Документація по застосунку «WhatUCread»

**Назва застосунку:** WhatUCread

**Опис:** Даний застосунок створений в першу чергу для мобільного користування (Android і iOS), але ним також можна користуватися через браузер. Користувач має можливість переглядати для себе книги та різні твори. Також, можна такі твори читати або додавати до улюблених, зберігати в себе в акаунті.

**Функції:**

* Реєстрація нового користувача через пошту.

Опис: Містить методи для створення нового користувача через базу даних Firebase та функцію аутентифікації.

Код:

Future<String> signUpUser(String email, String password, String userName) async{

    String retVal = "error";

    UserModel \_user = UserModel();

    try{

      //await \_auth.createUserWithEmailAndPassword(email: email, password: password);

      UserCredential \_authResult =

              await \_auth.createUserWithEmailAndPassword(email: email, password: password);

      \_user.uid = \_authResult.user!.uid;

      \_user.email = \_authResult.user!.email;

      \_user.userName = userName;

      String \_returnString = await Database().createUser(\_user);

      if(\_returnString == "success"){

        retVal = "success";

      }

      //retVal = "success";

    } on PlatformException catch(e){

      retVal = e.message!;

    } catch(e){

      print(e);

    }

    return retVal;

  }

* Вхід до існуючого користувача через пошту

Опис: : Містить методи для входу до існуючого користувача в реальному часі та використання його даних через Firebase.

Код:

Future<String> loginUserWithEmail(String email, String password) async{

    String retVal = "error";

    try{

      UserCredential \_authResult = await \_auth.signInWithEmailAndPassword(email: email, password: password);

      \_currentUser = await Database().getUserInfo(\_authResult.user!.uid);

      if(\_currentUser != null){

        retVal = "success";

      }

    } on PlatformException catch(e){

      retVal = e.message!;

    } catch(e) {

      print(e);

    }

    return retVal;

  }

* Вхід через google акаунт

Опис: Використовує запис з сервісу google та автоматично налаштовує інформацію про користувач(ім’я, час реєстрації, ідентифікаційний номер, тощо)

Код:

Future<String> loginUserWithGoogle() async{

    String retVal = "error";

    const List<String> scopes = <String>[

      'email',

      'https://www.googleapis.com/auth/contacts.readonly',

    ];

    GoogleSignIn \_googleSignIn = GoogleSignIn(

      // Optional clientId

      clientId: 'your-client\_id.apps.googleusercontent.com',

      scopes: scopes,

    );

    UserModel \_user = UserModel();

    try{

      GoogleSignInAccount? \_googleUser = await \_googleSignIn.signIn();

      GoogleSignInAuthentication \_googleAuth = await \_googleUser!.authentication;

      final AuthCredential credential = GoogleAuthProvider.credential(idToken: \_googleAuth.idToken, accessToken: \_googleAuth.accessToken);

      UserCredential \_authResult = await \_auth.signInWithCredential(credential);

      if(\_authResult.additionalUserInfo!.isNewUser){

        \_user.uid = \_authResult.user!.uid;

        \_user.email = \_authResult.user!.email;

        \_user.userName = \_authResult.user!.displayName;

        Database().createUser(\_user);

      }

      \_currentUser = await Database().getUserInfo(\_authResult.user!.uid);

      if(\_currentUser != null){

        retVal = "success";

      }

    } on PlatformException catch(e){

      retVal = e.message!;

    } catch(e){

      print(e);

    }

    return retVal;

  }

**Модулі:**

* UserModel.

Код:

class BookModel {

  String? id; // ідентифікаційний номер книги

  String? name; // назва книги

  String? author; // автор

  String? description; // опис

  String? bookID; // номер книги

  String? bookImage; // зображення книги

  double? rating; // рейтинг книги

  bool? isFavorite; // Параметр для визначення чи є книга в списку "Улюблені"

  List<BookChapter>? chapterList;

  BookModel({

    this.id,

    this.name,

    this.author,

    this.description,

    this.bookID,

    this.bookImage,

    this.rating,

    this.chapterList,

    this.isFavorite = false, // Початкове значення isFavorite - false

    //this.dateCompleted,

  });

  BookModel.fromDocumentSnapshot({required DocumentSnapshot<Map<String, dynamic>> doc}) {

    id = doc.id;

    Map<String, dynamic>? data = doc.data();

    if (data != null) {

      name = data["name"];

      author = data["author"];

      description = data["description"];

      bookID = data["bookID"];

      bookImage = data["bookImage"];

      rating = data["rating"];

      isFavorite = data["isFavorite"];

      // Отримання списку глав із документа

      if (data["chapterList"] != null) {

        List<dynamic> chapterDataList = data["chapterList"];

        chapterList = chapterDataList.map((chapterData) => BookChapter.fromDocumentSnapshot(doc: chapterData)).toList();

      }

    }

  }

Опис: Модуль використовується для збереження інформації про книги та отримання такої інформації з бази даних.

* BookModel.

Код:

class UserModel {

  String? uid; // індіфікаційний номер користувача

  String? email; // елекктрона пошта

  String? userName; // ім'я або назва

  String? userID; // номер користувача

  List<dynamic>? favoriteBooks; // список улюблених книг

  Timestamp? accountCreated; // час створення/реєстрації

  UserModel({

    this.uid,

    this.email,

    this.userName,

    this.userID,

    this.favoriteBooks,

    this.accountCreated,

  });

  UserModel.fromDocumentSnapshot({required DocumentSnapshot<Map<String, dynamic>> doc}) {

    uid = doc.id;

    email = doc.get('email');

    userName = doc.get('userName');

    userID = doc.get('userID');

    favoriteBooks = doc.get("favoriteBooks");

    accountCreated = doc.get('accountCreated');

  }

Опис: Модуль використовується для збереження інформації про нинішнього користувача та відображення цієї інформації на вікні або екрані застосунку.

**Сервіси:**

* Database: використовується для збереження даних та визначення стану користувача в застосунку.

Код:

class Database {

  final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

  // Функція для генерації рандомного ID

  String generateRandomID() {

    // Генеруємо рандомний ID, наприклад, складаючи 6 випадкових цифр

    Random random = Random();

    String id = '';

    for (int i = 0; i < 6; i++) {

      id += random.nextInt(10).toString();

    }

     return id;

  }

  // Функція для додавання даних користувача до Firestore

  Future<String> createUser(UserModel user) async {

    String retVal = "error";

    bool isUniqueID = false;

    // Повторюємо генерацію рандомного ID і перевірку його унікальності, доки не буде знайдено унікальний ID

    while (!isUniqueID) {

      // Генеруємо новий рандомний ID

      String randomID = generateRandomID();

      // Перевіряємо, чи є ID унікальним

      isUniqueID = await isIDUnique(randomID);

      // Якщо ID унікальний, додаємо дані до Firestore та завершуємо цикл

      if (isUniqueID) {

        try {

        await FirebaseFirestore.instance.collection("users").doc(user.uid).set({

          'userName': user.userName,

          'email': user.email,

          'favoriteBooks': user.favoriteBooks,

          'userID': randomID, // Додаємо рандомний унікальний ID

          'accountCreated': Timestamp.now(),

          });

          retVal = "success";

        } catch (e) {

          print("Помилка при додаванні даних до Firestore: $e");

        }

      }

    }

    return retVal;

  }

  Future<UserModel> getUserInfo(String uid) async {

    UserModel retVal = UserModel();

    try{

      DocumentSnapshot \_docSnapshot = await \_firestore.collection("users").doc(uid).get();

      retVal.uid = uid;

      retVal.userName = \_docSnapshot.get("userName");

      retVal.email = \_docSnapshot.get("email");

      retVal.favoriteBooks = \_docSnapshot.get("favoriteBooks");

      retVal.userID = \_docSnapshot.get("userID");

      retVal.accountCreated = \_docSnapshot.get("accountCreated");

    }catch(e){

      print(e);

    }

    return retVal;

  }

Опис: Використовується для створення користувача в базі даних та його редагування з часом. Зберігає головні критерії користувачів, а саме їх назву, номер, час реєстрації, електронну пошту, тощо.

Також, використовуються інші методи, які містяться в інших вікнах і є для них унікальними по типу додавання і видалення книги з улюблених, налаштування самого статусу книги, тощо.

**Додатково:** *UML документації неможливо помітити в документації, тому вона мітиться в тій же папці, що і сама документація. В документації вказані головні елементи застосунку, які визначають її роботу та функціонування.*