ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Факултет ,,ФКСТ“

КУРСОВ ПРОЕКТ ПО ПРОГРАМИРАНЕ ЗА МОБИЛНИ УСТРОЙСТВА

(Тема: Електронен тефтер за рецепти - SWEET TREAT*)*

Изготвил:

Габриела Айгърова

46 група

Фак.номер: 121220046

Подпис:

гр.София

2023 г.

Съдържание

Увод

В днешно време употребата на мобилни устройства стана неотменима част от ежедневието на хора във всякаква възраст. Вече съществуват мобилни приложения, които улесняват живота на хората във всяко едно отношение.

Според проучване приложенията за готвене се използват най-много от потребители между 18 и 34 години. Около 45% от потребители на мобилни приложения за готвене са между 18 и 34 години, докато само около 16% са над 55 години. Ниският процент на възрастните хора, използващи такъв тип приложения, се дължи на факта, че някои приложения могат да са прекалено сложни или да имат твърде много функции, което води до затруднения при употребата им от по-възрастните потребители.

Приложението, което съм разработила за този курсов проект, цели да модернизира и дигитализира тефтера (бележника) с рецепти, който е характерен за почти всяко домакинство. То е разработено с възможно най-опростен дизайн и функционалност с цел употребата му от по-възрастната аудитория. Това не възпрепятства по никакъв начин то да бъде използвано и от хора на всякаква възраст имащи страст към кулинарията.

Анализ на съществуващи разработки

- **Paprika Recipe Manager** – Това приложение позволява на потребителите да запазват рецепти от уеб страници или ръчно. То организира рецептите в категории и ви дава възможност да ги търсите по различни критерии. Предимствата му са, че запазва рецепти от различни източници, включително уеб страници, и да ги организирате в категории. Също така има функционалност за търсене на рецепти по различни критерии, като например по съставки, категории или ключови думи. Недостатъците на това приложение са, че интерфейсът може да бъде труден за някои потребители, особено за начинаещи. Някои потребители съобщават за проблеми със синхронизацията между устройствата или загуба на данни.

- **Tasty** – Това приложение предлага рецепти с видео и кратки инструкции за приготвянето им. Предимствата на приложението са, че много от рецептите имат видео ръководства, които потребителя може да гледа, докато готви. Рецептите са подробни, като включват списък със съставките, точни мерки, време на готвене и брой порции. Към недостатъците спада факта, че приложението съдържа много реклами, които могат бъдат неприятни за някои потребители. Друг минус на Tasty - не всички рецепти са лесни за готвене, тъй като някои от тях могат да изискват специални умения или техники. Също така някои от рецептите може да съдържат необичайни или трудно достъпни съставки.

- **CookBook** - приложение за съхраняване на рецепти за готвене. Към плюсовете на това приложение спада голямата библиотека с рецепти, с която потребителя разползага, включително рецепти от различни кухни и категории, което го прави подходящо за всички вкусове и предпочитания. CookBook предлага лесен начин за споделяне на рецепти с приятели и семейство чрез електронна поща, съобщения и социални медии. Въпреки че има голяма библиотека с рецепти, не всички от тях са висококачествени. Някои рецепти могат да бъдат непълни или да съдържат грешки. Интерфейсът на приложението може да бъде малко затруднителен за навигация в началото, особено за потребители, които не са свикнали с подобни приложения.

Проектиране

За реализация на проекта са използвани:

Android Studio, защото чрез него може да се тества кода в реално време на различни видове устройства с Android – таблети, телефони и т.н. Студиото дава възможност да се симулират различни хардуерни функции като GPS, сензори за движение и много други.

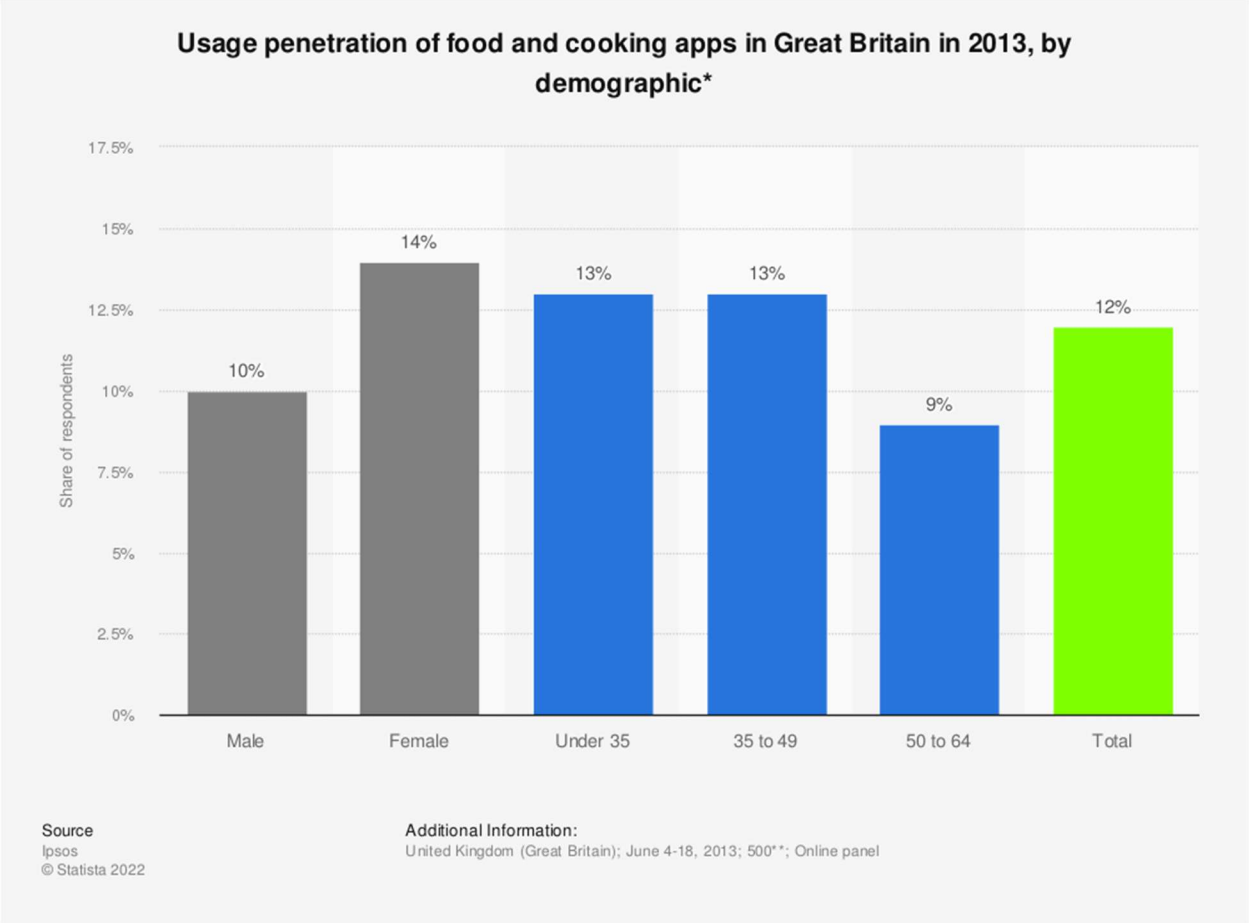
1. Имплементация на подходящи функционалности и екрани
    - „Welcome“ Екран
    - Екран „Вход“
    - Екран „Регистрация“
    - Валидации на данни
    - Екран „Моя профил“
    - Екран „Добави рецепта“
    - Екран „Моите рецепти“ (при налични такива за съответния потребител)
    - Екран, който уведомява потребителя, че няма добавени рецепти

2. Добавяне, извличане на данни във Firebase
    - Firebase Authentication
    - Firebase Database
    - Firebase Firestore

3. Потребителски интерфейс
    - Лого на приложението
    - Съвместими цветове и стил на текста
    - Опростен интерфейс

Кой ще използва продукта?

Основната група потребители, към които е насочена реализацията на приложението, са хората над 55 години, но разбира се е подходящо за хора на всякаква възраст, почитатели на готварството. Целта е да се създаде възможно най-опростено приложение за съхранение на рецепти, което да е удобно за употреба и от начинаещите в използването на мобилни устройства и приложения. *Фигура 1* изобразява изследване сред хора, които използват готварски приложения, проведено във Великобритания през 2013г. Можем да направим извод, че групата над 50 годишна възраст използват най-малко такъв тип приложения. Една от предполагаемите причини за този факт е сложността на съществуващите приложения. Затова приложението трябва да се изготви по начин, по който да може да се използва както млади хора така и най-вече от по – възрастното население.



Какви входни данни са нужни?

Единственото нужно на потребителите за достъп до функционалностите на приложението, е да имат имейл и парола. Проектът предоставя възможност за използване на Firebase DB, където да се записва информацията за всеки потребител и за лесното управление на различните акаунти.

Какво представлява Firebase и защо е избран за реализацията на проекта?

Firebase реализира Backend-as-a-Service решение както за мобилни, така и за уеб-базирани приложения, което включва услуги за изграждане, тестване и управление на приложения. BaaS позволява да се премахне необходимостта от управление на бекенд бази данни и получаване на съответен хардуер. Вместо това могат да се включат в приложение чрез специални API за всяка отделна услуга. В случая с Firebase има 7 от тях, които покриват целия спектър от back-end технологии за дадено приложение.

Причината, поради която, са използвани услугите и технологиите на Firebase е свързана главно с лесния достъп и управление на данни. Както Realtime DB, така и Firestore могат да актуализират бързо данните ни и да поддържат синхронизацията с множество други бази данни. Офлайн достъпът позволява приложението да бъде използваемо по всяко време.

Как добавяме Firebase към Android studio?

За да използваме Firebase в приложението си за Android, трябва да го регистрираме във Firebase акаунта ни чрез въвеждане на пълното име на пакета, в който се съдържа.

Задължително е да включим конфигурационния Firebase файл с име google-services.json към директорията на модула на ниво приложение. Конфигурационният файл на Firebase съдържа уникални, но несекретни идентификатори за текущия проект. Той асоциира приложението с конкретен проект на Firebase и неговите ресурси (бази данни, места за съхранение и т.н.). Конфигурацията включва „Firebase опции“, които представляват параметри, изисквани от услугите на Firebase и Google за комуникация с API сървъра на Firebase и за свързване на клиентски данни с Firebase проекта, към който принадлежи самото приложение.

За да активираме продуктите на Firebase в приложението си, добавяме google-services към Gradle файловете в Android Studio. В Gradle файл на ниво проект (build.gradle) дефинираме правилата, за да включим Google Services Gradle.

Автентификация

За успешен достъп до приложението, първо получаваме идентификационни данни за удостоверяване от потребителя. Тези идентификационни данни могат да бъдат имейл адресът и парола, след което се предават на Firebase Authentication SDK. После backend услугите проверяват данните и връщат отговор на клиента.

Първият път, когато потребител се регистрира, данните за потребителския профил във Firebase Auth се попълват с помощта на наличната информация. Потребителят се е регистрира с имейл адрес и парола, но се попълва само свойството на основния имейл адрес, но не и паролата. Firebase Auth не позволява видимост на паролите.

Токен за удостоверяване

Създава се от Firebase, когато потребител влезе в приложението. Тези токени са подписани JWT, които сигурно идентифицират клиента в проект на Firebase. Те съдържат основна информация за профила и низа за идентификация на потребителя, който е уникален за проекта Firebase.

Текущ потребител

Когато клиент се регистрира или влезе, той става текущият потребител на екземпляра за удостоверяване. Екземплярът запазва състоянието на потребителя, така че опресняването или рестартирането на приложението да не загуби информацията за потребителя. Когато той излезе, екземплярът на Auth спира да поддържа препратка към потребителския обект и вече не запазва състоянието му; няма текущ потребител. Въпреки това, потребителският екземпляр продължава да бъде напълно функционален.

Firebase Firestore

Firebase Firestore е облачно хранилище на NoSQL база данни в реално време. Предназначена е за корпоративна употреба, което включва мащабируемост, сложни модели на данни и разширени опции за заявки. Firebase конзолата може да се използва за преглед на данни в двете бази данни.

Модел, по който са създадени данните:

Следвайки модела на данни NoSQL на Cloud Firestore, ще се съхраняват данни в документи, които съдържат полета, съпоставящи се със стойности. Тези документи ще се съхраняват в колекции, които са самите контейнери за документите, които ще се ползват, за организиране на данните и за създаване на заявки. Документите поддържат много различни типове данни, от прости низове и числа до сложни, вложени обекти. Има и възможността за създаване на подколекции в документи и за изграждане на йерархични структури, които се мащабират с нарастването на базата данни.

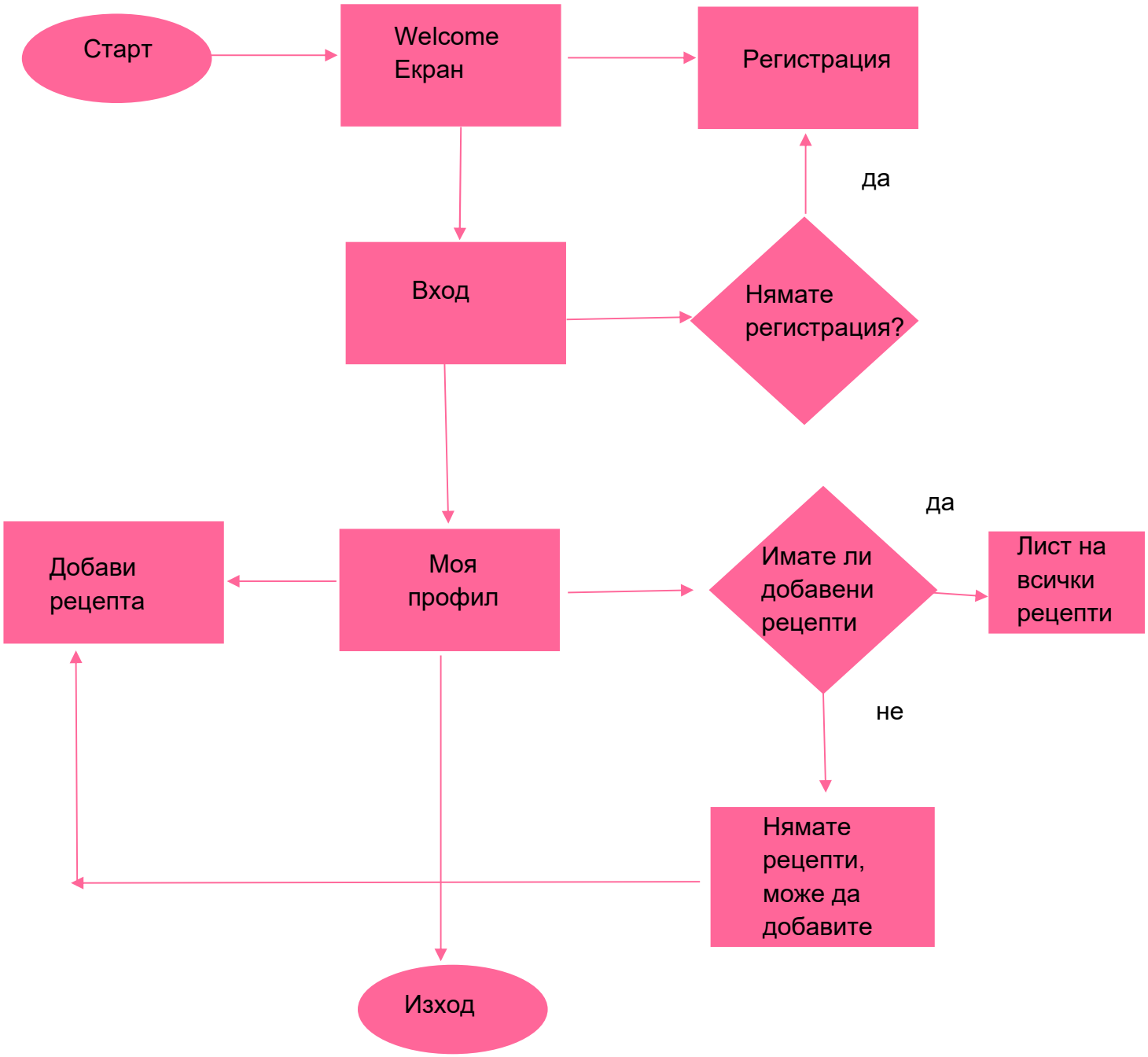
Как ще бъдат достъпвани функционалностите от потребителя?

Едно от най – важните условия за отлична работа на приложението е то да бъде лесно използваемо. Потребителите не трябва да изпитват затруднения с употребата му. При стартиране на проекта, се появява „Welcome“ екран, който съдържа два бутона – един за вход и друг за регистрация. Ако потребителя няма регистрация, той може да направи такава дори, ако вече е в екрана за „Вход“ чрез препратката към екрана за регистрация.

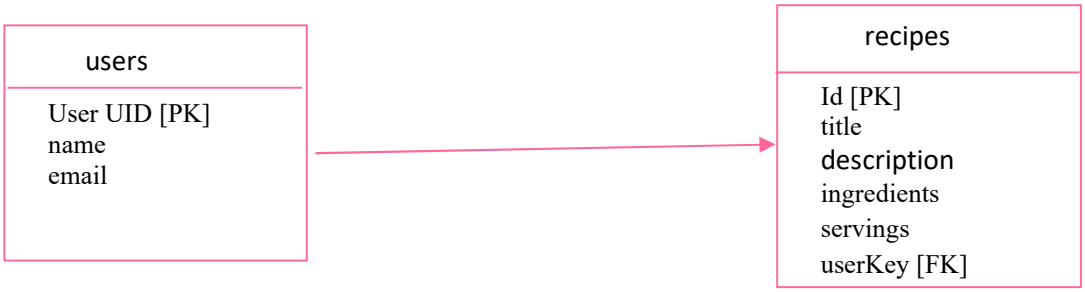
След като бъде направена регистрацията, потребителя бива препращан към екрана за вход. Това се прави за да се гарантира, че входът в системата е винаги свързан с регистрацията на потребителя, а не е възможно някой да се върне към системата след известно време и да влезе без да се регистрира.

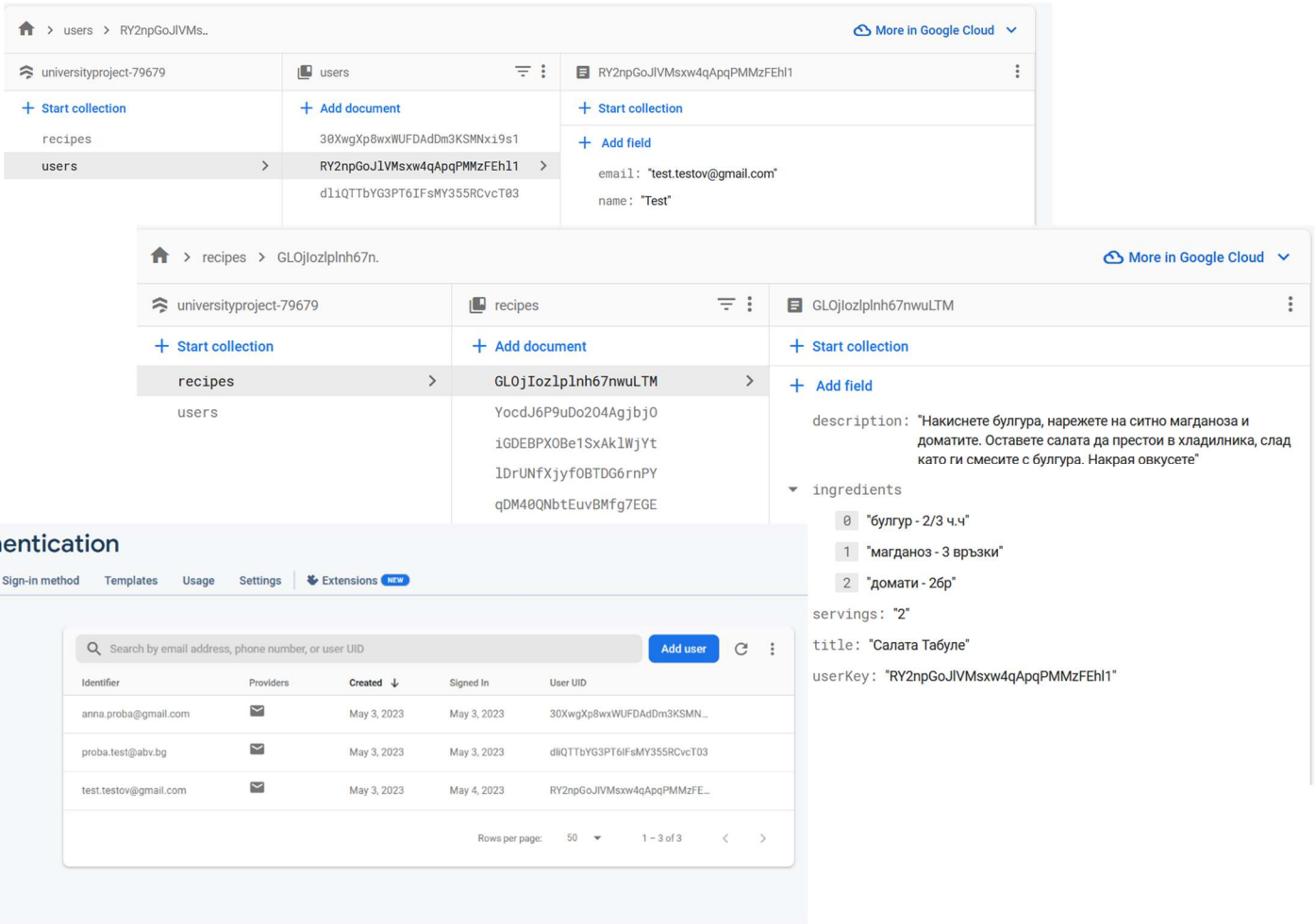
След успешно влизане - приложението отвежда потребителя към екрана „Моя профил“. В него има карта с текущото местоположение на потребителя, както и магазини, от които могат да се закупят необходимите продукти (те са валидни за пет града в България – София, Пловдив, Варна, Бургас и Русе). Също в този екран има три бутона – един за добавяне на рецепта, друг за показване на наличните рецепти на текущия потребител и третия е за изход от приложението. Ако потребителя няма рецепти, той бива препратен към екран с бутон към екрана за добавяне на рецепта.

За най – добрата репрезентация на работата на приложението е представено с помощта на диаграма:



ER Диаграма:

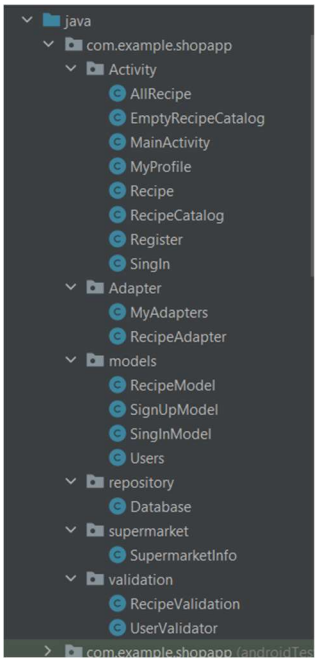




Вход и регистрация

След като свързахме базата към Android Studio, следва да създадем няколко профила на потребители и да тестваме работата на Firebase. Както може да се види, всеки потребител има свой уникален идентификационен номер – UID

Реализация



Логиката на приложението е разделена основно в следните package-и: activity(там се намират всички класове, които служат за управление на жизнения цикъл на екраните – създадени чрез xml-и), adapter (връзките между данните и потребителския интерфейс.), models (моделите, които се използват в приложението), repository (за методите свързани с базата данни), supermarket (за информацията свързана с координатите на супермаркетите), validation (валидациите в приложението).

В MainActivity класа (началният екран, който се показва на потребителя при стартиране на приложението – „Welcome“ екрана) има два бутона – за Вход и Регистация

```
public class MainActivity extends AppCompatActivity {

    DatabaseReference databaseReference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        databaseReference = FirebaseDatabase.getInstance().getReference();
    }

    public void onRegisterButtonClick(View view) {
        Intent intent = new Intent(MainActivity.this, Register.class);
        startActivity(intent);
    }

    public void onSingInButtonClick(View view) {
        Intent intent = new Intent(MainActivity.this, SingIn.class);
        startActivity(intent);
    }
```

xml activity-то:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="@drawable/main"
    tools:context=".Activity.MainActivity">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="left"
        android:padding="0dp"
        android:paddingTop="0dp">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="181dp"
            android:layout_height="200dp"
            android:layout_marginStart="9dp"
            android:layout_marginTop="-9dp"
            android:paddingLeft="-70dp"
            app:srcCompat="@drawable/logo" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:padding="10dp"
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="-29dp">

        <TextView
            android:id="@+id/register_title"
            android:layout_width="290dp"
            android:layout_height="150dp"
            android:layout_marginStart="30dp"
            android:layout_marginTop="270dp"
            android:text=" Добре дошли\n             в\n    Sweet Treat"
            android:textSize="35dp"
            android:textStyle="bold"
            android:textColor="@android:color/holo_purple"
            tools:ignore="MissingConstraints"
            tools:layout_editor_absoluteX="100dp"
            tools:layout_editor_absoluteY="75dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:paddingTop="430dp"
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="-29dp">

        <Button
            android:id="@+id/buttonSingIn"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onSingInButtonClick"
            android:text="Вход"
            tools:ignore="MissingConstraints"
            tools:layout_editor_absoluteX="158dp"
            tools:layout_editor_absoluteY="600dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingTop="500dp"
        android:paddingLeft="130dp"
        tools:layout_editor_absoluteX="-107dp"
        tools:layout_editor_absoluteY="0dp">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRegisterButtonClick"
        android:text="Регистрация"
        tools:layout_editor_absoluteX="158dp"
        tools:layout_editor_absoluteY="500dp"
        tools:ignore="MissingConstraints" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

По-голямата част от останалите xml activity-та са реализирани по този начин.

В SingIn класа (който е activity към activity_sing_in2.xml) са представени методите за вход (при натискане на бутона) и метода за регистрация (при натискане на „Нямате акаунт? Регистрирайте се сега!“), който служи за препратка към Register (който е activity към activity_reg.xml).

```
Button button = findViewById(R.id.singIn);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String emailValue = email.getText().toString();
            String passwordValue = password.getText().toString();

            SingInModel singInModel = new SingInModel(emailValue,
passwordValue);
            if (userValidator.isUserLogInValid(singInModel, SingIn.this)) {
                Intent intent = new Intent(SingIn.this, MyProfile.class);
                startActivity(intent);
```

```
            }

        }
    });

    register.setOnClickListener(view -> startActivities(new Intent[]{new
Intent(SingIn.this, Register.class)}));
}
```

При Register клас наблюдаваме при натискане бутона отвеждане към layout-а на register-а, верификация на данните, при които, ако всичко е наред се запазват в базата. След успешно запазване на данните в базата, потребителя бива препратен към екрана за вход.

```
Button button = findViewById(R.id.singIn);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String nameValue = name.getText().toString();
            String emailValue = email.getText().toString();
            String passwordValue = password.getText().toString();
            String rePasswordValue = rePassword.getText().toString();

            SignUpModel signUpModel = new SignUpModel(nameValue, emailValue,
passwordValue, rePasswordValue);

            if (userValidator.isUserValid(signUpModel, Register.this)){
                Intent intent = new Intent(Register.this, SingIn.class);
                startActivity(intent);
                finish();
            }
        }
    });
}
```

В UserValidator класа се изпълнява проверка на данните, въведени от потребителя и не премина към следващата стъпка – запазване в базта, докато не бъдат въведени валидни данни. Потребителя вижда какво е объркал с помощта на Toast (най-често се използва за информиране на потребителя за важни събития или действия).

```
public boolean isUserValid(SignUpModel signUpModel, Register register) {
    boolean flag = true;
    String error = "";
    if (TextUtils.isEmpty(signUpModel.getName())) {
        error += "Моля, въведете име\n";
        flag = false;
    }

    if (TextUtils.isEmpty(signUpModel.getEmail())) {
```

```
        error += "Моля, въведете имейл\n";
        flag = false;
    } else if (!isValidEmail(signUpModel.getEmail())) {
        error += "Моля, въведете валиден имейл\n";
        flag = false;
    }

    if (TextUtils.isEmpty(signUpModel.getPassword())) {
        error += "Моля, въведете парола\n";
        flag = false;
    } else if (isPasswordStrong(signUpModel.getPassword())) {
        error += "(Паролата трябва да съдържа малки и големи букви, цифри и
поне един символ" + //
                "\nдължина между 6 и 20 сумвола)\n";
        flag = false;

    }

    if (TextUtils.isEmpty(signUpModel.getRePassword())) {
        error += "Моля, повторете паролата\n";
        flag = false;

    } else if (!isPasswordMatch(signUpModel.getPassword(),
signUpModel.getRePassword())) {
        error += "Паролата трябва да съвпада";
        flag = false;

    }

    if (flag) {
        Users user = new Users(signUpModel.getName(),signUpModel.getEmail(),
signUpModel.getPassword());
        registerUser(user, register);
        return signInSuccessful;
    } else {
        Toast toast = Toast.makeText(register.getBaseContext(), error,
Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }
    return false;
}
```

Другите методи за валидация, които има в UserValidator са за проверка на силна парола (isPasswordStrong) – съдържаща главни и малки букви, цифри и символи. Общият им брой трябва да бъде между 6 и 20 символа. Има метод за проверка на валиден имейл (isValidEmail), метод за проверка на съвпадаща парола (isPasswordMatch). Също така, освен показания по-горе метод за валидация на потребител при регистрация има сходен при вход.

В MyProfile (който е activity към activity_after_sign_in.xml) в onCreate метода се съдържа действията, случващи се при натискането на трите бутона.

При „Изход“ бутона – извършва се logout на потребителя, където чрез метода signOut() на FirebaseAuth се излиза от текущо активния акаунт в Firebase Authentication. Изтриват се запаметените данни за имейл и парола от SharedPreferences, като използва методите remove() и apply().

```
Button exitButton = findViewById(R.id.button);
exitButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        FirebaseAuth.getInstance().signOut();
        SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.remove("email");
        editor.remove("password");
        editor.apply();
        startActivity(new Intent(getApplicationContext(),
MainActivity.class));
        finish();
    }
});
```

Следващия бутон служи за пренасочване към Recipe класа (който е activity към activity_recipe.xml).

```
Button button = findViewById(R.id.buttonAddRecipe);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MyProfile.this, Recipe.class);
        startActivity(intent);
        finish();
    }
});
```

При бутона за извеждане на всички рецепти се наблюдава проверка в базата, като се вземе Uid-то на текущия потребител и чрез Query се проверява дали в колекцията "recipes" има рецепта с "userKey", отговарящ на Uid-то на текущия потребител. Ако не съществува, от това следва, че този потребител няма запазени рецепти и се пренасочва, към екран, в който това му се съобщава и има възможност да се върне обратно към екрана „Моя профил“ или да отиде към екран „Добави рецепта“, където да добави своята първа рецепта.

Ако обаче вече има запазени рецепти, той бива пренасочен към екрана „Моите рецепти“. (за който отговаря класа RecipeCatalog)

```
Button buttonAllRecipes = findViewById(R.id.buttonAllRecipes);
buttonAllRecipes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        List<RecipeModel> myObjectList = new ArrayList<>();
        Query query = firestore.collection("recipes").whereEqualTo("userKey",
firebaseAuth.getCurrentUser().getUid());
        query.addSnapshotListener(new EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot snapshot, @Nullable
FirebaseFirestoreException e) {

                if (e != null) {
                    Intent intent = new Intent(MyProfile.this,
EmptyRecipeCatalog.class);
                    startActivity(intent);
                    finish();
                }
                myObjectList.clear();

                if (snapshot.isEmpty()) {
                    Intent intent = new Intent(MyProfile.this,
EmptyRecipeCatalog.class);
                    startActivity(intent);
                    finish();
                } else {
                    Intent intent = new Intent(MyProfile.this,
RecipeCatalog.class);
                    startActivity(intent);
                    finish();
                }

            }
        });
    }
});
```

При класа RecipeCatalog се създава RecyclerView (позволяващ ефективното показване на списъци и големи колекции от данни в приложенията). Използван е и adapter -> adapter = new MyAdapters(myObjectList) – чрез, който се създава нов обект от класа MyAdapters, който се използва за преобразуване на обектите взети от базата под формата на List в редове на RecyclerView.

В класа MyProfile в метода onCreate след методите за бутоните има метод за настройване на картата setUpMap(). Той се използва за инициализиране на картата и добавяне на маркер, който показва текущото местоположение на потребителя. Методът onLocationChanged() се изпълнява когато местоположението на потребителя се промени, създавайки обект от тип Location, който съдържа информация за текущото местоположение на потребителя. За зумване на екрана на мобилното приложение се използва ScaleListener. Също има още два метода, които са onResume() и onPause() – те се отнася до работата с местоположението на устройството. Чрез onResume() се проверява дали има необходимите разрешения за достъп до местоположението, ако няма, то извежда съобщение за грешка.

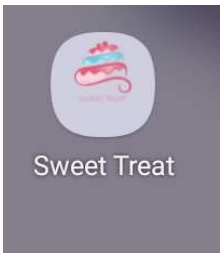
В метода onPause() приложението премахва заявките за актуализация на местоположението. Това е важно за оптимизиране на работата на приложението и спестяване на батерията на устройството.

Класа, в който са по-голямата част от методите за работа с базата е Database. Той съдържа методи за извеждане рецептите на потребителя (allRecipeOfUser), за настройване на адаптера за рецептите (setUpRecipeAdapter), за добавяне на рецепта в базата (addRecipeToDatabase), както и за регистрация и вход на потребител.

Освен валидация за потребителите има валидация и за рецептите – RecipeValidation. В този клас се проверяват дали полетата на рецептите са коректно въведени при създаване.

Друг клас, който се използва в приложението е SupermarketInfo, в който се задават координатите на някой предложения за магазини, от които потребителите могат да си набавят необходимите продукти (налични за: София, Пловдив, Бургас, Варна и Русе) и им поставя маркер, за да могат да се отбележат на картата. Маркерът, който се използва за магазините е логото на приложението (SWEET TREAT), а този за текущото местоположение – маркер за локация.

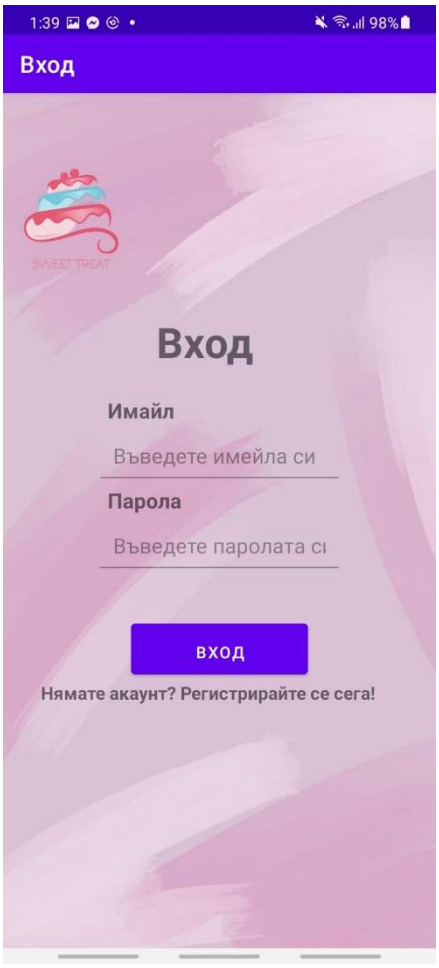
Потребителски ръководство



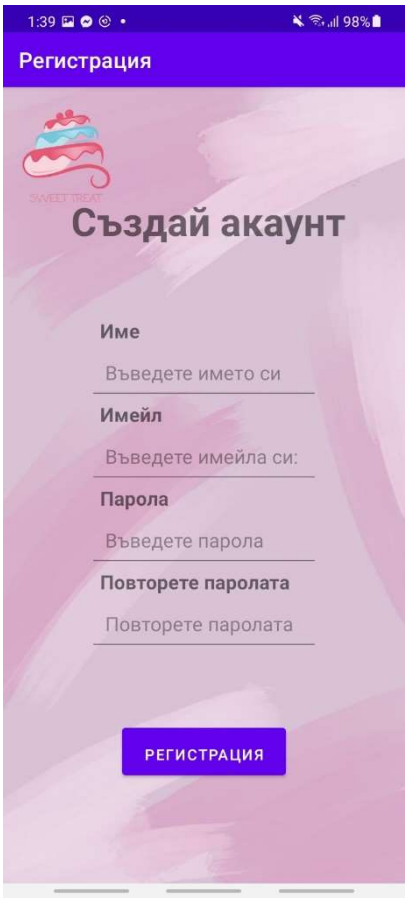
Иконата на приложението, чрез която да се намери по-лесно приложението SWEET TREAT

Начален екран – „Welcome“ екран

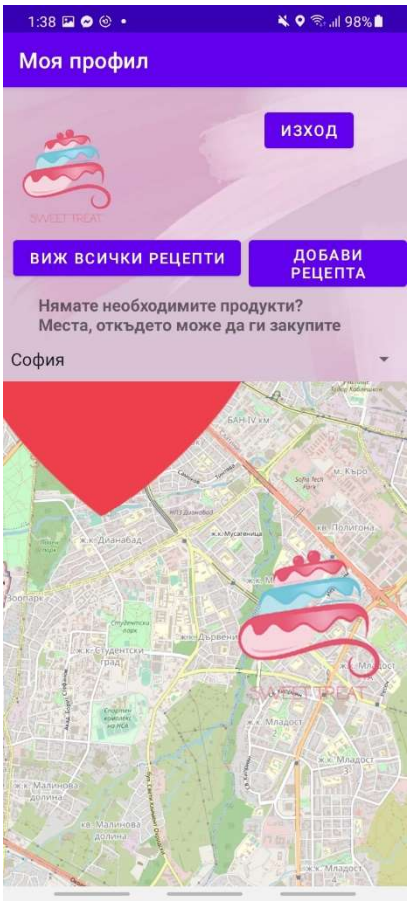
Екран „Вход“



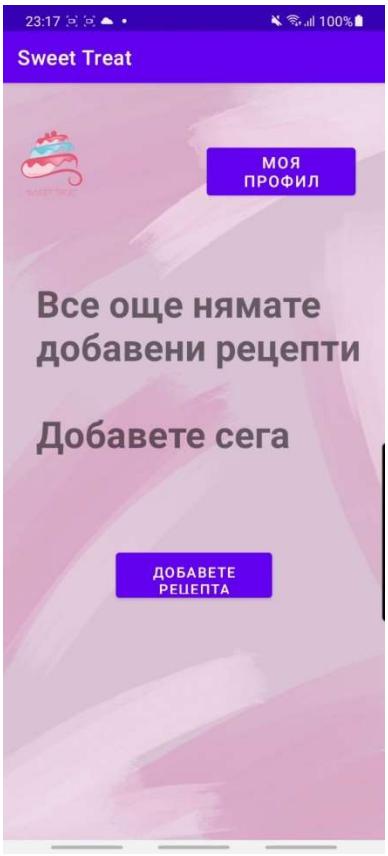
Екран „Регистрация“



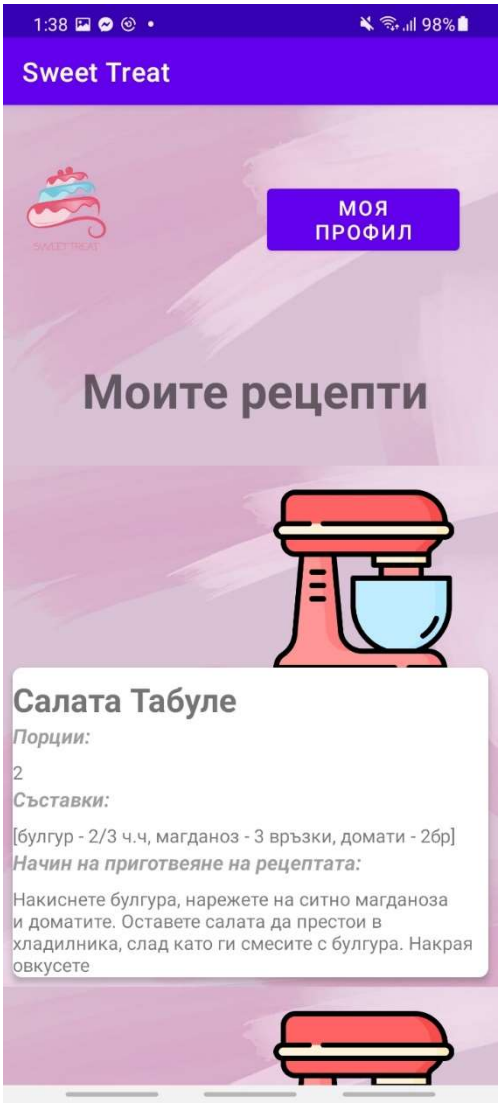
Екран „Моя профил“



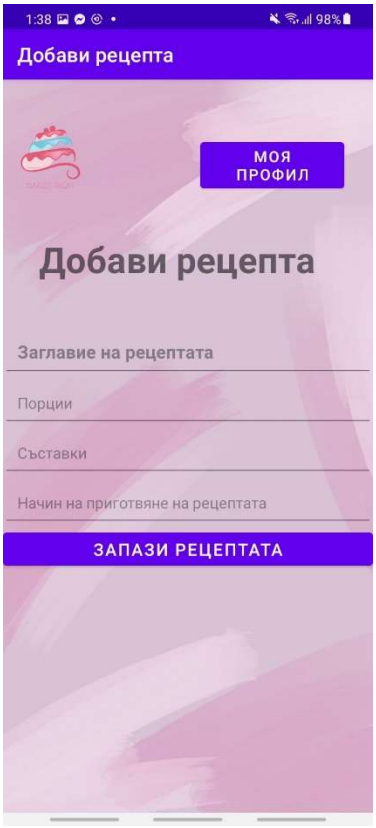
Екран при липса на рецепти



Екран „Моите рецепти“



Екран „Добави рецепта“



Заключение

SWEET TREAT може да се разпознае по неговото лого. Също така за приложението е характерно, че сравнително лесен за употреба и предразполага употребата му от всички възрастови групи, без значение дали са начинаещи в употребата на мобилни устройства или не. Създаването на приложението „SWEET TREAT“ протече в няколко етапа: проучване, планиране, проектиране, разработване в среда и тестване. Срещайки нуждите на таргет групата, смятам, че на този етап на разработка, продуктът е готов за употреба.

Литература

Firebase-> https://firebase.google.com/docs

Developer android-> https://developer.android.com/docs

Statista -> https://www.statista.com/statistics/286480/food-and-cooking-app-users-in-great-britain-by-deomographic/

Stackoverflow -> https://stackoverflow.com/

Google Images