδομένα.
- Τι είναι ο "ασύγχρονος" (ή “async") κώδικας;
  - Κώδικας που δεν εκτελείται αμέσως αλλά εκτελείται κάποια στιγμή στο μέλλον. Ο υπόλοιπος κώδικας συνεχίζει την εκτέλεση χωρίς να περιμένει την ολοκλήρωση του async κώδικα.

Const warning

- Prefer const with constant constructors.dart (prefer_const_constructors)
- Solution:
  - We use const in as many constructors as we can
  - Insert at the beginning of the file: // ignore_for_file: prefer_const_constructors
  - Insert in file analysis_options.yaml under rules
    - prefer_const_constructors : false

```
rules:
  prefer_const_constructors : false
  # avoid_print: false  # Uncomment to disable the `avoid_print` rule
  # prefer_single_quotes: true  # Uncomment to enable the `prefer_single_quotes` rule
```

Variable late

- When the compiler considers that a variable should not be null we use late
  - The compiler with late "trusts" us to initialize the non-nullable variable after initializing it

```
late List<GDPData> _chartData;
late TooltipBehavior _tooltipBehavior;
```

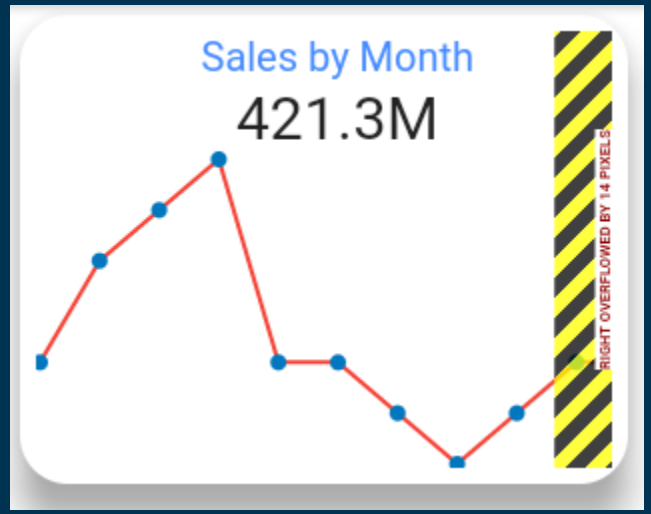
- Lazily initializing a variable
  - The variable may not be needed and its initialization may be "expensive".
  - In the example below if result is never used the "expensive" function _getResult is never called

```
  // This is the program's only call to _getResult().
late String result = _getResult(); // Lazily initialized.
```

Common Overflow Error

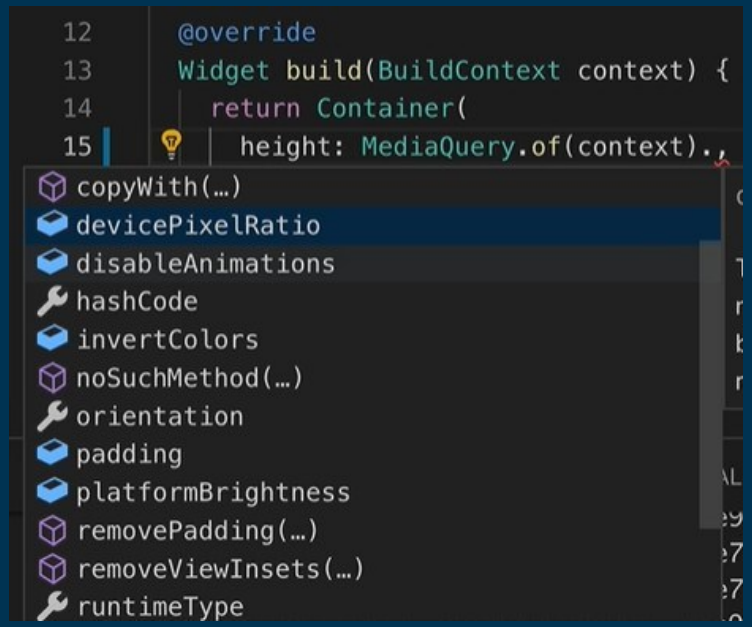
- In the example, the column tries cover more space than the space that can be allocated inside the row, causing an overflow error. The row in the example is the Parent of the Column. Solution:

```
child: Row(
  children: [
    Flexible(
      child: Column(
        // code
              ),
            ),
          ]
        )
```



Get Dynamically Device Size

- With the MediaQuery class we have several options:



- The size option gives us the height and width.
  - We can use the percentage of these values by multiplying the height/width with the percentage (60% -> 0.6).

```
@override
Widget build(BuildContext context) {
  return Container(
    height: MediaQuery.of(context).size.height * 0.6,
```

SQFlite

- Υποστηρίζεται από Android και σε iOS
- Στις εφαρμογές για κινητά για να διατηρήσουμε τα δεδομένα σε τοπικό επίπεδο χρησιμοποιούμε την τοπική βάση δεδομένων SQlite
- **SQFlite** plugin : https://pub.dev/packages/sqflite
- CRUD operations
  - Τα create, read, update, και delete (CRUD) είναι οι τέσσερις βασικές λειτουργίες της μόνιμης αποθήκευσης

Βήματα

- Setup **SQFlite**
  - Εισάγουμε τα παρακάτω πακέτα στο pubspec.yaml file κάτω από flutter sdk με αυτή την ακριβή στοίχιση/κενά.
    sqflite: κατάλληλη έκδοση
    path: κατάλληλη έκδοση
    path_provider: κατάλληλη έκδοση
- Open database
  - Create database
- Create tables
- Perform CRUD
- Flutter Packages versions:
  - any : update στην τελευταία έκδοση ακόμα και αν δεν είναι συμβατή
  - κενό: update στην τελευταία έκδοση αλλά θα λάβει υπόψιν την συμβατότητα

```
environment:
  sdk: ">=2.7.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.0.0+4
  path: ^1.8.0
  path_provider: ^2.0.7
```

```
  sqflite:
  path: any
  path_provider:
```

Δημιουργία βάσης

- Η δημιουργία βάσης επιστρέφει ένα future.
- Το getDatabasesPath() επιστρέφει το μονοπάτι που αποθηκεύεται στη συσκευή ανάλογα αν είναι android/iOS
- path.join: συνενώνει το μονοπάτι αποθήκευσης με το όνομα που δίνουμε στη βάση μας πχ ‘mydb.db’
- openDatabase(): αν υπάρχει η βάση στο συγκεκριμένο μονοπάτι την ανοίγει διαφορετικά την δημιουργεί
- Στο onCreate δίνουμε την εντολή δημιουργίας του πίνακα
- Ακολουθεί τους κανόνες τις SQL

```
static Future<sql.Database> database() async{
    //δημιουργια βασης αν δεν υπαρχει
final dbPath = await sql.getDatabasesPath(); //to path αναλογα android/iOS
return sql.openDatabase(path.join(dbPath,'mydb.db'),version:1,
onCreate:_createDB,
); //αν υπαρχει το αρχειο ανοιγει την βαση διαφορετικα την δημιουργει
```

```
static Future _createDB(db,version){
    return db.execute('''
    CREATE TABLE notes(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      title TEXT NOT NULL,
      text TEXT NOT NULL,
      extra TEXT)'''); //επιστρεφει ενα future
}
```

Εισαγωγή στην βάση

- db.insert(): κάνει εισαγωγή δεδομένων στον αντίστοιχο πίνακα table
- conflictAlgorithm: επιλέγεις τι θέλεις να γίνει σε περίπτωση που το id (PRIMARY KEY) υπάρχει ήδη π.χ. να γίνει αντικατάσταση ή καμία ενέργεια

```
static Future<void> insert( String table, Map<String,Object> data) async{
 print(data); //τυπωνω τα δεδομένα για να δω οτι ειναι σωστά
 final db = await DBHelper.database();
 db.insert(
   table,
   data,
   conflictAlgorithm:
   sql.ConflictAlgorithm.ignore);
//με το await εδω περιμενουμε να τελειωσουν ολες οι διαδικασιες στην συναρτηση insert


}
```

Επιστροφή δεδομένων

- dq.query: Όταν δοθεί μόνο το όνομα του πίνακα (όπως εδώ) επιστρέφει ότι περιέχει ο πίνακας
  - Ονομαστική μεταβλητή where: εκτελεί το WHERE στην εντολή SELECT

- Επιστρέφει ένα future και πιο συγκεκριμένα μια λίστα map που έχει String και dynamic

```
static Future<List<Map<String,dynamic>>> getData(table) async {
 final db = await DBHelper.database();
    return db.query(table); //επιστρεφει μια λιστα απο maps
 }
```

```
static Future<List<Map<String,dynamic>>> getData(table) async {
 final db = await DBHelper.database();
    return db.query(table,where: "title LIKE '%titleofnote%'");
 }
```