Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего 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 моильных приложений»**

**Тема: Разработка клиент-серверного приложения**

Выполнила:

ст. гр. ПРИ-117

А.И. Рощина

Принял:

Козаев А.Т.

Владимир, 2020

1. ЦЕЛЬ РАБОТЫ

Реализовать программную систему с клиент-серверным взаимодействием.

1. ВЫПОЛНЕНИЕ РАБОТЫ

Репозиторий проекта:

<https://github.com/flamebrier/calibration-front>

***Задания***

1. регистрация и авторизация в системе;

2.просмотр/управление доступными сущностями для пользователей;

3. управление своей учетной записью для пользователей;

5. 2 индивидуальные функции

***Ход работы***

1. Код регистрации/авторизации

final FirebaseAuth \_auth = FirebaseAuth.instance;

final GoogleSignIn googleSignIn = GoogleSignIn();

Future<String> signInWithGoogle() async {

await Firebase.initializeApp();

final GoogleSignInAccount googleSignInAccount = await googleSignIn.signIn();

final GoogleSignInAuthentication googleSignInAuthentication =

await googleSignInAccount.authentication;

final AuthCredential credential = GoogleAuthProvider.credential(

accessToken: googleSignInAuthentication.accessToken,

idToken: googleSignInAuthentication.idToken,

);

final UserCredential authResult =

await \_auth.signInWithCredential(credential);

final User user = authResult.user;

if (user != null) {

assert(!user.isAnonymous);

assert(await user.getIdToken() != null);

final User currentUser = \_auth.currentUser;

assert(user.uid == currentUser.uid);

print('signInWithGoogle succeeded: $user');

return '$user';

}

return null;

}

void signOutGoogle() async {

await googleSignIn.signOut();

print("User Signed Out");

}

1. Код работы с запросами по http

import 'dart:convert';

import 'dart:developer';

import 'dart:io';

import 'package:calibration/data/models.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:http/http.dart' as http;

class Loader {

static final instance = Loader.\_();

Loader.\_() {

FirebaseAuth.instance?.authStateChanges()?.listen((user) async {

userToken = await user?.getIdToken();

});

}

init() async {

userToken = await FirebaseAuth.instance?.currentUser?.getIdToken();

}

Profile currentProfile;

String userToken;

Map<String, String> get headers => {

HttpHeaders.contentTypeHeader: 'application/json',

HttpHeaders.authorizationHeader: 'Bearer $userToken'

};

Future<http.Response> getUser() async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/users/getprofile');

final response = await http.get(uri, headers: headers);

return response;

}

Future<http.Response> getCategoryStats() async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/users/statistics');

final response = await http.get(uri);

return response;

}

Future<http.Response> setUserNick(String newNick) async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/users/changenick');

var params = {"value": newNick};

final response =

await http.post(uri, headers: headers, body: jsonEncode(params));

return response;

}

Future<http.Response> getCategories() async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/quiz/categories');

final response = await http.get(uri);

return response;

}

Future<http.Response> getQuizById() async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/quiz/2');

final response = await http.get(uri, headers: headers);

return response;

}

Future<http.Response> getQuizes(int catId) async {

Uri uri = Uri.http('10.0.2.2:5000', '/api/quiz/bycategory/$catId');

final response = await http.get(uri);

return response;

}

Future<http.Response> createCategory(String name) async {

var params = {"BaseCategoryId": 0, "Name": name};

Uri uri = Uri.http('10.0.2.2:5000', '/api/quizgenerator/addcategory');

final response =

await http.post(uri, headers: headers, body: jsonEncode(params));

return response;

}

Future<http.Response> createQuiz(int catId, FilterTypes ft) async {

var params = {

"QuizCategoryId": catId,

"FilterType": ft.index,

"TopFilter": null

};

// var params = {"QuizCategoryId": 1, "FilterType": 1, "TopFilter": null};

log(params.toString());

Uri uri = Uri.http('10.0.2.2:5000', '/api/quizgenerator/reddit');

final response =

await http.post(uri, headers: headers, body: jsonEncode(params));

log(response.body.toString());

return response;

}

Future<http.Response> createSession(

StartSettings settings, int quizId) async {

var params = {

"maxPlayersCount": settings.maxPeople ?? 2,

"gameSessionTimeInMinutes": settings.maxTime.inMinutes ?? 10,

"countOfElementsToChoose": settings.goodsCounter ?? 10

};

Uri uri = Uri.http('10.0.2.2:5000', '/api/session/create/$quizId');

final response =

await http.post(uri, headers: headers, body: jsonEncode(params));

return response;

}

Future<http.Response> joinSession(int sessionId) async {

var params = {"Value": sessionId};

Uri uri = Uri.http('10.0.2.2:5000', '/api/session/join');

final response =

await http.post(uri, headers: headers, body: jsonEncode(params));

return response;

}

}

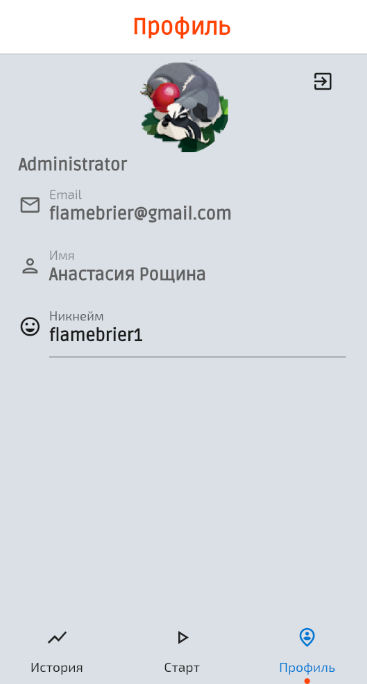
****

Рисунок 1 - Редактирование профиля

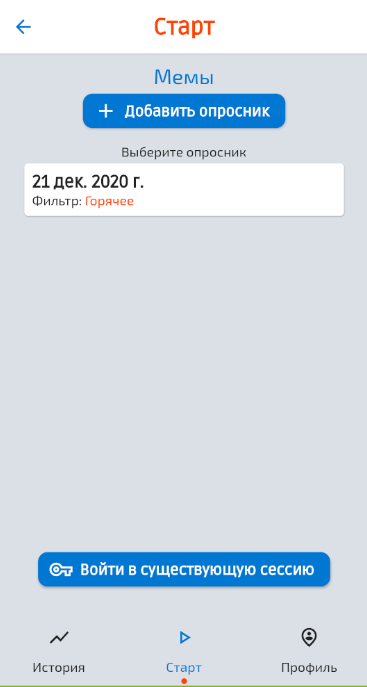
****

Рисунок 2 - Отправка данных на сервер для запроса изображений

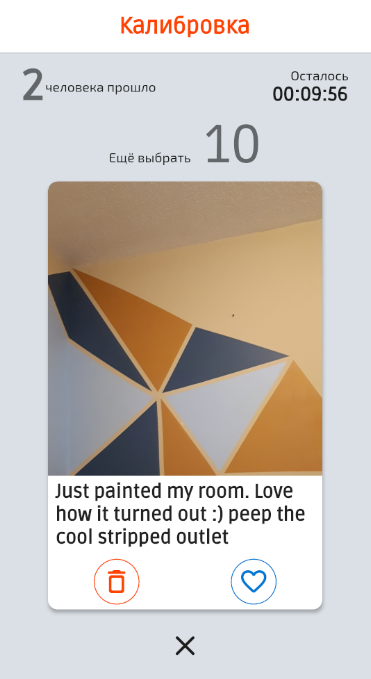
****

Рисунок 3 - Получение изображений и текста с сервера

1. ВЫВОД

В процессе выполнения лабораторной работы была реализована программная система с клиент-серверным взаимодействием