



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

ФАКУЛЬТЕТ \_\_\_\_\_ «Информатика и системы управления»

КАФЕДРА \_\_\_\_\_ «Теоретическая информатика и компьютерные технологии»

**Летучка № 3**  
**по курсу «Разработка мобильных приложений»**

Студент группы ИУ9-71Б Окутин Д. А.

Преподаватель Посевин Д. П.

*Москва 2024*

# 1 Цель

Разобраться с использованием MySQL базы данных в Dart приложениях.

## 2 Задание

Необходимо разработать приложение для работы с базой данных MySQL. Приложение должно иметь функционал сохранения сущности в таблицу, получения списка сущностей, очистка таблицы.

Интегрировать полученный сервис в лабораторную работу номер 4.

## 3 Реализация

Исходный код представлен в листинге 1.

Листинг 1: main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:mysql1/mysql1.dart';
3
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'MySQL App',
13      home: UserForm(),
14    );
15  }
16 }
17
18 class UserForm extends StatefulWidget {
19   @override
20   _UserFormState createState() => _UserFormState();
21 }
22
23 class _UserFormState extends State<UserForm> {
24   final _formKey = GlobalKey<FormState>();
25   String _name = '';
```

```

26 String _email = '';
27 int _age = 0;
28 List<Map> _users = [];
29
30 Future<MySQLConnection> _getConnection() async {
31     return await MySqlConnection.connect(
32         ConnectionSettings(
33             host: 'students.yss.su',
34             port: 3306,
35             user: 'iu9mobile',
36             db: 'iu9mobile',
37             password: 'bmstubmstul23',
38         ),
39     );
40 }
41
42 Future<void> _createTable() async {
43     final conn = await _getConnection();
44     await conn.query('''
45         CREATE TABLE IF NOT EXISTS users (
46             id INT AUTO_INCREMENT PRIMARY KEY,
47             name VARCHAR(100),
48             email VARCHAR(100),
49             age INT
50         )
51     ''');
52     await conn.close();
53 }
54
55 Future<void> _saveUser() async {
56     final conn = await _getConnection();
57     await conn.query('INSERT INTO users (name, email, age) VALUES (?, ?,
58         ?)',
59         [_name, _email, _age]);
60     await conn.close();
61 }
62
63 Future<void> _fetchUsers() async {
64     final conn = await _getConnection();
65     var results = await conn.query('SELECT name, email, age FROM users ');
66     ;
67     setState(() {
68         _users = results
69             .map((row) => {'name': row[0], 'email': row[1], 'age': row
70                 [2]})
71             .toList();

```

```

69     });
70     await conn.close();
71 }
72
73 Future<void> _deleteUsers() async {
74     final conn = await _getConnection();
75     await conn.query('DELETE FROM users');
76     await conn.close();
77 }
78
79 @override
80 void initState() {
81     super.initState();
82     _createTable(); //
83 }
84
85 @override
86 Widget build(BuildContext context) {
87     return Scaffold(
88         appBar: AppBar(title: Text('User Management')),
89         body: Padding(
90             padding: const EdgeInsets.all(16.0),
91             child: Form(
92                 key: _formKey,
93                 child: Column(
94                     children: [
95                         TextFormField(
96                             decoration: InputDecoration(labelText: 'Name'),
97                             onSave: (value) => _name = value!,
98                         ),
99                         TextFormField(
100                             decoration: InputDecoration(labelText: 'Email'),
101                             onSave: (value) => _email = value!,
102                         ),
103                         TextFormField(
104                             decoration: InputDecoration(labelText: 'Age'),
105                             keyboardType: TextInputType.number,
106                             onSave: (value) => _age = int.tryParse(value!) ?? 0,
107                         ),
108                         SizedBox(height: 20),
109                         ElevatedButton(
110                             onPressed: () {
111                                 if (_formKey.currentState!.validate()) {
112                                     _formKey.currentState!.save();

```

```

113         _saveUser();
114     }
115 },
116     child: Text('
117 '),
118 ),
119 ElevatedButton(
120     onPressed: () {
121         _fetchUsers();
122     },
123     child: Text('
124 '),
125 ),
126 ElevatedButton(
127     onPressed: () {
128         _deleteUsers();
129     },
130     child: Text('
131 '),
132 ),
133 Expanded(
134     child: ListView.builder(
135         itemCount: _users.length,
136         itemBuilder: (context, index) {
137             return ListTile(
138                 title: Text(
139                     '${_users[index]['name']} (${_users[index]['
140 age']})'),
141                 subtitle: Text(_users[index]['email']),
142             );
143         },
144     ),
145 ),
146 );
147 }
148 }

```

## **4 Результаты**

Результат представлен на рисунке 1.

# User Management

Имя

---

Емейл

---

Возраст

---

Сохранить пользователя

Получить всех пользователей

Удалить всех пользователей

Рис. 1 — Интерфейс приложения

## **5 Выводы**

В результате работы было создано приложение для взаимодействия с базой данных MySQL.