Kolegium Nauk Przyrodniczych
Uniwersytet Rzeszowski

Przedmiot:

Programowanie Urządzeń Mobilnych

Nazwa projektu:

Lista zadań (aplikacja to-do)

Wykonał:

Karol Bury, Informatyka rok III, lab1

Prowadzący: dr inż. Piotr Lasek

Rzeszów 2021

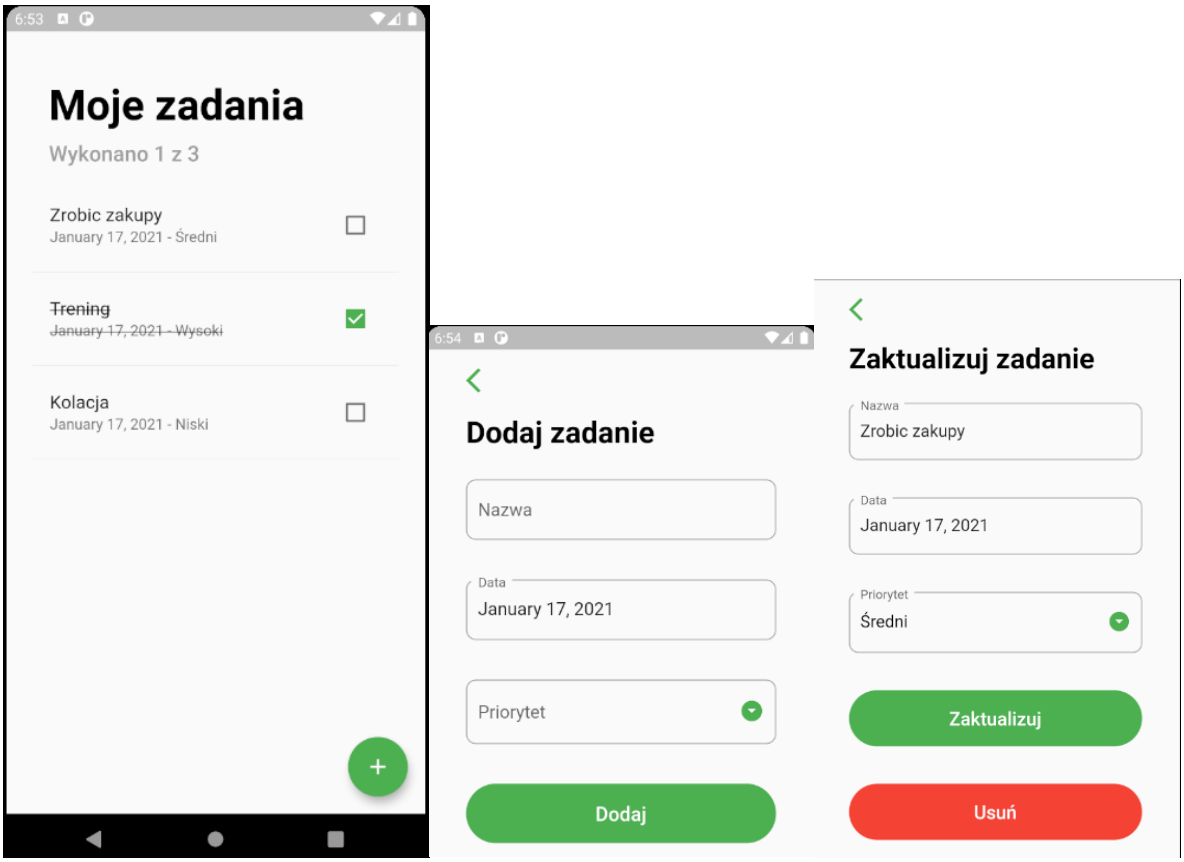
Spis treści:

1. Opis aplikacji

Projekt „Lista zadań (aplikacja to-do) to aplikacja, w której możemy dodawać zadania, które mamy do wykonania, dzięki czemu możemy łatwo zarządzać naszym czasem. Zadania mają przypisany priorytet (niski, średni lub wysoki). Gdy wykonamy dane zadanie zaznaczamy przy nim checkboxa i jego status zmienia się na wykonane.

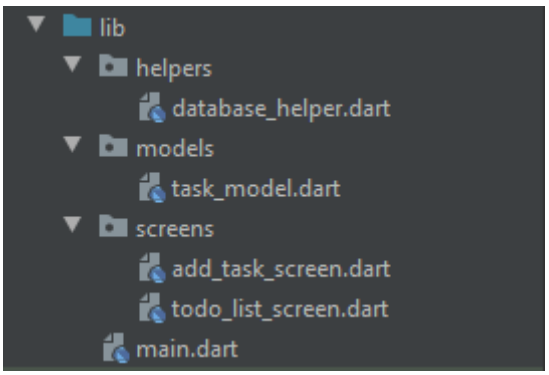
Zadania możemy edytować oraz usuwać.

Aplikacja została stworzona przy użyciu języka Dart z frameworkiem Flutter.



2. Opis funkcjonalności

W folderze „lib” znajdują się najważniejsze pliki.



Są to:

- folder helpers a w nim plik database_helper.dart
- folder models a w nim plik task_model.dart
- folder screen a w nim:
    - plik add_task_screen.dart
    - plik todo_list_screen.dart
- plik main.dart

2.1. database_helper.dart

Ten plik odpowiada za bazę danych. W bazie są następujące pola: id, tytuł, data, priorytet, status (0 gdy zadanie jest niewykonane oraz 1 gdy zadanie jest wykonane).

```
String colId = 'id';
String colTitle = 'title';
String colDate = 'date';
String colPriority = 'priority';
String colStatus = 'status';
```

Pobieranie listy zadań oraz dodawanie zadań:

```
Future<List<Task>> getTaskList() async {
  final List<Map<String, dynamic>> taskMapList = await getTaskMapList();
  final List<Task> taskList = [];
  taskMapList.forEach((taskMap) {
    taskList.add(Task.fromMap(taskMap));
  });
  taskList.sort((taskA, taskB) => taskA.date.compareTo(taskB.date));
  return taskList;
}

Future<int> insertTask(Task task) async {
  Database db = await this.db;
  final int result = await db.insert(tasksTable, task.toMap());
  return result;
}
```

Aktualizacja oraz usuwanie zadań:

```
  Future<int> updateTask(Task task) async {
    Database db = await this.db;
    final int result = await db.update(
        tasksTable,
        task.toMap(),
        where: '$colId = ?',
        whereArgs: [task.id],
    );
    return result;
  }

  Future<int> deleteTask(int id) async {
    Database db = await this.db;
    final int result = await db.delete(
      tasksTable,
      where: '$colId = ?',
      whereArgs: [id],
    );
    return result;
  }
}
```

2.2. task_model.dart

```dart
class Task {
  int id;
  String title;
  DateTime date;
  String priority;
  int status; // 0 - zadanie niewykonane, 1 - zadanie wykonane

  Task({this.title, this.date, this.priority, this.status});
  Task.withId({this.id, this.title, this.date, this.priority, this.status});

  Map<String, dynamic> toMap() {
    final map = Map<String, dynamic>();
    if (id != null) {
      map['id'] = id;
    }
    map['title'] = title;
    map['date'] = date.toIso8601String();
    map['priority'] = priority;
    map['status'] = status;
    return map;
  }

  factory Task.fromMap(Map<String, dynamic> map) {
    return Task.withId(
        id: map['id'],
        title: map['title'],
        date: DateTime.parse(map['date']),
        priority: map['priority'],
      status: map['status'],
    );
  }
}
```

2.3. add_task_screen.dart

Plik obsługujący ekran dodawania zadań. Poniżej widać funkcje usuwania oraz dodawania i aktualizacji.

```dart
_delete() {
  DatabaseHelper.instance.deleteTask(widget.task.id);
  widget.updateTaskList();
  Navigator.pop(context);
}

_submit() {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    print('$_title, $_date, $_priority');

    //Dodanie zadania do bazy danych
    Task task = Task(title: _title, date: _date, priority: _priority);
    if (widget.task == null) {
      task.status = 0;
      DatabaseHelper.instance.insertTask(task);
    } else{
      //Zaktualizowanie zadania
      task.id = widget.task.id;
      task.status = widget.task.status;
      DatabaseHelper.instance.updateTask(task);
    }

    widget.updateTaskList();
    Navigator.pop(context);
  }
}
```

Fragment kodu:

```
Container(
  margin: EdgeInsets.symmetric(vertical: 20.0),
  height: 60.0,
  width: double.infinity,
  decoration: BoxDecoration(
    color: Theme.of(context).primaryColor,
    borderRadius: BorderRadius.circular(30.0),
  ),  // BoxDecoration
  child: FlatButton(
    child: Text(
        widget.task == null ? 'Dodaj' : 'Zaktualizuj',
        style: TextStyle(
          color: Colors.white,
          fontSize: 20.0),  // TextStyle
      ),  // Text
    onPressed: _submit,
  ),  // FlatButton
),  // Container
widget.task != null ? Container(
  margin: EdgeInsets.symmetric(vertical: 20.0),
  height: 60.0,
  width: double.infinity,
  decoration: BoxDecoration(
    color: Colors.red,
    borderRadius: BorderRadius.circular(30.0),
  ),  // BoxDecoration
```

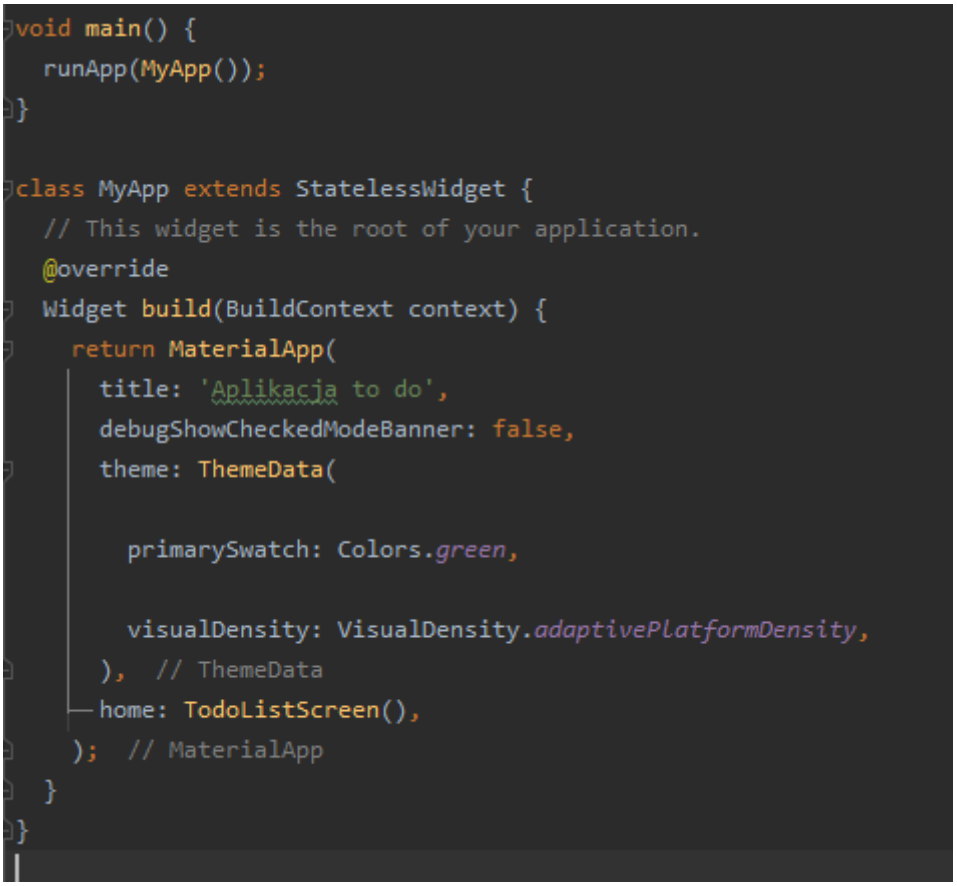
2.4. todo_list_screen.dart

Plik obsługujący wyświetlanie listy zadań.

```
return ListView.builder(
  padding: EdgeInsets.symmetric(vertical: 50.0),
  itemCount: 1 + snapshot.data.length,
  itemBuilder: (BuildContext context, int index) {
    if (index == 0) {
      return Padding(
        padding: EdgeInsets.symmetric(horizontal: 40.0, vertical: 20.0),
        child:  Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
          Text('Moje zadania', style: TextStyle(
              color: Colors.black,
              fontSize: 40.0,
              fontWeight: FontWeight.bold,
            ),  // TextStyle
            ),  // Text
          SizedBox(height: 10.0),
          Text(
              'Wykonano ${completedTaskCount} z ${snapshot.data.length}',
              style: TextStyle(
              color: Colors.grey,
              fontSize: 20.0,
              fontWeight: FontWeight.w600,
            ),  // TextStyle
            ),  // Text
          ]  // <Widget>[]
    ),  // Column
      );  // Padding
    }
    return _buildTask(snapshot.data[index - 1]);
  },
```

2.5. main.dart

Plik, który ustala ekran główny, główny kolor, tytuł oraz uruchamia aplikację.



```dart
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Aplikacja to do',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(

        primarySwatch: Colors.green,

        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),  // ThemeData
      home: TodoListScreen(),
    );  // MaterialApp
  }
}
```

Podsumowanie

Język Dart z frameworkiem Flutter jest prostym środowiskiem do tworzenia graficznego interfejsu użytkownika.