Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа №9

по дисциплине "Технологии разработки мобильных приложений"

ТЕМА РАБОТЫ:

Разработка клиент-серверного мобильного приложения

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

преподаватель кафедры ИСПИ

Рощина А.И.

Владимир 2023 г.

Цель работы:

Научиться разрабатывать мобильное приложение с использованием клиент-серверной архитектуры и протокола Rest API.

Выполнение работы:

Серверная часть приложения была написана на фреймворке ASP.NET Core, предметная область приложения – «Многозадачный календарь».

Она включает в себя пользователя (без ролей – разделение происходит только на аутентифицированных и неаутентифицированных пользователей). Аутентифицированные пользователи имеют доступ к своему личному кабинету (и могут взаимодействовать с остальными сущностями системы – а именно создавать, редактировать и просматривать свои группы, создавать и просматривать свои запланированные мероприятия и задачи в календаре, а также получать отчеты).

Неаутентифицированный пользователь должен войти в систему или зарегистрироваться в ней – чтобы получить свой личный токен (по которому на уровне middleware сервера осуществляется проверка аутентификации запроса каждого пользователя). В случае, если пользователь отправляет запрос без токена (или токен устарел) – то сервер дает ответ HTTP 401 Unathorized. В случае, если же токен существует – но пользователь не имеет доступа к какой-то информации (например, не может редактировать чужие группы и задачи, участвовать в мероприятиях, в которых он не был приглашен – то тогда сервер возвращает ответ HTTP 403 Forbidden.

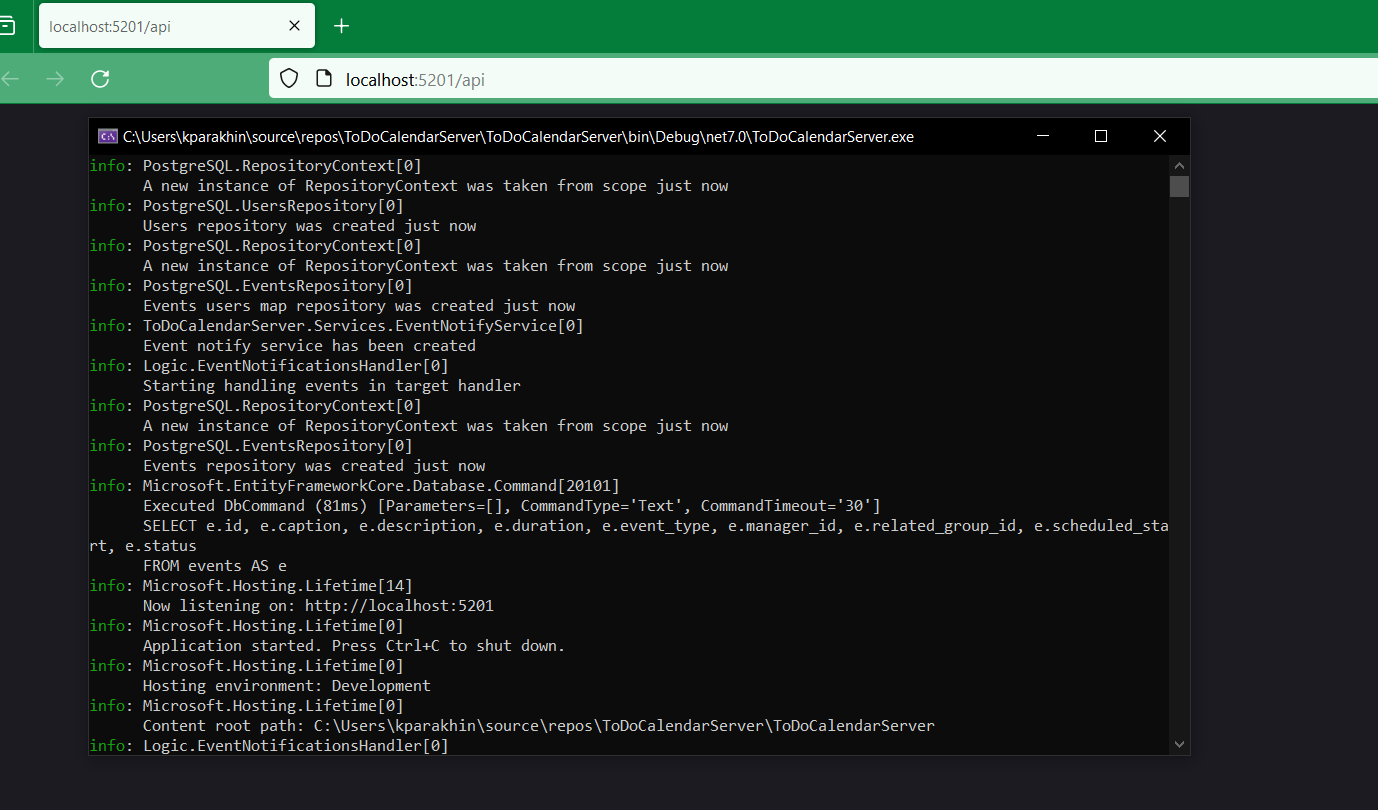


Рисунок 1. Запуск серверного приложения на порту localhost:5201

Клиентская часть приложения будет написана с использованием фреймворка Flutter.

Для выполнения http запросов к API нашего сервера будем использовать пакет: *http: ^0.13.3*

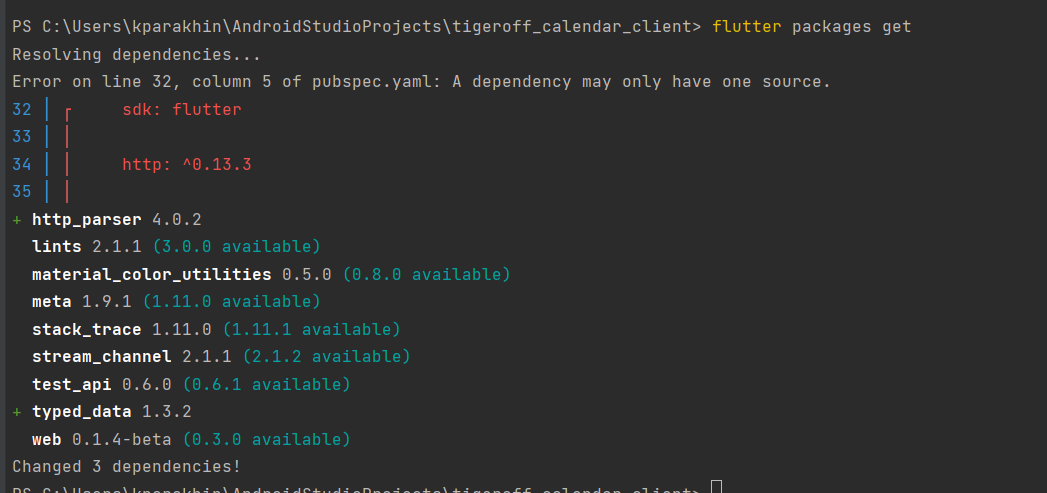


Рисунок 2. Получение зависимостей для работы с http в терминале Flutter

Для корректного «общения» серверного и клиентского приложения – необходимо обеспечить их протокол взаимодействия. Для этого будет использовать HTTP Restful API.

Для передачи данных между двумя HTTP-клиентами будут использоваться json-объекты, которые необходимо корректно сериализовать/десериализовать – поэтому в клиентском приложении были созданы модели по аналогии с серверными моделями.

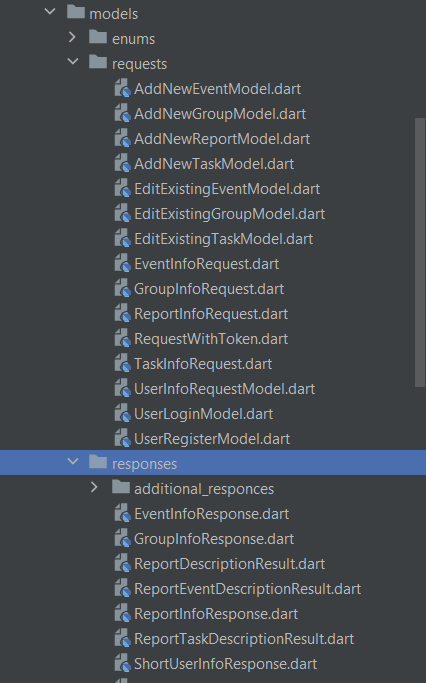


Рисунок 3. Модели клиентского приложения

Создадим основные страницы (виджеты) – для регистрации, авторизации, для личного кабинета пользователя (с перечислением мероприятий, групп, задач и отчетов пользователя), а также дополнительные виджеты для добавления/редактирования мероприятия в календарь, новой группы, новой задачи на реализацию, нового отчета (по мероприятиям или отчетам).

Листинг запускаемого файла main.dart:

void main() {  
 runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'Многозадачный календарь',  
 theme: ThemeData(  
 primarySwatch: Colors.*lightGreen*,  
 ),  
 initialRoute: '/',  
 routes: {  
 '/': (context) => HomePage(),  
 '/login': (context) => LoginPage(),  
 '/register': (context) => RegisterPage(),  
 '/session/user\_page': (context) => UserPage(),  
 '/session/events\_page': (context) => UserInfoMapPage(),  
 '/session/additional\_page': (context) => AdditionalPageWidget()  
 },  
 );  
 }  
}

Листинг начальной страницы home\_page.dart:

class HomePage extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Календарь Tigeroff приветствует вас сегодня !'),  
 ),  
 body: Center(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(50.0)),  
 minimumSize: Size(200, 80),  
 ),  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => LoginPage()),  
 );  
 },  
 child: Text('Авторизация'),  
 ),  
 SizedBox(height: 40),  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(50.0)),  
 minimumSize: Size(200, 80),  
 ),  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => RegisterPage()),  
 );  
 },  
 child: Text('Регистрация'),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

Листинг странички для авторизации:

class LoginPage extends StatelessWidget {  
 final TextEditingController emailController = TextEditingController();  
 final TextEditingController passwordController = TextEditingController();  
  
 bool isAlerted = false;  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Вход в существующую учетную запись'),  
 ),  
 body: Padding(  
 padding: EdgeInsets.all(16.0),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 TextField(  
 controller: emailController,  
 decoration: InputDecoration(  
 labelText: 'Электронная почта: ',  
 ),  
 ),  
 SizedBox(height: 16.0),  
 TextField(  
 controller: passwordController,  
 obscureText: true,  
 decoration: InputDecoration(  
 labelText: 'Пароль: ',  
 ),  
 ),  
 SizedBox(height: 25.0),  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(36.0)),  
 minimumSize: Size(150, 60),  
 ),  
 onPressed: () {  
 login(context);  
 },  
 child: Text('Войти'),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
  
 Future<void> login(BuildContext context) async {  
 String email = emailController.text;  
 String password = passwordController.text;  
  
 var model = new UserLoginModel(email: email, password: password);  
  
 var requestMap = model.toJson();  
  
 final url = Uri.*parse*('http://localhost:5201/users/login');  
 final headers = {'Content-Type': 'application/json'};  
 final body = jsonEncode(requestMap);  
  
 final response = await http.post(url, headers: headers, body: body);  
  
 var jsonData = jsonDecode(response.body);  
 var responseContent = ResponseWithToken.fromJson(jsonData);  
  
 MySharedPreferences mySharedPreferences = new MySharedPreferences();  
  
 await mySharedPreferences.clearData();  
  
 await mySharedPreferences.saveDataWithExpiration(response.body, const Duration(days: 7));  
  
 if (responseContent.result) {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(builder: (context)  
 => UserPage()),  
 );  
 } else {  
 showDialog(  
 context: context,  
 builder: (context) => AlertDialog(  
 title: Text('Ошибка!'),  
 content: Text('Неверный логин или пароль!'),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.*pop*(context);  
 },  
 child: Text('OK'),  
 ),  
 ],  
 ),  
 );  
 }  
  
 emailController.clear();  
 passwordController.clear();  
 }  
  
 showAppDialog(BuildContext context) {  
 print("Showing app dialog");  
 showDialog(context: context,  
 builder: (context) {  
 return AlertDialog(  
 title: const Text(  
 "This is a dialog that works.",  
 ),  
 icon: const Icon(Icons.*delete*),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.*of*(context).pop();  
 },  
 child: const Text("OK"),  
 ),  
 ],  
 );  
 });  
 }  
  
 Widget showAlertDialog(BuildContext context) {  
 return MaterialApp(  
 title: 'Flutter Demo',  
 debugShowCheckedModeBanner: false,  
 theme: ThemeData(  
 primarySwatch: Colors.*blue*,  
 ),  
 home: Scaffold(body: SafeArea(child: Builder(  
 builder: (context) {  
 return TextButton(child: Text("Show dialog"), onPressed: () => showAppDialog(context),);  
 }  
 ))),  
 );  
 }  
}

Листинг моделей для авторизации (которая отправляется на сервер по HTTP-POST запросу в виде json-строки и приходит ответом в виде другой строки, десериализуемой в модель ответа):

class UserLoginModel {  
  
 final String email;  
 final String password;  
  
 UserLoginModel({  
 required this.email,  
 required this.password,  
 });  
  
 Map<String, dynamic> toJson() {  
 return {  
 'email': email,  
 'password': password,  
 };  
 }  
  
 String serialize() {  
 return jsonEncode(toJson());  
 }  
}

Модель ответа:

class ResponseWithToken extends Response{  
  
 final int userId;  
 final String? token;  
  
 ResponseWithToken({  
 required bool result,  
 String? outInfo,  
 required this.userId,  
 this.token  
 }) :super(result: result, outInfo: outInfo);  
  
 factory ResponseWithToken.fromJson(Map <String, dynamic> json) {  
 return ResponseWithToken(  
 result: json['result'],  
 outInfo: json['out\_info'],  
 userId: json['user\_id'],  
 token: json['token']  
 );  
 }  
}

В случае успешного ответа – загружается страница личного кабинета (user\_page.dart), при этом параметры идентификатора пользователя и его токен аутентификации сохраняются в локальном кэше с использованием пакета

import 'package:shared\_preferences/shared\_preferences.dart';

Листинг класса для реализации кэширования:

class MySharedPreferences {  
 static const String *\_keyData* = 'myData';  
 static const String *\_keyExpiration* = 'expirationTime';  
  
 // Function to save data with an expiration date to SharedPreferences  
 Future<bool> saveDataWithExpiration(String data, Duration expirationDuration) async {  
 try {  
 SharedPreferences prefs = await SharedPreferences.*getInstance*();  
 DateTime expirationTime = DateTime.now().add(expirationDuration);  
 await prefs.setString(*\_keyData*, data);  
 await prefs.setString(*\_keyExpiration*, expirationTime.toIso8601String());  
 print('Data saved to SharedPreferences.');  
 return true;  
 } catch (e) {  
 print('Error saving data to SharedPreferences: $e');  
 return false;  
 }  
 }  
  
 // Function to get data from SharedPreferences if it's not expired  
 Future<String?> getDataIfNotExpired() async {  
 try {  
 SharedPreferences prefs = await SharedPreferences.*getInstance*();  
 String? data = prefs.getString(*\_keyData*);  
 String? expirationTimeStr = prefs.getString(*\_keyExpiration*);  
 if (data == null || expirationTimeStr == null) {  
 print('No data or expiration time found in SharedPreferences.');  
 return null; // No data or expiration time found.  
 }  
  
 DateTime expirationTime = DateTime.*parse*(expirationTimeStr);  
 if (expirationTime.isAfter(DateTime.now())) {  
 print('Data has not expired.');  
 // The data has not expired.  
 return data;  
 } else {  
 // Data has expired. Remove it from SharedPreferences.  
 await prefs.remove(*\_keyData*);  
 await prefs.remove(*\_keyExpiration*);  
 print('Data has expired. Removed from SharedPreferences.');  
 return null;  
 }  
 } catch (e) {  
 print('Error retrieving data from SharedPreferences: $e');  
 return null;  
 }  
 }  
  
 // Function to clear data from SharedPreferences  
 Future<void> clearData() async {  
 try {  
 SharedPreferences prefs = await SharedPreferences.*getInstance*();  
 await prefs.remove(*\_keyData*);  
 await prefs.remove(*\_keyExpiration*);  
 print('Data cleared from SharedPreferences.');  
 } catch (e) {  
 print('Error clearing data from SharedPreferences: $e');  
 }  
 }  
}

Страница пользователя (загружаемая после успешной авторизации и регистрации) содержит в себе виджет с несколькими состояниями – которые переключаются с использованием визуального элемента BottomNavigationBar – на каждом из выбираемом состоянии виджета находится свой контент (для работы с мероприятиями, группами, задачами или отчетам пользователя, соответственно)

Листинг виджета user\_page.dart:

class UserPage extends StatelessWidget {  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 theme: ThemeData(  
 primarySwatch: Colors.*lightGreen*,  
 ),  
 home: Home(),  
 );  
 }  
}  
  
class Home extends StatefulWidget {  
  
 @override  
 State<Home> createState() => \_HomeState();  
}  
  
class \_HomeState extends State<Home> {  
 int \_currentIndex = 0;  
  
 @override  
 void initState(){  
 super.initState();  
 }  
  
 final List<Widget> \_children = [  
 EventPlaceholderWidget(  
 color: Colors.*red*,  
 text: 'Главная страница пользователя',  
 index: 0  
 ),  
  
 EventPlaceholderWidget(  
 color: Colors.*green*,  
 text: 'Страница создания мероприятия',  
 index: 1),  
  
 GroupPlaceholderWidget(  
 color: Colors.*blueAccent*,  
 text: 'Страница создания новой группы',  
 index: 2),  
  
 TaskPlaceholderWidget(  
 color: Colors.*lime*,  
 text: 'Страница создания новой задачи',  
 index: 3),  
  
 ReportPlaceholderWidget(  
 color: Colors.*deepOrangeAccent*,  
 text: 'Страница создания нового отчета',  
 index: 4)  
 ];  
  
 void onTabTapped(int index) {  
 setState(() {  
 \_currentIndex = index;  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Личный кабинет календаря пользователя'),  
 ),  
 body: \_children[\_currentIndex],  
 bottomNavigationBar: BottomNavigationBar(  
 type: BottomNavigationBarType.fixed,  
 backgroundColor: Colors.*lightBlue*,  
 selectedItemColor: Colors.*greenAccent*,  
 unselectedItemColor: Colors.*grey*,  
 currentIndex: \_currentIndex,  
 onTap: onTabTapped,  
 items: [  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*toc*),  
 label: 'Главная страница',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_alarm*),  
 label: 'Создать новое мероприятие',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_business\_outlined*),  
 label: 'Создать новую группу',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_task*),  
 label: 'Создать новую задачу',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_comment\_rounded*),  
 label: 'Создать новый отчет',  
 ),  
 ],  
 ),  
 );  
 }  
}

Листинг виджета для работы с мероприятиями пользователя:

class EventPlaceholderWidget extends StatelessWidget {  
 final Color color;  
 final String text;  
 final int index;  
  
 final int groupId = 10;  
 late String token;  
  
 final TextEditingController eventCaptionController = TextEditingController();  
 final TextEditingController eventDescriptionController = TextEditingController();  
 final TextEditingController scheduledStartController = TextEditingController();  
 final TextEditingController durationController = TextEditingController();  
 final TextEditingController eventTypeController = TextEditingController();  
 final TextEditingController eventStatusController = TextEditingController();  
  
 EventPlaceholderWidget(  
 {  
 required this.color,  
 required this.text,  
 required this.index  
 });  
  
 Future<void> addNewEvent(BuildContext context) async  
 {  
 String caption = eventCaptionController.text;  
 String description = eventDescriptionController.text;  
 String scheduledStart = scheduledStartController.text;  
 String duration = durationController.text;  
 String eventType = eventTypeController.text;  
 String eventStatus = eventStatusController.text;  
  
 var guestIds = [2];  
  
 MySharedPreferences mySharedPreferences = new MySharedPreferences();  
  
 var cachedData = await mySharedPreferences.getDataIfNotExpired();  
  
 if (cachedData != null){  
 var json = jsonDecode(cachedData.toString());  
 var cacheContent = ResponseWithToken.fromJson(json);  
  
 var userId = cacheContent.userId;  
 var token = cacheContent.token.toString();  
  
 var model = new AddNewEventModel(  
 userId: (userId),  
 token: token,  
 caption: caption,  
 description: description,  
 start: scheduledStart,  
 duration: duration,  
 eventType: eventType,  
 eventStatus: eventStatus,  
 groupId: groupId,  
 guestIds: guestIds);  
  
 var requestMap = model.toJson();  
  
 final url = Uri.*parse*('http://localhost:5201/events/schedule\_new');  
 final headers = {'Content-Type': 'application/json'};  
 final body = jsonEncode(requestMap);  
 final response = await http.post(url, headers: headers, body: body);  
  
 var jsonData = jsonDecode(response.body);  
 var responseContent = Response.fromJson(jsonData);  
 if (responseContent.result){  
 if (responseContent.outInfo != null){  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(  
 content: Text(responseContent.outInfo.toString())  
 )  
 );  
 }  
 }  
 }  
 else {  
 showDialog(  
 context: context,  
 builder: (context) => AlertDialog(  
 title: Text('Ошибка!'),  
 content: Text('Создание нового мероприятия не произошло!'),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.*pop*(context);  
 },  
 child: Text('OK'),  
 ),  
 ],  
 ),  
 );  
 }  
  
 eventCaptionController.clear();  
 eventDescriptionController.clear();  
 scheduledStartController.clear();  
 durationController.clear();  
 eventTypeController.clear();  
 eventStatusController.clear();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Padding(  
 padding: EdgeInsets.all(16.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 text,  
 style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(height: 30.0),  
 if(index == 0) ...[  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(36.0)),  
 minimumSize: Size(150, 60),  
 ),  
 onPressed: () {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => UserInfoMapPage()),);  
 },  
 child: Text('Перейти к вашему личному кабинету'),  
 ),  
 SizedBox(height: 30.0),  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(36.0)),  
 minimumSize: Size(150, 60),  
 ),  
 onPressed: () {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => AdditionalPageWidget()),);  
 },  
 child: Text('Страничка с картинками'),  
 ),  
 SizedBox(height: 30.0),  
 ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*green*,  
 foregroundColor : Colors.*white*,  
 shadowColor: Colors.*greenAccent*,  
 elevation: 3,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(36.0)),  
 minimumSize: Size(150, 60),  
 ),  
 onPressed: () {  
 MySharedPreferences mySharedPreferences = new MySharedPreferences();  
 mySharedPreferences.clearData();  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => HomePage()),);  
 },  
 child: Text('Выйти'),  
 ),  
 ],  
 if(index == 1) ...[  
 SizedBox(height: 8.0),  
 TextField(  
 controller: eventCaptionController,  
 decoration: InputDecoration(  
 labelText: 'Наименование мероприятия:',  
 ),  
 ),  
 SizedBox(height: 8.0),  
 TextFormField(  
 controller: eventDescriptionController,  
 maxLines: null,  
 decoration: InputDecoration(  
 labelText: 'Описание меропрития:',  
 ),  
 ),  
 SizedBox(height: 8.0),  
 TextField(  
 controller: scheduledStartController,  
 decoration: InputDecoration(  
 labelText: 'Время начала мероприятия',  
 ),  
 ),  
 SizedBox(height: 8.0),  
 TextField(  
 controller: eventTypeController,  
 decoration: InputDecoration(  
 labelText: 'Тип мероприятия',  
 ),  
 ),  
 SizedBox(height: 8.0),  
 TextField(  
 controller: eventStatusController,  
 decoration: InputDecoration(  
 labelText: 'Статус мероприятия',  
 ),  
 ),  
 ],  
 if(index == 1) ...[  
 SizedBox(height: 16.0),  
 ElevatedButton(  
 onPressed: () {  
 addNewEvent(context);  
 },  
 child: Text('Создать новое мероприятие'),  
 ),  
 ],  
 ]  
 ),  
 );  
 }  
}

По аналогии, есть виджеты для работы с группами, задачами и отчетами.

Также присутствует инструмент как бы личного кабинета – для просмотра информации о текущих мероприятий в календаре пользователя, списка его групп, задач и отчетов.

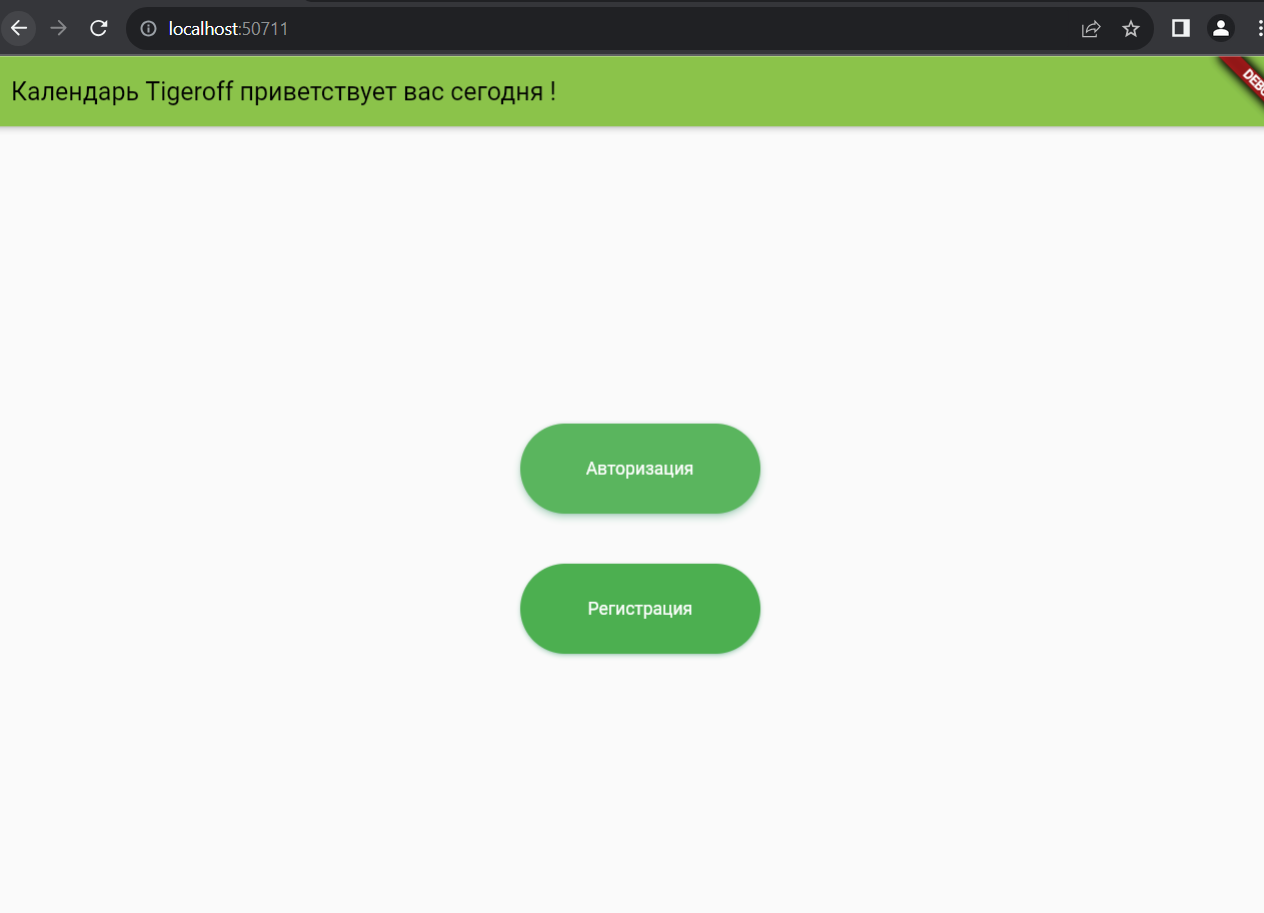
Листинг виджета личного кабинета user\_info\_map.dart:

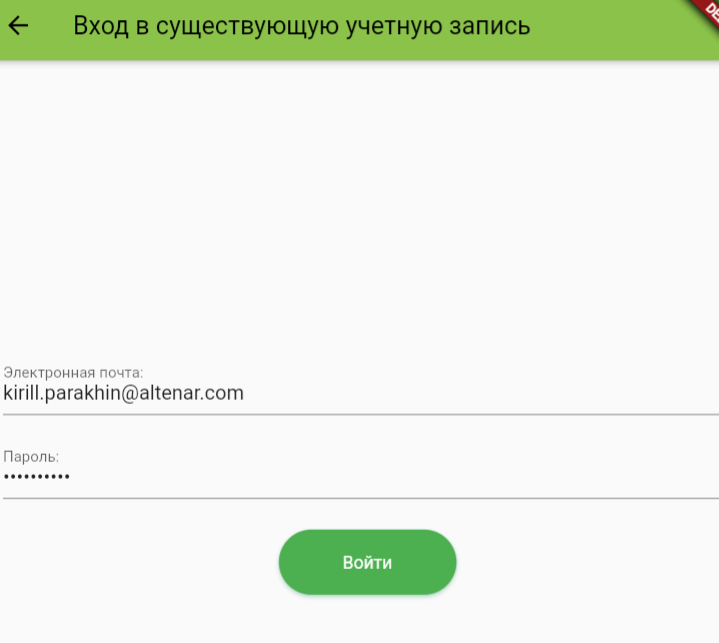
class UserInfoMapPage extends StatelessWidget {  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 theme: ThemeData(  
 primarySwatch: Colors.*lightGreen*,  
 ),  
 home: HomeMap(),  
 );  
 }  
}  
  
class HomeMap extends StatefulWidget {  
  
 @override  
 State<HomeMap> createState() => \_HomeState();  
}  
  
class \_HomeState extends State<HomeMap> {  
 int \_currentIndex = 0;  
  
 @override  
 void initState(){  
 super.initState();  
 }  
  
 final List<Widget> \_children = [  
 EventsListPageWidget(),  
 GroupsListPageWidget(),  
 TasksListPageWidget(),  
 ReportsListPageWidget()  
 ];  
  
 void onTabTapped(int index) {  
 setState(() {  
 \_currentIndex = index;  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Личный кабинет пользователя'),  
 ),  
 body: \_children[\_currentIndex],  
 bottomNavigationBar: BottomNavigationBar(  
 type: BottomNavigationBarType.fixed,  
 backgroundColor: Colors.*lightBlue*,  
 selectedItemColor: Colors.*greenAccent*,  
 unselectedItemColor: Colors.*grey*,  
 currentIndex: \_currentIndex,  
 onTap: onTabTapped,  
 items: [  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_alarm*),  
 label: 'Мои мероприятия',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_business\_outlined*),  
 label: 'Мои группы',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_task*),  
 label: 'Мои задачи на реализацию',  
 ),  
 BottomNavigationBarItem(  
 icon: Icon(Icons.*add\_comment\_rounded*),  
 label: 'Мои отчеты',  
 ),  
 ],  
 ),  
 );  
 }  
}

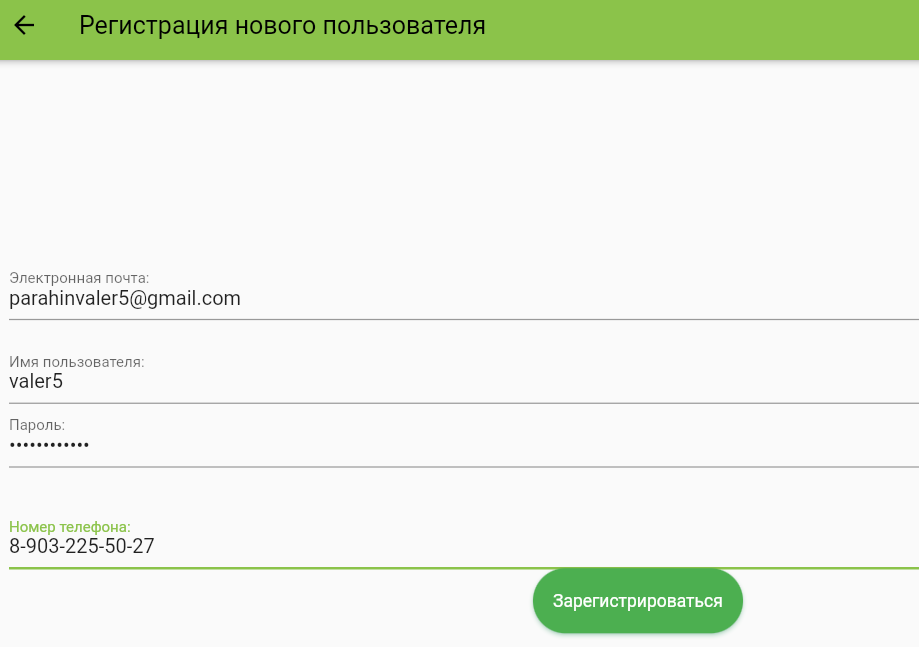
Листинг страницы с календарем пользователя (списком его мероприятий) – events\_list\_page.dart:

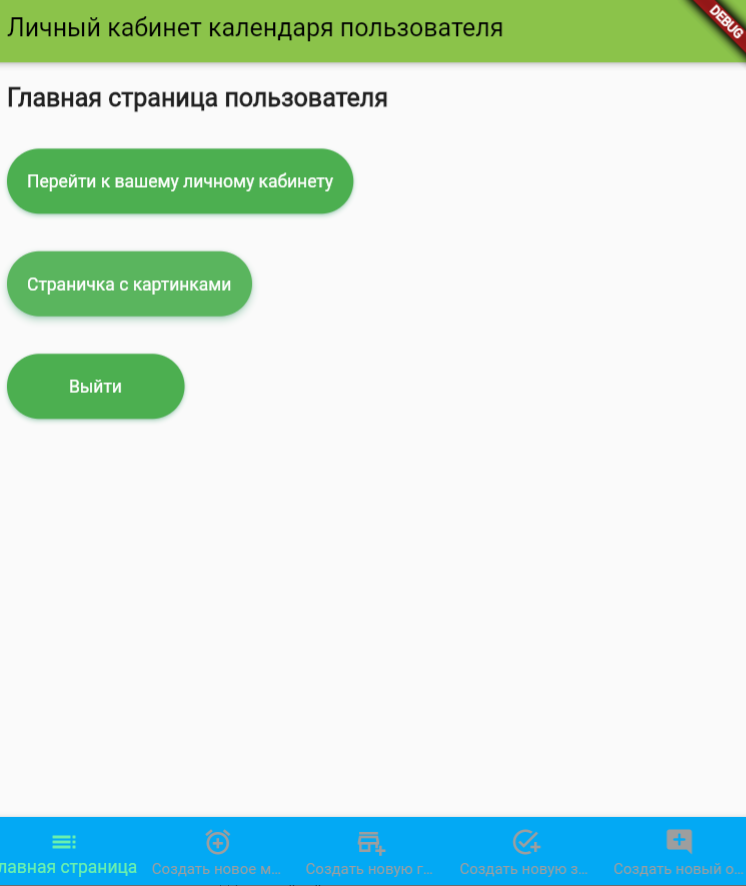
class EventsListPageWidget extends StatefulWidget {  
  
 @override  
 EventsListPageState createState() => EventsListPageState();  
}  
  
class EventsListPageState extends State<EventsListPageWidget> {  
  
 @override  
 void initState() {  
 super.initState();  
 getUserInfo();  
 }  
  
 final uri = 'http://localhost:5201/users/get\_info';  
 final headers = {'Content-Type': 'application/json'};  
 bool isColor = false;  
  
 final EnumAliaser aliaser = new EnumAliaser();  
  
 List<EventInfoResponse> eventsList = [];  
  
 Future<void> getUserInfo() async {  
  
 MySharedPreferences mySharedPreferences = new MySharedPreferences();  
  
 var cachedData = await mySharedPreferences.getDataIfNotExpired();  
  
 if (cachedData != null){  
 var json = jsonDecode(cachedData.toString());  
 var cacheContent = ResponseWithToken.fromJson(json);  
  
 var userId = cacheContent.userId;  
 var token = cacheContent.token.toString();  
  
 var model = new UserInfoRequestModel(userId: userId, token: token);  
 var requestMap = model.toJson();  
  
 var url = Uri.*parse*(uri);  
 final body = jsonEncode(requestMap);  
  
 final response = await http.post(url, headers: headers, body: body);  
  
 var jsonData = jsonDecode(response.body);  
 var responseContent = GetResponse.fromJson(jsonData);  
  
 if (responseContent.result) {  
 var userRequestedInfo = responseContent.requestedInfo.toString();  
  
 var data = jsonDecode(userRequestedInfo);  
 var userEvents = data['user\_events'];  
  
 var fetchedEvents =  
 List<EventInfoResponse>  
 .from(userEvents.map(  
 (data) => EventInfoResponse.fromJson(data)));  
  
 setState(() {  
 eventsList = fetchedEvents;  
 });  
 }  
 }  
 else {  
 setState(() {  
 showDialog(  
 context: context,  
 builder: (context) => AlertDialog(  
 title: Text('Ошибка!'),  
 content:  
 Text(  
 'Произошла ошибка при получении'  
 ' полной информации о пользователе!'),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.*pop*(context);  
 },  
 child: Text('OK'),  
 ),  
 ],  
 ),  
 );  
 });  
 }  
 }  
  
 List<Appointment> getAppointments(List<EventInfoResponse> fetchedEvents){  
 MaterialColor color = Colors.*blue*;  
  
 List<EventAppointment> meetings =  
 List.from(  
 fetchedEvents.map((data) =>  
 new EventAppointment(  
 data.start,  
 data.duration,  
 data.caption)));  
  
 List<Appointment> appointments =  
 List.from(  
 meetings.map((data) =>  
 new Appointment(  
 startTime: data.startTime,  
 endTime: data.endTime,  
 subject: data.subject,  
 color: color)));  
  
 return appointments;  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text('Календарь мероприятий'),  
 leading: IconButton(  
 icon: Icon(Icons.*arrow\_back*),  
 onPressed: () {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => UserPage()),);  
 },  
 ),  
 ),  
 body: SfCalendar(  
 view: CalendarView.week,  
 firstDayOfWeek: 1,  
 initialDisplayDate: DateTime.now(),  
 initialSelectedDate: DateTime.now(),  
 dataSource: MeetingDataSource(getAppointments(eventsList)),  
 ),  
 ),  
 /\*  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text('Список мероприятий'),  
 leading: IconButton(  
 icon: Icon(Icons.arrow\_back),  
 onPressed: () {  
 Navigator.pushReplacement(  
 context,  
 MaterialPageRoute(  
 builder: (context) => UserPage()),);  
 },  
 ),  
 ),  
 body: ListView.builder(  
 itemCount: eventsList.length,  
 itemBuilder: (context, index) {  
 final data = eventsList[index];  
 return Card(  
 color: isColor ? Colors.red : Colors.teal,  
 elevation: 15,  
 child: InkWell(  
 onTap: () {  
 setState(() {  
 isColor = !isColor;  
 });  
 },  
 child: Padding(  
 padding: EdgeInsets.all(25),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 'Название мероприятия: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 utf8.decode(data.caption.codeUnits),  
 style: TextStyle(  
 color: Colors.white,  
 fontWeight: FontWeight.bold,  
 ),  
 ),  
 SizedBox(height: 8),  
 Text(  
 'Описание мероприятия: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 utf8.decode(data.description.codeUnits),  
 style: TextStyle(  
 color: Colors.white,  
 fontWeight: FontWeight.bold,  
 ),  
 ),  
 SizedBox(height: 8),  
 Text(  
 'Время начала мероприятия: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 data.start.toString(),  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 SizedBox(height: 8),  
 Text(  
 'Продолжительность мероприятия: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 data.duration.toString(),  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 SizedBox(height: 8),  
 Text(  
 'Тип события: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 aliaser.GetAlias(  
 aliaser.getEventTypeEnumValue(data.eventType)),  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 SizedBox(height: 8),  
 Text(  
 'Статус события: ',  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 Text(  
 aliaser.GetAlias(  
 aliaser.getEventStatusEnumValue(data.eventStatus)),  
 style: TextStyle(  
 color: Colors.white,  
 ),  
 ),  
 SizedBox(height: 8),  
 ElevatedButton(  
 child: Text('Перейти к мероприятию'),  
 onPressed: () {},  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 },  
 ),  
 ),  
 \*/  
 );  
 }  
}

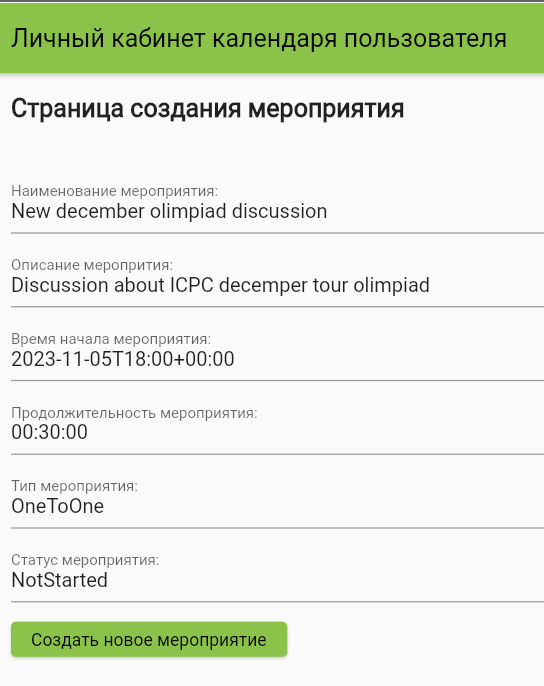
Тестирование приложения:

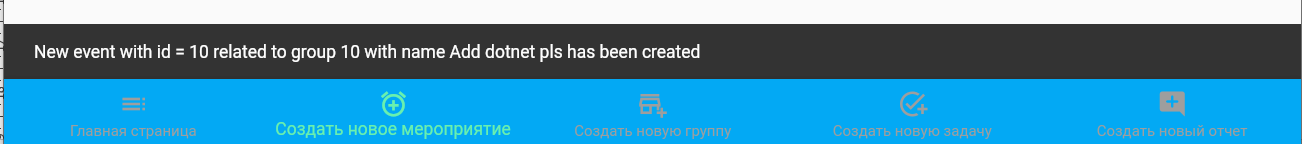


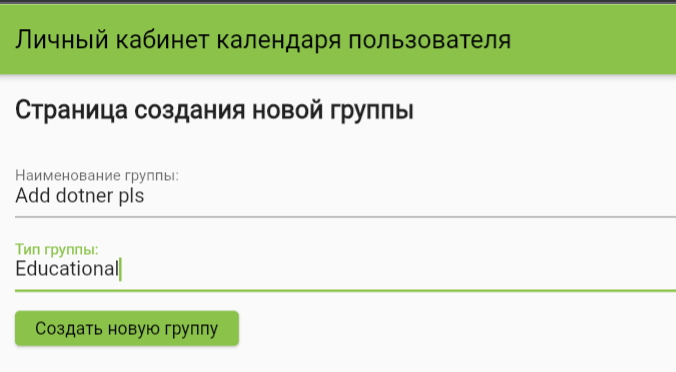


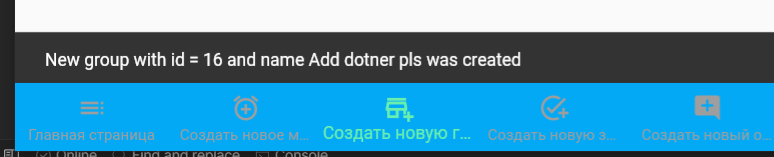


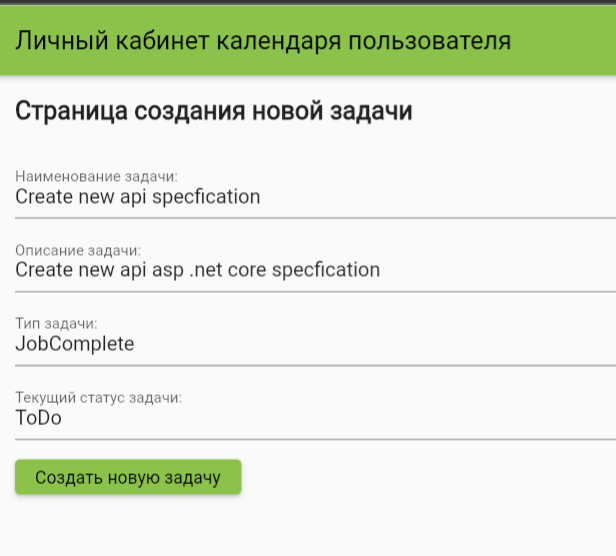


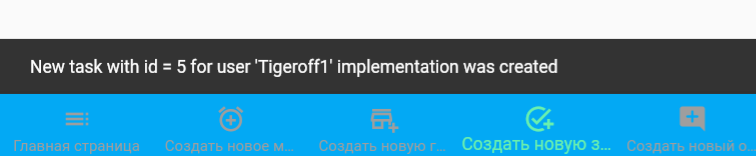


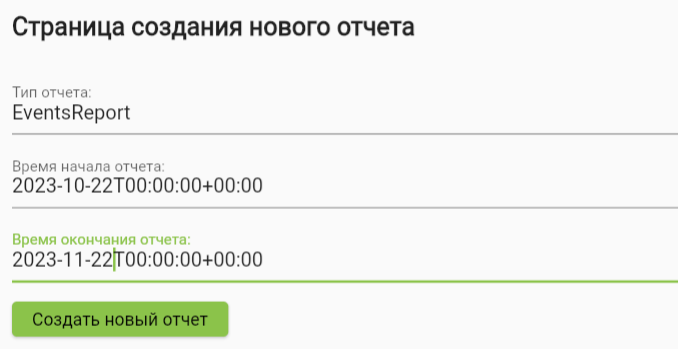


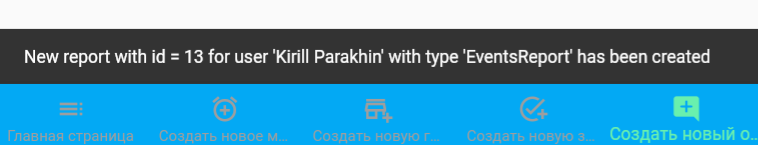


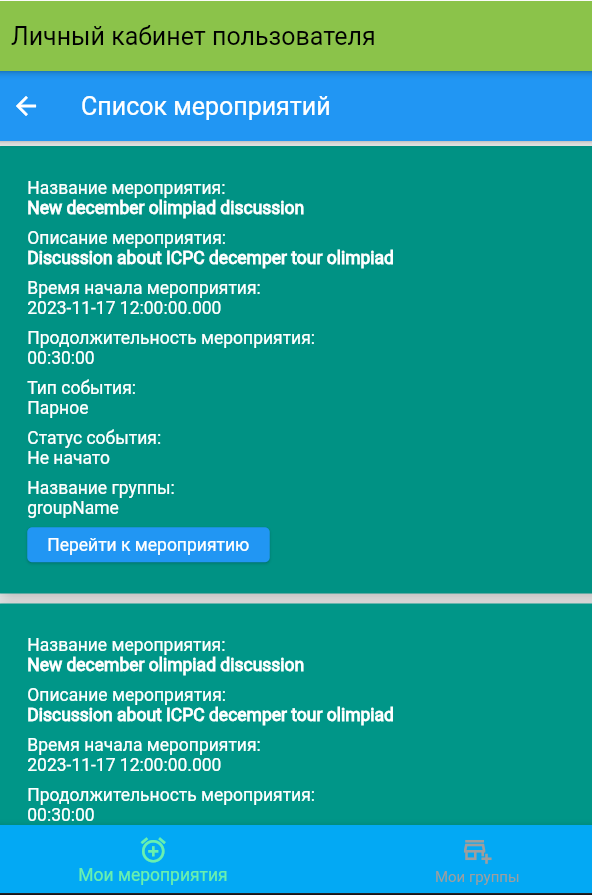


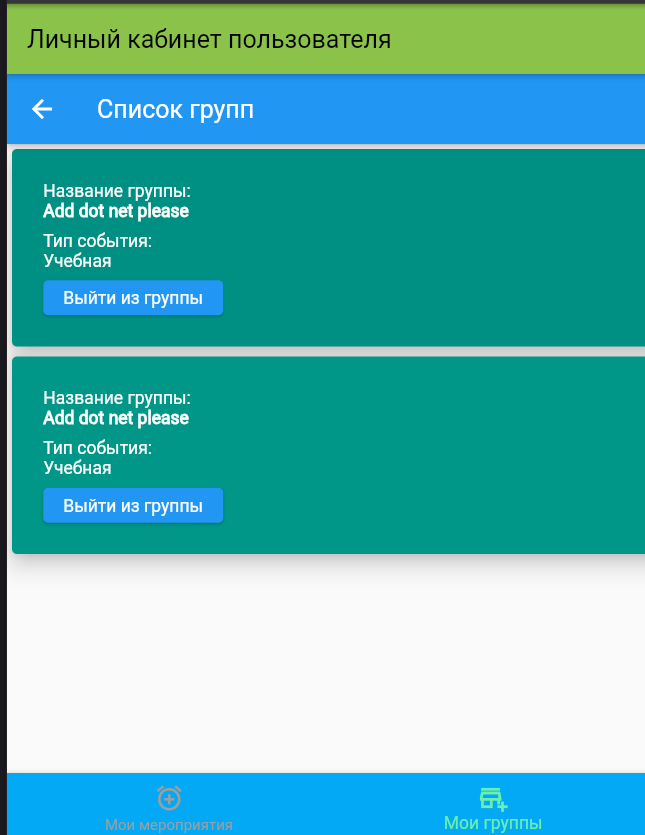




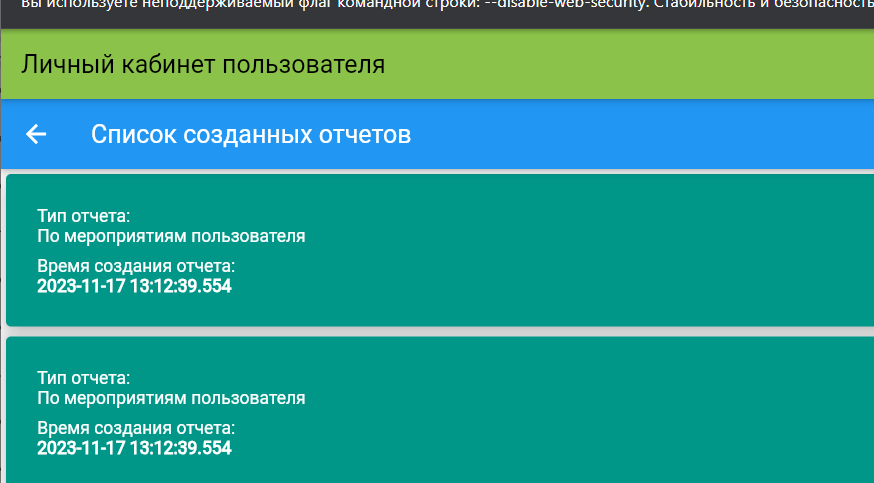












Вывод

В результате выполнения работы я разработал клиент-серверное приложения для операционной системы Android на фреймворе Flutter.