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 №11

по дисциплине "Технологии разработки мобильных приложений"

ТЕМА РАБОТЫ:

Знакомство с Flutter

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

преподаватель кафедры ИСПИ

Рощина А.И.

Владимир 2023 г.

Цель работы:

Реализовать приложение с использованием фреймворка Flutter.

Выполнение работы

Реализация NavigationBar:

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',  
 ),  
 ],  
 ),  
 );  
 }  
}

Реализация отображения списка cобытий из календаря (с их названием, описанием, временем начала и продолжительностью):

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('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: () {},  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 },  
 ),  
 ),  
 );  
 }  
}

Реализация добавления нового элемента в список мероприятий:

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: durationController,  
 decoration: InputDecoration(  
 labelText: 'Продолжительность 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'),  
 ),  
 ],  
 ]  
 ),  
 );  
 }  
}

Реализация смены цвета фона у элементов списка:

return Card(  
 color: isColor ? Colors.*red* : Colors.*teal*,  
 elevation: 15,  
 child: InkWell(  
 onTap: () {  
 setState(() {  
 isColor = !isColor;  
 });  
 },

Реализация смены изображения по клику на него (и отдельно смена изображения автоматически каждые 3 секунды – в отдельном методе):

class AdditionalPageWidget extends StatefulWidget {  
  
 @override  
 \_AdditionalPageState createState() => \_AdditionalPageState();  
}  
  
class \_AdditionalPageState extends State<AdditionalPageWidget> {  
 String pictureUrl = "https://source.unsplash.com/random/800x600";

@override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Список мероприятий'),  
 leading: IconButton(  
 icon: Icon(Icons.*arrow\_back*),  
 onPressed: () {  
 Navigator.*pushReplacement*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => UserPage()),);  
 },  
 ),  
 ),  
 body: Scaffold(  
 body: Center(  
 child: GestureDetector(  
 onTap: changeURLByClick,  
 child: Image.network(pictureUrl),  
 ),  
 ),  
 )  
 );  
 }  
  
 void changeURLByClick() {  
 setState(() {  
 pictureUrl = "https://source.unsplash.com/random/800x600/?" +  
 "q=${new DateTime.now().millisecondsSinceEpoch}";  
 });  
 }  
  
  
 void changeURLByTimer() {  
 Timer.periodic(Duration(seconds: 3), (timer) {  
 setState(() {  
 pictureUrl = "https://source.unsplash.com/random/800x600/?" +  
 "q=${new DateTime.now().millisecondsSinceEpoch}";  
 });  
 });  
 }  
}

Реализация смены изображения автоматически раз в 3 секунды:

void changeURLByTimer() {  
 Timer.periodic(Duration(seconds: 3), (timer) {  
 setState(() {  
 pictureUrl = "https://source.unsplash.com/random/800x600/?" +  
 "q=${new DateTime.now().millisecondsSinceEpoch}";  
 });  
 });  
}

Тестирование приложения:

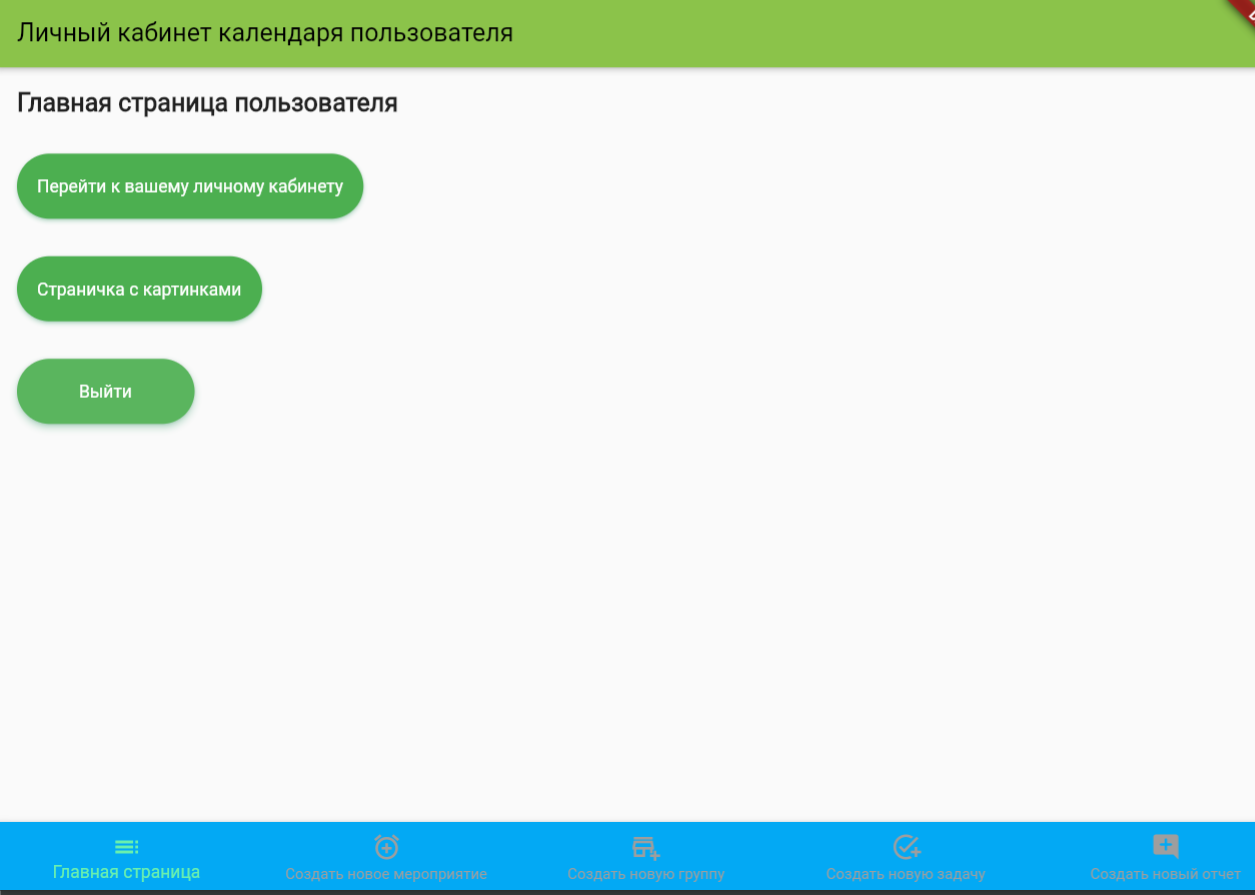


Рисунок 1. Виджет с использованием элементов navigation bar

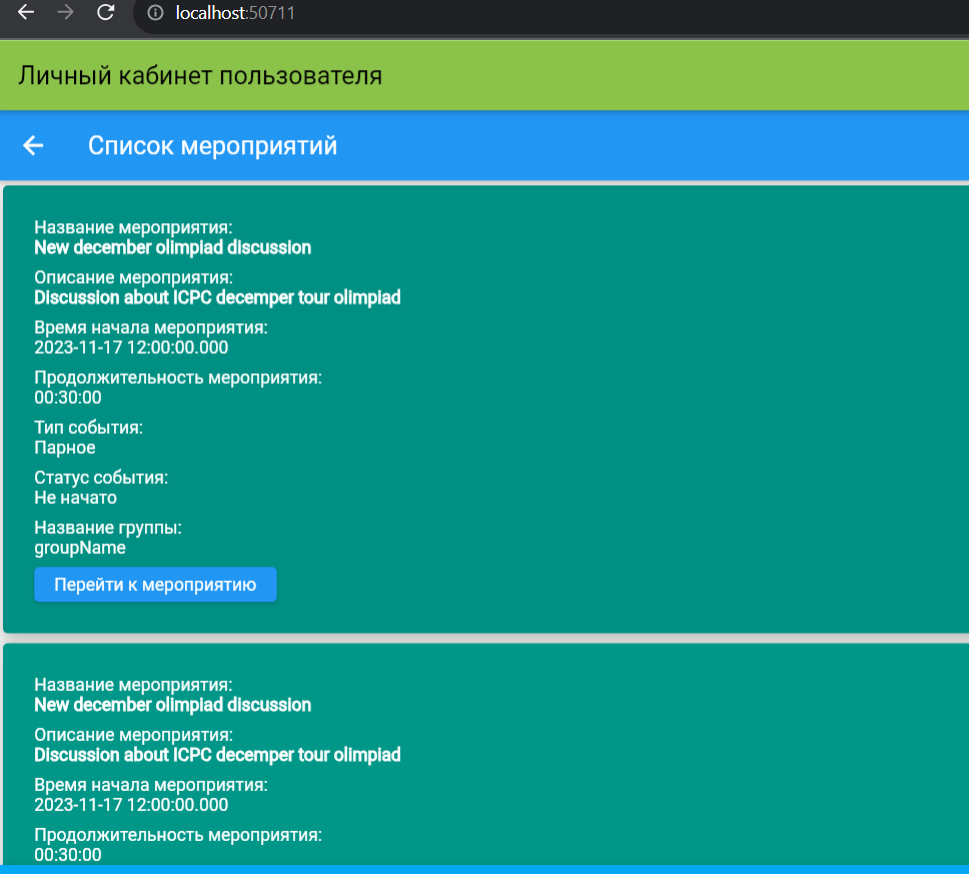


Рисунок 2. Виджет со списком элементов мероприятий

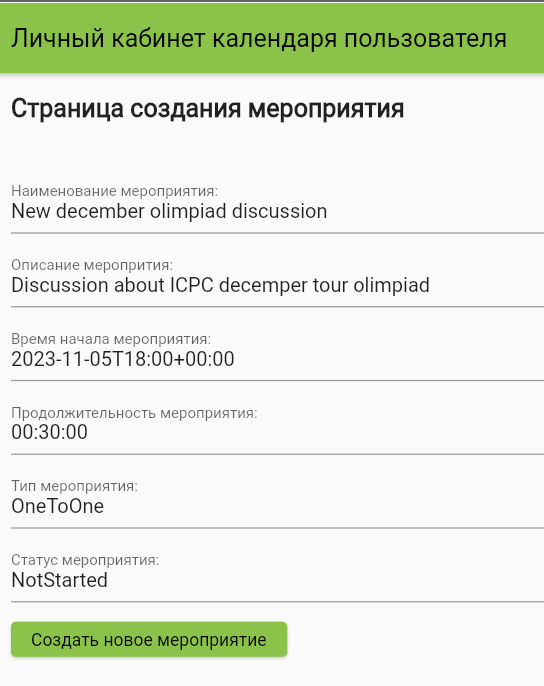
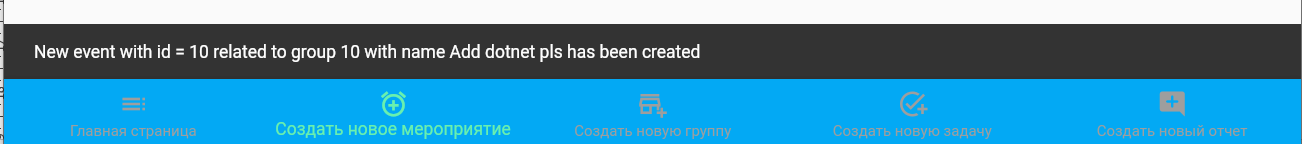


Рисунок 3.1 – 3.2 Добавление нового элемента в список



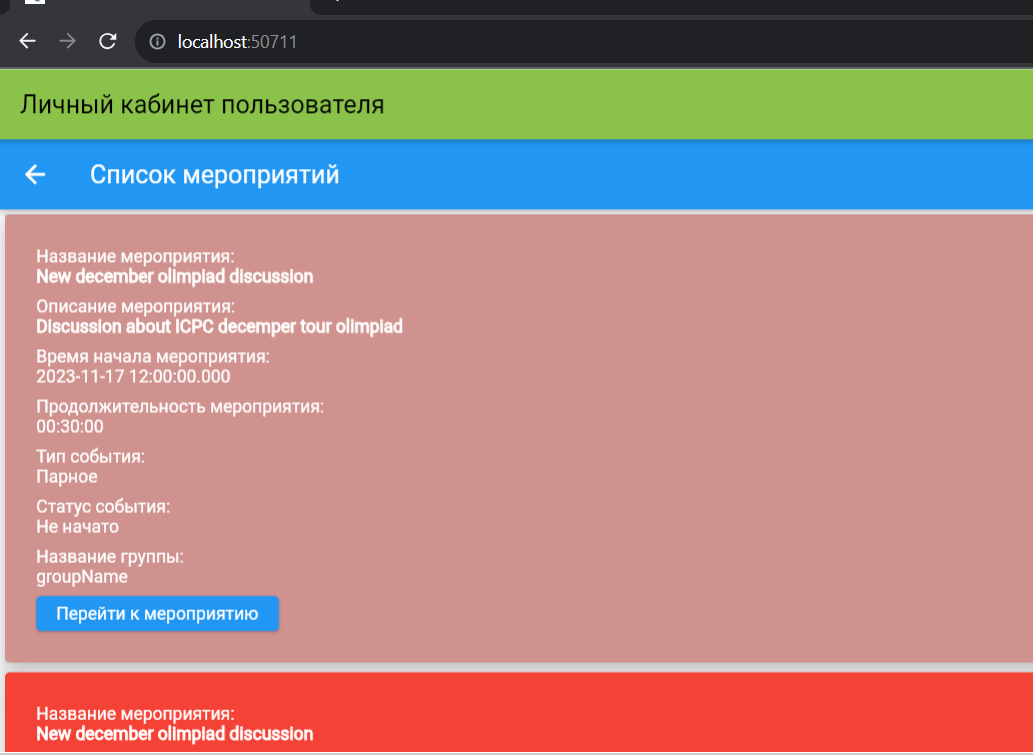
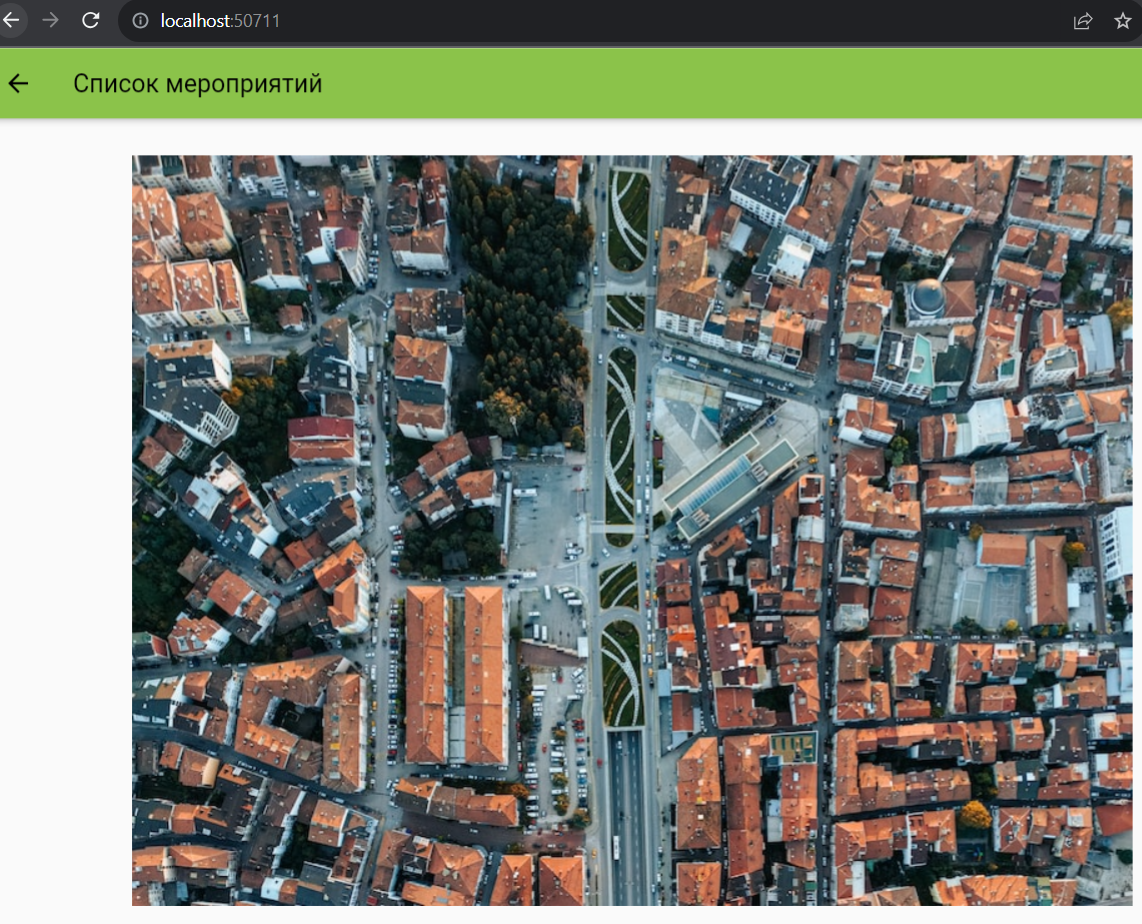
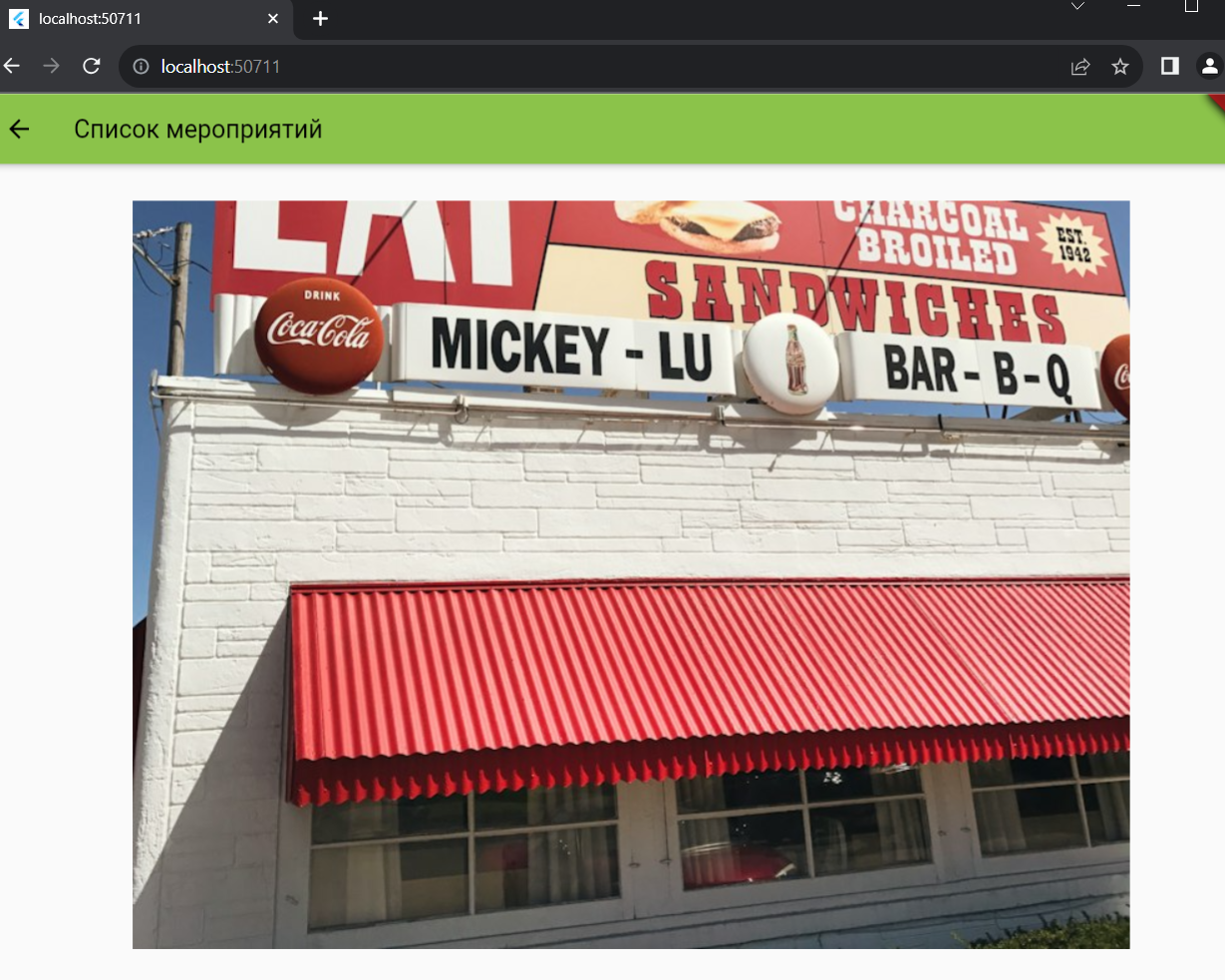


Рисунок 4. Смена цвета элемента списка при клике





Рисунки 5.1 – 5.2. Смена изображений на виджете автоматически и по нажатию

Вывод:

В результате выполнения работы, я реализовал приложение с использованием фреймворка Flutter.