

Outfitr App

Ανάπτυξη Android εφαρμογής στα πλαίσια του μαθήματος
“Ανάπτυξη Εφαρμογών Κινητών Συσκευών”



Περιεχόμενα

Σκοπός της εφαρμογής.....	2
Τυπικοί χρήστες.....	2
Λειτουργικότητα της εφαρμογής.....	2
Καινοτομία.....	6
Περιγραφή του κώδικα.....	7
Βάση Δεδομένων.....	7
Περιγραφή των 6 activities.....	9
1. Login Activity.....	9
2. Signup Activity.....	11
3. Main Activity.....	12
4. Upload Tops Activity.....	13
5. Upload Bottoms Activity.....	14
6. Upload Shoes Activity.....	15
Περιγραφή των 5 fragments.....	15
1. Tops Fragment.....	15
2. Bottoms Fragment.....	16
3. Shoes Fragment.....	16
4. Create Fragment.....	16
5. Account Fragment.....	16
Περιγραφή των επιπλέον κλάσεων.....	17
1. User.....	17
2. My Adapter.....	18
3. DataClass.....	18
Mockup Design.....	19

Test Account

email: account@gmail.com

password: new123

Εικόνες που μπορείτε να χρησιμοποιήσετε για testing:

- [Tops](#)
- [Bottoms](#)
- [Shoes](#)

Σκοπός της εφαρμογής

Σκοπός της εφαρμογής είναι η οργάνωση της ντουλάπας του χρήστη και ο συνδυασμός των ρούχων του για τη δημιουργία outfits.

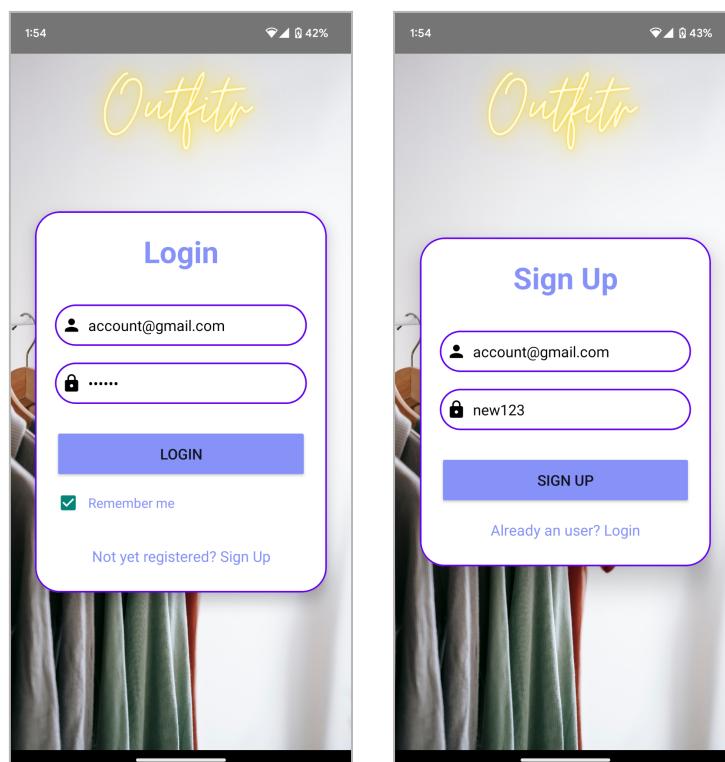
Τυπικοί χρήστες

Χρήστες της εφαρμογής είναι όσοι ενδιαφέρονται για το στυλ και τη μόδα.

Λειτουργικότητα της εφαρμογής

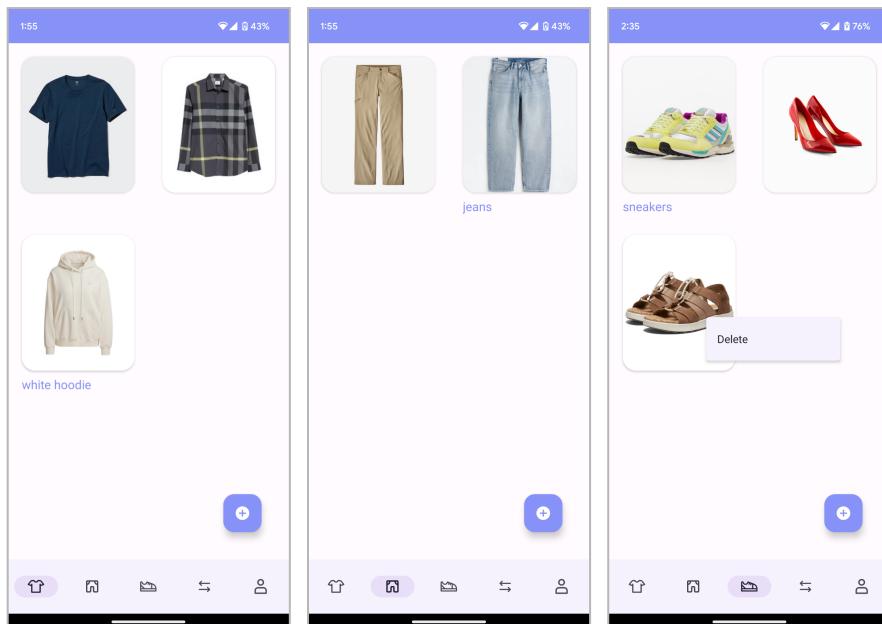
Προσοχή: Η εφαρμογή απαιτεί σύνδεση στο διαδίκτυο για να λειτουργήσει.

Η εφαρμογή υποστηρίζει **login** και **signup** και μπορεί να “θυμάται” τον χρήστη ώστε να μην χρειάζεται να κάνει login κάθε φορά που την ανοίγει. Ο χρήστης μπορεί να κάνει signup με το email του.

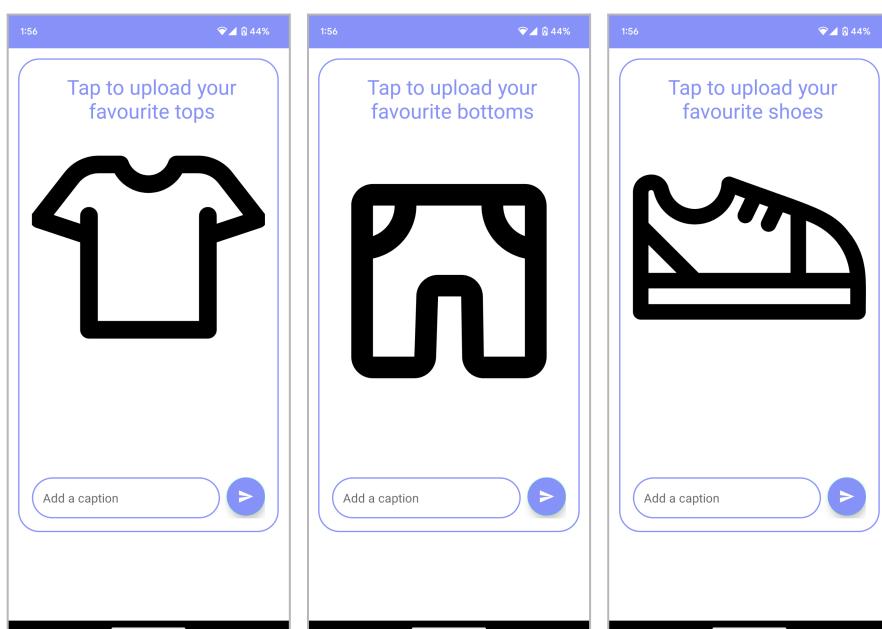


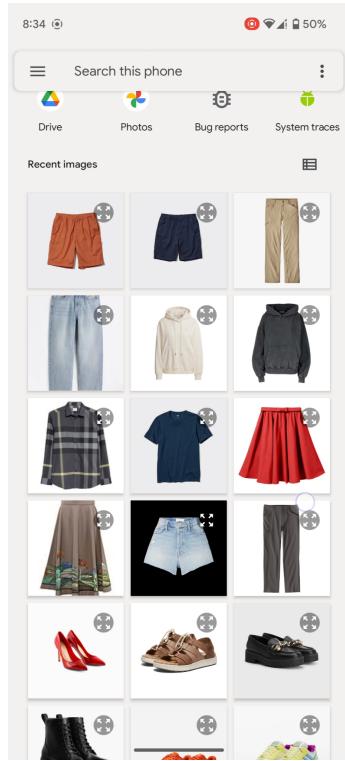
Αφότου ο χρήστης συνδεθεί, η εφαρμογή έχει ένα **bottom navigation bar** με πέντε καρτέλες (Tops, Bottoms, Shoes, Create και Account).

Στην καρτέλα **Tops** εμφανίζονται τα tops του χρήστη (δεν εμφανίζεται τίποτα αν ο χρήστης δεν έχει ανεβάσει εικόνες ή αν δεν έχει πρόσβαση στο διαδίκτυο) και υπάρχει κάτω δεξιά κουμπί “+” ώστε να προσθέσει ο χρήστης tops. Αν ο χρήστης θέλει να διαγράψει μια εικόνα, αρκεί να πατήσει παρατεταμένα την εικόνα και έπειτα το κουμπί “Delete” που θα εμφανιστεί. Αντίστοιχες είναι και οι καρτέλες **Bottoms** και **Shoes**.

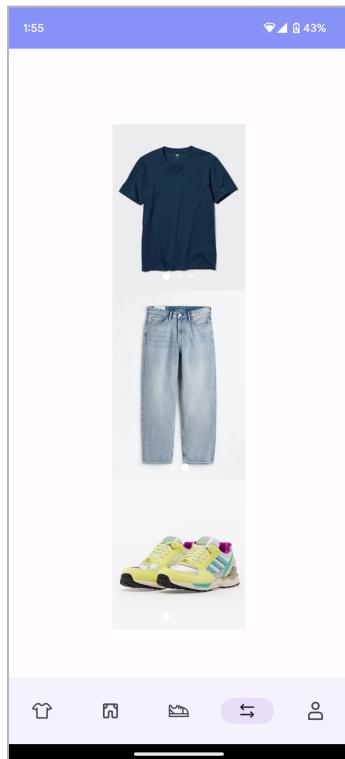


Πατώντας το κουμπί “+”, ο χρήστης μπορεί να ανεβάσει εικόνες από τη συλλογή του αντίστοιχα σε κάθε καρτέλα. Ο χρήστης μπορεί να ανεβάζει μία εικόνα κάθε φορά, δηλαδή δεν μπορεί να επιλέξει πολλές εικόνες από τη συλλογή του και να τις ανεβάσει ταυτόχρονα.

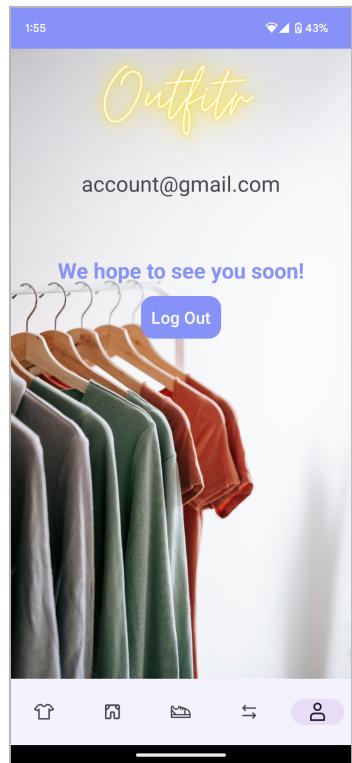




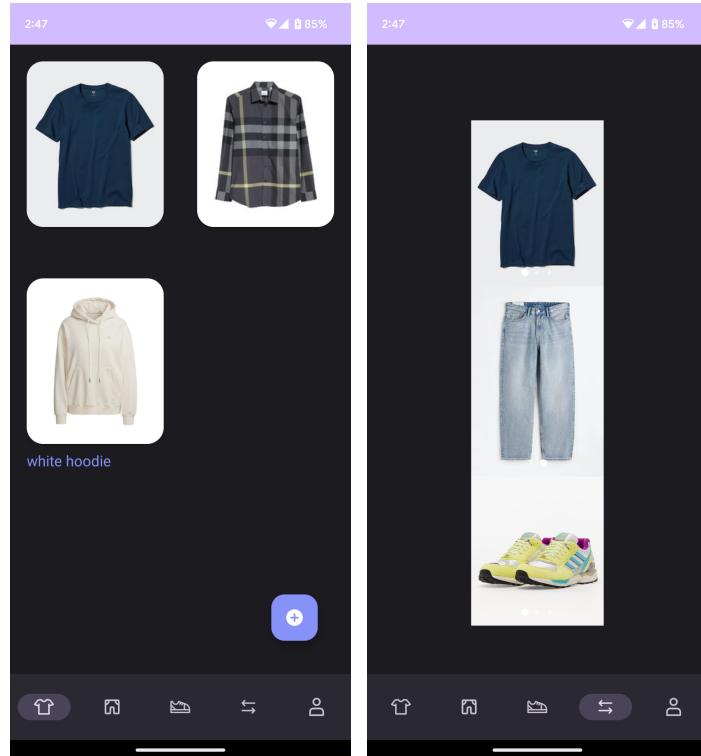
Στην καρτέλα **Create**, εμφανίζονται τα ρούχα του χρήστη σε sliders ώστε να μπορεί να τα συνδυάσει και να δημιουργήσει outfits (δεν εμφανίζεται τίποτα αν ο χρήστης δεν έχει ανεβάσει τίποτα ή αν δεν έχει σύνδεση στο διαδίκτυο).



Στην καρτέλα **Account**, εμφανίζεται το email του χρήστη και το κουμπί “Log Out” σε περίπτωση που ο χρήστης θέλει να αποσυνδεθεί από την εφαρμογή.

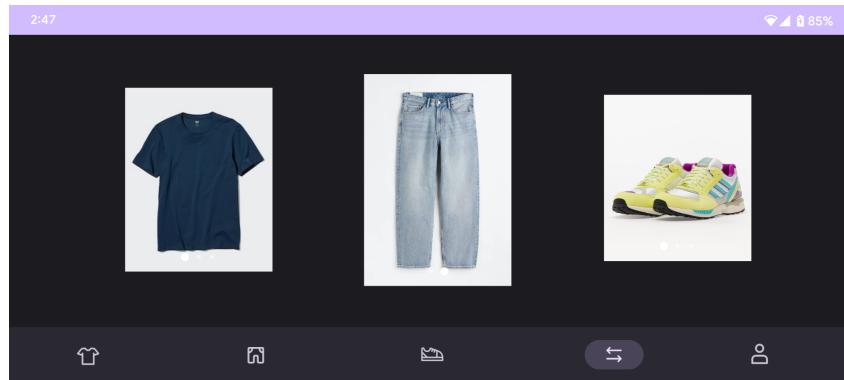


Αν το κινητό είναι ρυθμισμένο σε dark mode, τότε και η εφαρμογή είναι σε dark mode.



Η εφαρμογή υποστηρίζει μόνο **portrait mode**, καθώς δεν θα έβγαζε νόημα ο χρήστης να την χρησιμοποιήσει σε **landscape mode**. Ιδιαίτερα στην καρτέλα **Create**, δεν θα μπορούσαν τα ρούχα να είναι το ένα κάτω από το άλλο για τη δημιουργία outfits. Όπως φαίνεται στην

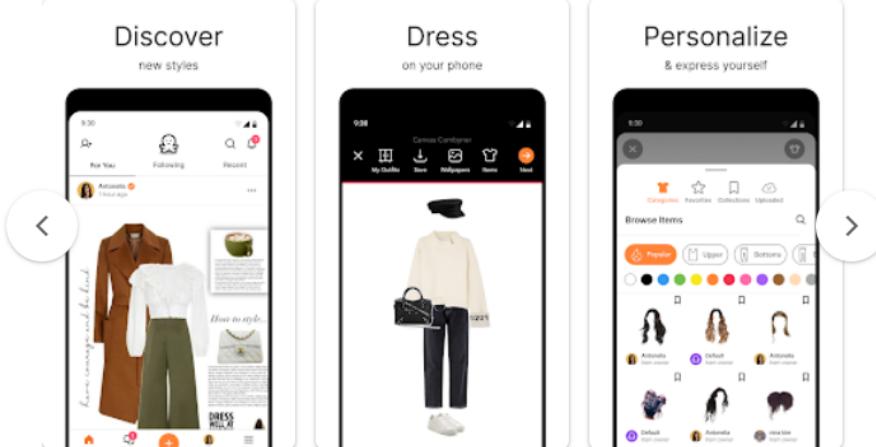
παρακάτω εικόνα θα χαλούσε όλο το concept της εφαρμογής. Για τη διατήρηση, λοιπόν, της συνέπειας, όλη η εφαρμογή είναι σε portrait mode.



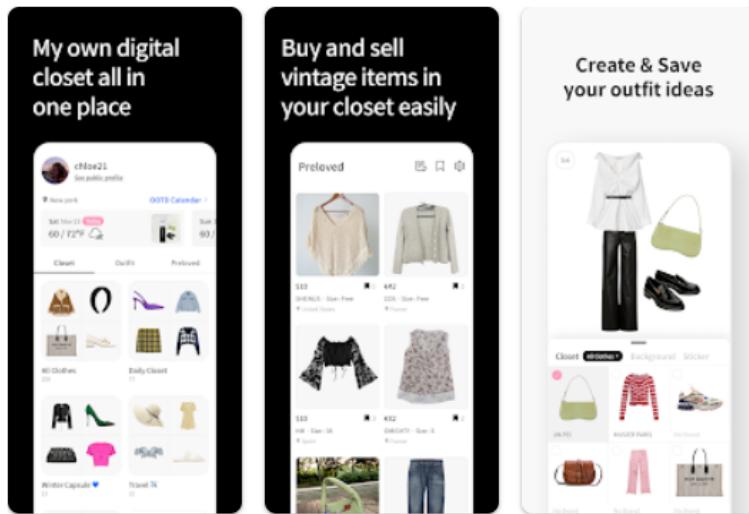
Καινοτομία

Όσον αφορά την **καινοτομία** της εφαρμογής, υπάρχουν αντίστοιχες εφαρμογές στο Play Store. Όμως καμία δεν στοχεύει στην απλότητα την οποία εμείς επιθυμούμε. Οι αντίστοιχες εφαρμογές που υπάρχουν στο Play Store έχουν υπερβολικά πολλές λειτουργίες και το User Interface τους είναι πολύ περίπλοκο και “μπουκωμένο”.

Επίσης, εμείς χρησιμοποιούμε ένα διαφορετικό τρόπο για το συνδυασμό των ρούχων. Συγκεκριμένα, χρησιμοποιούμε sliders κι έτσι ο χρήστης μπορεί να δημιουργήσει outfits εύκολα και γρήγορα μέσω swipping. Από την άλλη μεριά, οι άλλες εφαρμογές χρησιμοποιούν “drag and drop” τεχνολογία, αντίστοιχα με ένα πρόγραμμα δημιουργίας collage.



Combyne - Outfit Creation



Acloset - AI Fashion Assistant

Περιγραφή του κώδικα

Η εφαρμογή έχει 6 **activities**:

1. Login activity
2. Signup activity
3. Main activity
4. Upload tops activity
5. Upload bottoms activity
6. Upload shoes activity

To main activity αποτελείται από 5 **fragments**:

1. Tops fragment
2. Bottoms fragment
3. Shoes fragment
4. Create fragment
5. Account fragment

Στον κώδικα υπάρχουν επίσης και οι κλάσεις:

1. User
2. MyAdapter
3. DataClass

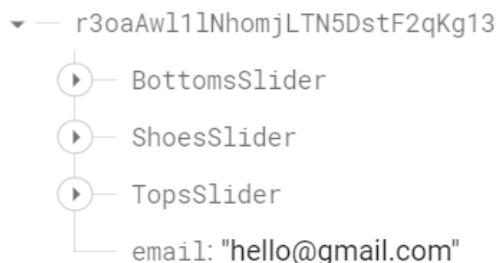
Βάση Δεδομένων

Η εφαρμογή διαθέτει μηχανισμό μόνιμης αποθήκευσης δεδομένων σε **Firebase**. Χρησιμοποιείται το **firebase authentication** για το sign up και το login το χρηστών και **realtime database** για την αποθήκευση των ρούχων του κάθε χρήστη.

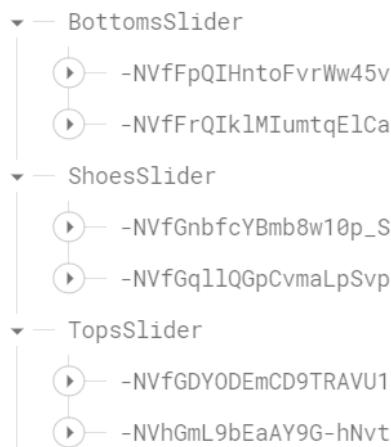
Συγκεκριμένα, στο realtime database υπάρχει ένα node “**Users**” με τα ID όλων των χρηστών.



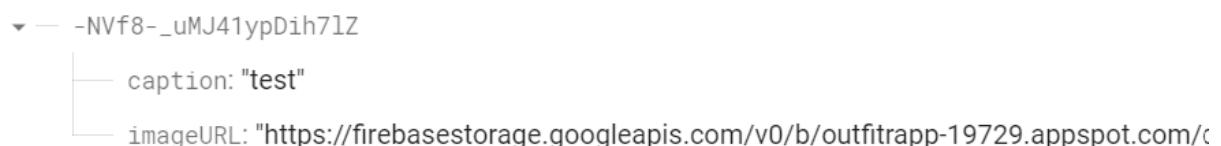
Το ID λαμβάνεται από το firebase authentication. Κάτω από το ID του κάθε χρήστη υπάρχει το email του καθώς και τρία ακόμη nodes (TopsSlider, BottomsSlider και ShoesSlider), όπως φαίνεται στην παρακάτω εικόνα.



Στο TopsSlider αποθηκεύονται όλες οι εικόνες που είναι τα tops του χρήστη. Αντίστοιχα για το BottomsSlider και το ShoesSlider.



Κάθε εικόνα έχει ένα id και κάθε id είναι ένας κόμβος με στοιχεία το caption της εικόνας και το url της εικόνας.



Περιγραφή των 6 activities

1. Login Activity

Το **Login Activity** είναι ο main launcher της εφαρμογής που διαχειρίζεται τις εισόδους χρηστών στην εφαρμογή.

Πρώτα ελέγχεται αν το checkbox Remember me είναι checked στην `onClick` του signup button από την προηγούμενη σύνδεση χρήστη και αν είναι αληθείς τότε πηγαίνει στο Main Activity με τον χρήστη να “βλέπει” την σελίδα των tops του.

```
// Checks if the user wanted to be Logged in
SharedPreferences preferences=getSharedPreferences( name: "checkbox",MODE_PRIVATE);
String checkbox = preferences.getString( s: "remember", s1: "" );
if(checkbox.equals("true")){
    Intent intent = new Intent( packageContext: LoginActivity.this,MainActivity.class );
    startActivity(intent);
}
```

Ωστόσο αν δεν είναι αληθής η προηγούμενη συνθήκη, ανάλογα με την συμπεριφορά του χρήστη γίνονται αντίστοιχες ενέργειες. (Προσπάθεια εισόδου με κωδικούς ή κλείσιμο εφαρμογής).

Το authentication και άρα η σύνδεση του γίνεται μόνο όταν τα στοιχεία που πληκτρολογεί ισοδυναμούν με κάποια από τα στοιχεία που υπάρχουν ήδη στην βάση όπου και εμφανίζεται μήνυμα *“Login Successful”* ειδάλλως εμφανίζεται *“Login Failed”* αν ο κωδικός είναι λανθασμένος.

Σε περίπτωση που ο χρήστης αφήσει κενά τα πεδία email ή password παίρνει αντίστοιχο μήνυμα και αν το email δεν είναι σωστό τότε προκύπτει μήνυμα λάθους *“Please enter correct email”*.

```

@Override
public void onClick(View v) {
    String email = loginEmail.getText().toString();
    String pass = loginPassword.getText().toString();
    if (!email.isEmpty() & Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        if (!pass.isEmpty() ) {
            auth.signInWithEmailAndPassword(email, pass)
                .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                    @Override
                    public void onSuccess(AuthResult authResult) {
                        Toast.makeText( context: LoginActivity.this, text: "Login Successful", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));
                        finish();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText( context: LoginActivity.this, text: "Login Failed", Toast.LENGTH_SHORT).show();
                    }
                });
        } else {
            loginPassword.setError("Empty fields are not allowed");
        }
    } else if (email.isEmpty()) {
        loginEmail.setError("Empty fields are not allowed");
    } else {
        loginEmail.setError("Please enter correct email");
    }
}

```

Ο χρήστης έχει την δυνατότητα να παραμένει Logged in στην εφαρμογή αν το επιλέξει. Αν πατήσει το checkbox και έπειτα το button τότε θα παραμένει συνδεδεμένος στην εφαρμογή χωρίς να κάνει συνέχεια σύνδεση, εώς ότου κάνει logout, διαφορετικά αν δεν το πατήσει με το που ξανά μπει στην εφαρμογή θα ξανά βρεθεί στο **Login Activity**. Η υλοποίηση αυτού γίνεται στην `onCheckedChanged` συναρτηση.

```

// Checks if checkboxes are clicked or not
@user1
remember.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    2 usages @user1
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {

        if (compoundButton.isChecked()){
            SharedPreferences preferences = getSharedPreferences( name: "checkbox", MODE_PRIVATE);
            SharedPreferences.Editor editor = preferences.edit();
            editor.putString( s: "remember", s1: "true");
            editor.apply();
            Toast.makeText( context: LoginActivity.this, text: "Checked", Toast.LENGTH_SHORT).show();
       }else if (!compoundButton.isChecked()){
            SharedPreferences preferences = getSharedPreferences( name: "checkbox", MODE_PRIVATE);
            SharedPreferences.Editor editor = preferences.edit();
            editor.putString( s: "remember", s1: "false");
            editor.apply();
            Toast.makeText( context: LoginActivity.this, text: "Unchecked", Toast.LENGTH_SHORT).show();
        }
    }
});

```

2. Signup Activity

Το **Signup Activity** είναι υπεύθυνο για την δημιουργία χρηστών στην βάση της εφαρμογής. Στην `onClick` με το Firebase Authentication δημιουργεί στην βάση τον χρήστη μόνο αν πληρεί τα εξής: Ο κωδικός να έχει τουλάχιστον 6 χαρακτήρες και το email να είναι της μορφής `name@domain.tld` με επιτρεπτά characters τα νούμερα και τα σύμβολα εκτός email τύπου `username.@domain.com`, `.user.name@domain.com`, `er-name@domain.com.`, `username@.com`.

Μόνο τότε ο χρηστης προστίθεται στη βάση με το μοναδικό του ID και το email του και του εμφανίζεται το μήνυμα “*SignUp Successful*” σε οποιαδήποτε άλλη περίπτωση παίρνει το μήνυμα “*SignUp Failed*”.

```
public void onClick(View view) {  
  
    String regexPattern = "^(?=.{1,64}@)[A-Za-z0-9\\\\_\\-]+(\\.[A-Za-z0-9\\\\_\\-]+)*@^" + "[^-][A-Za-z0-9\\\\_\\-]+(\\.[A-Za-z0-9\\\\_\\-]+)*(\\.[A-Za-z]{2,})$";  
    String user = signupEmail.getText().toString().trim();  
    String pass = signupPassword.getText().toString().trim();  
    if (user.isEmpty()) {  
        signupEmail.setError("Email cannot be empty");  
    }  
    if (pass.isEmpty()) {  
        signupPassword.setError("Password cannot be empty");  
    } else {  
        auth.createUserWithEmailAndPassword(user, pass)  
            .addOnSuccessListener(authResult -> {  
                // Creates Users  
                String userId = authResult.getUser().getUid();  
                User newUser = new User(user);  
                usersRef.child(userId).setValue(newUser);  
                User.setUserId(userId);  
  
                // Handle successful user registration  
                Toast.makeText(context, SignupActivity.this, text: "SignUp Successful", Toast.LENGTH_SHORT).show();  
                startActivity(new Intent(packageContext, SignupActivity.this, LoginActivity.class));  
            })  
            .addOnFailureListener(e -> {  
                // Handle user registration failure  
                if (!user.matches(regexPattern)) {  
                    Toast.makeText(context, SignupActivity.this, text: "Invalid email", Toast.LENGTH_SHORT).show();  
                } else if (pass.length() < 6) {  
                    Toast.makeText(context, SignupActivity.this, text: "Password must be at least 6 characters", Toast.LENGTH_SHORT).show();  
                } else {  
                    Toast.makeText(context, SignupActivity.this, text: "SignUp failed", Toast.LENGTH_SHORT).show();  
                }  
            });  
    }  
}
```

Επίσης, δημιουργεί και στο Realtime Database το path `Users`. Όταν η εγγραφή είναι πετυχημένη, προστίθεται το email και το ID του χρήστη στη βάση και έπειτα θα μπουν και τα ρούχα, παπούτσια που επιθυμεί ο κάθε χρήστης.

```
1 usage  
private DatabaseReference usersRef= FirebaseDatabase.getInstance().getReference(path: "Users");
```

Η δημιουργία στην συνάρτηση `onClick`:

```
// Creates Users  
String userId = authResult.getUser().getUid();  
User newUser = new User(user);  
usersRef.child(userId).setValue(newUser);  
User.setUserId(userId);
```

Τέλος, αφού γίνει η εγγραφή του μπορεί με το button να επιστρέψει στο **Login Activity** και να κάνει την σύνδεση στην εφαρμογή.

3. Main Activity

Το Main Activity είναι το activity που εμφανίζεται μετά το login του χρήστη στην εφαρμογή. Στο κάτω μέρος του έχει το menu (bottom navigation bar) και είναι υπεύθυνο για τη διαχείριση των 5 fragments (TopsFragment, BottomsFragment, ShoesFragment, CreateFragment, AccountFragment). Στο Main Activity χρησιμοποιείται η συνάρτηση **replaceFragment** για την αντικατάσταση των fragments.

```
private void replaceFragment(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.frame_layout,fragment);
    fragmentTransaction.commit();
}
```

Με τη δημιουργία του Main Activity καλείται η **replaceFragment(new TopsFragment())**; έτσι ώστε η εφαρμογή να ξεκινάει με το TopsFragment. Στη συνέχεια ορίζεται για το **bottomNavigationView** **setOnItemSelectedListener** έτσι ώστε ο χρήστης να μεταφέρεται στο αντίστοιχο fragment όταν πατάει το αντίστοιχο εικονίδιο στο menu.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMainBinding.inflate(LayoutInflater());
    setContentView(binding.getRoot());

    replaceFragment(new TopsFragment());

    binding.bottomNavigationView.setOnItemSelectedListener(item -> {

        int id = item.getItemId();

        if (id == R.id.tops)
            replaceFragment(new TopsFragment());
        else if (id == R.id.bottoms)
            replaceFragment(new BottomsFragment());
        else if (id == R.id.shoes)
            replaceFragment(new ShoesFragment());
        else if (id == R.id.create)
            replaceFragment(new CreateFragment());
        else if (id == R.id.account)
            replaceFragment(new AccountFragment());

        return true;
});

}
```

4. Upload Tops Activity

Στο συγκεκριμένο activity ο χρήστης μπορεί να επιλέξει και να ανεβάσει στον λογαριασμό του μια φωτογραφία από κάποιο μπλουζάκι που έχει στη συλλογή του, πατώντας πάνω στο εικονίδιο και στη συνέχεια το ανεβάζει είτε με κάποια περιγραφή είτε χωρίς.

Το upload γίνεται με τη χρήση της κλάσης **UploadActivity**. Σε αυτή τη κλάση γίνεται η σύνδεση με τη βάση, σχετικά με τις εικόνες για τα μπλουζάκια που ανεβάζει κάθε χρήστης. Επομένως χρησιμοποιώντας το **userId** οι εικόνες αποθηκεύονται στο path TopsSlider του κάθε χρήστη έχοντας έτσι ο κάθε ένας τις προσωπικές του εικόνες.

```
String userId = currentUser.getUid();
1 usage
FirebaseDatabase database = FirebaseDatabase.getInstance();
// Get the reference to the current user's node
1 usage
DatabaseReference userRef = database.getReference( path: "Users").child(userId);

// Create a new node inside the current user's node
2 usages
DatabaseReference databaseReference = userRef.child( pathString: "TopsSlider");
```

Έχουμε δημιουργήσει τη συνάρτηση **onActivityResult** η οποία ελέγχει για αρχή αν έχει επιλεχθεί κάποια εικόνα από τη συλλογή. Διαφορετικά αν ο χρήστης πάει στη συλλογή αλλά επιστρέψει στη κύρια εφαρμογή χωρίς να επιλέξει κάποια εικόνα ή χωρίς να αλλάξει την ήδη υπάρχον εικόνα, του εμφανίζεται το μήνυμα “No image”.

```
public void onActivityResult(ActivityResult result) {
    if(result.getResultCode()== Activity.RESULT_OK){
        Intent data=result.getData();
        imageUri=data.getData();
        uploadImage.setImageURI(imageUri);

    }else {
        Toast.makeText( context: UploadActivity.this, text: "No Image",Toast.LENGTH_SHORT).show();
    }
}
```

Έχουμε δημιουργήσει επίσης τη συνάρτηση **uploadImage.setOnClickListener** στην οποία ελέγχουμε αν έχει επιλεχθεί κάποια εικόνα από τον χρήστη. Στη περίπτωση που ο χρήστης έχει επιλέξει κάποια φωτογραφία η διαδικασία του upload συνεχίζεται κανονικά. Ωστόσο αν επιδιώξει να κάνει upload χωρίς να έχει επιλέξει κάποια φωτογραφία για το αντίστοιχο activity τότε του εμφανίζεται το μήνυμα “select image” χωρίς να εκτελείται κάποιο upload.

```

uploadButton.setOnClickListener(new View.OnClickListener() {
    @KallioPi Pliogka
    @Override
    public void onClick(View v) {
        if(imageUri !=null){
            uploadToFirebase(imageUri);
        }else{
            Toast.makeText(context: UploadActivity.this, text: "Select image",Toast.LENGTH_SHORT).show();
        }
    }
});

```

Ωστόσο χρησιμοποιώντας τη συνάρτηση `onSuccess` γίνεται το upload. Το upload μπορεί να γίνει είτε έχουμε κάποια περιγραφή του ρούχου είτε όχι πατώντας το βελάκι που υπάρχει κάτω δεξιά και αν το upload γίνει με επιτυχία τότε θα εμφανιστεί το μήνυμα “Uploaded” και θα αποθηκευτούν όλα τα καταλληλα δεδομένα, όπως η περιγραφή, και στη συνέχεια θα μεταφερθούμε στη MainActivity.

```

@Override
public void onSuccess(Uri uri) {
    DataClass dataClass=new DataClass(uri.toString(),caption);
    String key=databaseReference.push().getKey();
    databaseReference.child(key).setValue(dataClass);
    progressBar.setVisibility(View.INVISIBLE);
    Toast.makeText(context: UploadActivity.this, text: "Uploaded",Toast.LENGTH_SHORT).show();
    Intent intent= new Intent(packageContext: UploadActivity.this,MainActivity.class);
    startActivity(intent);
    finish();
}

```

Ωστόσο αν υπάρξει κάποιο σφάλμα στη διαδικασία θα εμφανιστεί το μήνυμα “failed” χρησιμοποιώντας τη συνάρτηση `onFailure`.

```

@Override
public void onFailure(@NonNull Exception e) {
    progressBar.setVisibility(View.VISIBLE);
    Toast.makeText(context: UploadActivity.this, text: "failed",Toast.LENGTH_SHORT).show();
}

```

Σε κάθε περίπτωση γίνεται η κατάλληλη ενημέρωση στο firebase.

5. Upload Bottoms Activity

Στο συγκεκριμένο activity ο χρήστης μπορεί να επιλέξει και να ανεβάσει στον λογαριασμό του μια φωτογραφία από κάποιο παντελόνι/φούστα/σορτς που έχει στη συλλογή του, πατώντας πάνω στο εικονίδιο και στη συνέχεια το ανεβάζει είτε με κάποια περιγραφή είτε όχι. Το upload γίνεται με αντίστοιχο τρόπο με τη διαφορά ότι οι εικόνες αποθηκεύονται στο node BottomsSlider του κάθε χρήστη στο firebase.

6. Upload Shoes Activity

Στο συγκεκριμένο activity ο χρήστης μπορεί να επιλέξει και να ανεβάσει στον λογαριασμό του μια φωτογραφία από κάποιο παπούτσι που έχει στη συλλογή του, πατώντας πάνω στο εικονίδιο και στη συνέχεια το ανεβάζει είτε με κάποια περιγραφή είτε όχι.

Το upload γίνεται με αντίστοιχο τρόπο με τη διαφορά ότι οι εικόνες αποθηκεύονται στο node ShoesSlider του κάθε χρήστη στο firebase.

Περιγραφή των 5 fragments

1. Tops Fragment

Αυτό το fragment είναι υπεύθυνο για την εμφάνιση των φωτογραφιών των χρηστών.

Οι εικόνες ταξινομούνται με τη χρήση **RecyclerView**. Εμφανίζει τις μπλούζες που οι χρήστες έχουν επιλέξει ότι θέλουν να έχουν στη ντουλάπα τους. Οι φωτογραφίες εμφανίζονται με τη σειρά που τις έχει επιλέξει ο χρήστης και με την περιγραφή. Επιπλέον δίνει τη δυνατότητα πατώντας το κουμπί “+” που βρίσκεται κάτω δεξιά να μεταφερθεί στο **Upload tops activity** για την εισαγωγή νέας εικόνας.

```
@Override  
public void onClick(View view) {  
    Intent intent = new Intent(getActivity(),UploadActivity.class);  
    startActivity(intent);  
    /* getActivity().finish(); */  
}
```

Τέλος ο χρήστης μπορεί να διαγράψει μια φωτογραφία με τη χρήση της συνάρτησης **onDeleteClick** πατώντας παρατεταμένα πάνω της, έτσι παίρνουμε τη θέση και το κλειδί της αντίστοιχης εικόνας και την αφαιρούμε από το firebase. Αν η διαγραφή πραγματοποιηθεί με επιτυχία εμφανίζει το μήνυμα “Image deleted”.

```
@Override  
public void onDeleteClick(int position) {  
    DataClass selected=dataList.get(position);  
    String selectedKey=selected.getKey();  
    StorageReference img=mStorage.getReferenceFromUrl(selected.getImageURL());  
  
    img.delete().addOnSuccessListener(new OnSuccessListener<Void>() {  
  
        @Override  
        public void onSuccess(Void unused) {  
            databaseReference.child(selectedKey).removeValue();  
            Toast.makeText(getActivity(), text: "Image deleted",Toast.LENGTH_SHORT).show();  
        }  
    });  
}
```

2. Bottoms Fragment

To Bottoms fragment λειτουργεί με αντίστοιχο τρόπο όπως αυτό του Tops fragment.

3. Shoes Fragment

To Shoes fragment λειτουργεί με αντίστοιχο τρόπο όπως αυτό του Tops fragment.

4. Create Fragment

To Create Fragment εμφανίζει τα ρούχα του χρήστη σε 3 image sliders, ώστε να μπορεί να τα συνδυάσει και να δημιουργήσει outfits. Για τα image sliders χρησιμοποιήθηκε το [“Why Not! Image Carousel!”](#). Αρχικά, δημιουργείται ένα DatabaseReference userRef το οποίο αναφέρεται στον κόμβο του Realtime Database με το id του τρέχοντα χρήστη.

```
// Get the reference to the current user's node
DatabaseReference userRef = database.getReference( path: "Users").child(userId);
```

Στη συνέχεια, κάθε slider παίρνει τις φωτογραφίες που είναι αποθηκευμένες στη βάση για τον τρέχοντα χρήστη. Για παράδειγμα, το image slider που είναι για τα παπούτσια θα πάρει τις φωτογραφίες που είναι στον κόμβο “ShoesSlider” στη βάση, όπως φαίνεται παρακάτω.

```
// ----- firebase why not image carousel Shoes -----
ImageCarousel carousel3 = view.findViewById(R.id.carousel3);

// Register lifecycle. For activity this will be lifecycle/getLifecycle() and for fragments it will be
carousel3.registerLifecycle(getLifecycle());

List<CarouselItem> list3 = new ArrayList<>();

userRef.child( pathString: "ShoesSlider")
    .addSingleValueEvent(new ValueEventListener() {
        2 usages    ± alexia-nt +1
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for(DataSnapshot data: dataSnapshot.getChildren()) {
                list3.add(new CarouselItem(data.child( path: "imageURL").getValue().toString()));
            }
            carousel3.setData(list3);
        }

        ± alexia-nt
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(getActivity(), text: "Failure",Toast.LENGTH_LONG).show();
        }
    });
});
```

5. Account Fragment

Αυτό το fragment είναι υπεύθυνο για το logout των χρηστών.

Εμφανίζει το email του χρήστη και το button που τον πηγαίνει στην **Login Activity**. Επίσης, θέτει το remember me checkbox σε false, δηλαδή unchecked αφού ο χρήστης επιθυμεί να αποσυνδεθεί.

```
// Clears data from intent and unchecks the remember me button
    + user1
    logout.setOnClickListener(new View.OnClickListener() {
        + user1
        @Override
        public void onClick(View view) {
            Sharedpreferences preferences = getActivity().getSharedpreferences(name: "checkbox", Context.MODE_PRIVATE);
            Sharedpreferences.Editor editor = preferences.edit();
            editor.putString(s: "remember", s1: "false");
            editor.apply();
            Intent intent = new Intent(getActivity(), LoginActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            getActivity().finish();
        }
    });
}
```

Περιγραφή των επιπλέον κλάσεων

1. User

Είναι μια βοηθητική κλάση που αναπαριστά έναν user. Ως χαρακτηριστικά έχει το email και το ID του χρήστη και αντίστοιχα περιέχει setters και getters για αυτά. Με την χρήση του δημιουργούμε το path του χρήστη στο Realtime Database.

```
public class User {
    // Class with the authenticated users
    3 usages
    private String email;
    1 usage
    private static String userId;

    //Getters and Setters
    + user1
    public String getEmail() { return email; }
    1 usage
    private static String userId;
    1 usage + user1
    public static void setUserId(String userId) {
        User.userId = userId;
    }
    + user1
    public static String getUserId() { return userId; }
    + alexia-nt +1
    public void setEmail(String email) { this.email = email; }
    1 usage + user1
    public User(String email) { this.email = email; }
}
```

2. My Adapter

Η κλάση αυτή αποτελεί τον προσαρμογέα και κάνει extend το RecyclerView.Adapter<MyAdapter.MyViewHolder>. Η κλάση αυτή χρησιμοποιείται για την προβολή των εικόνων μέσω του RecyclerView..

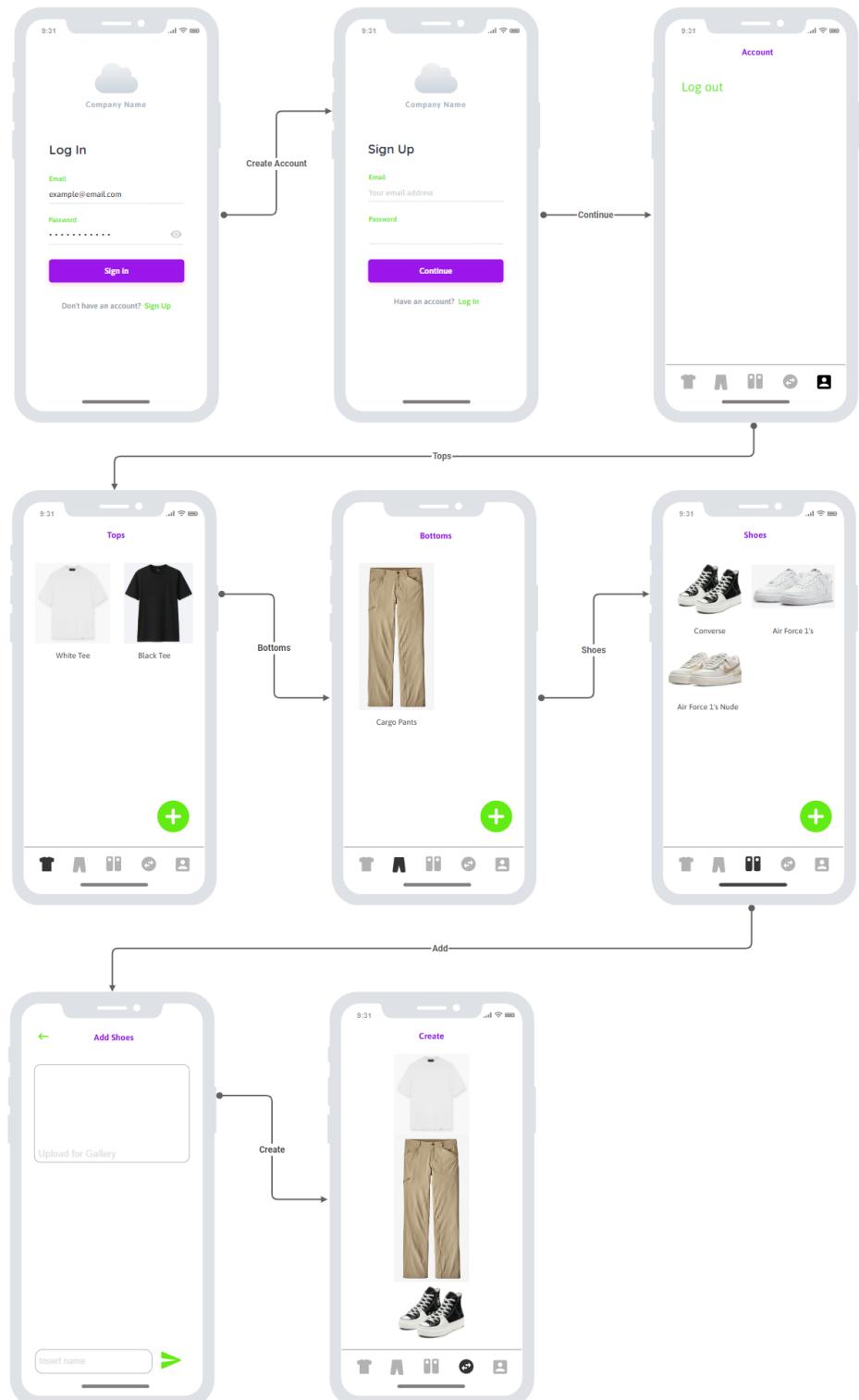
3. DataClass

Στη κλάση αυτή έχουν δημιουργηθεί οι κατάλληλες συναρτήσεις για να αποθηκεύουμε ή να παίρνουμε κάποια δεδομένα για το upload και την εμφάνιση (retrieve) των εικόνων.

```
public class DataClass {  
    2 usages  
    private String key;  
    3 usages  
    private String imageURL;  
    3 usages  
    private String caption;  
    ↳ Kallioopi Pliogka  
    public DataClass(){  
  
    }  
    3 usages ↳ Kallioopi Pliogka  
    @Exclude  
    public String getKey() { return key; }  
    3 usages ↳ Kallioopi Pliogka  
    @Exclude  
    public void setKey(String key) { this.key = key; }  
    4 usages ↳ Kallioopi Pliogka  
    public String getImageURL() { return imageURL; }  
    ↳ Kallioopi Pliogka  
    public void setImageURL(String imageURL) { this.imageURL = imageURL; }  
    1 usage ↳ Kallioopi Pliogka  
    public String getCaption() { return caption; }  
    ↳ Kallioopi Pliogka  
    public void setCaption(String caption) { this.caption = caption; }  
    3 usages ↳ Kallioopi Pliogka  
    public DataClass(String imageURL, String caption) {  
        this.imageURL = imageURL;  
        this.caption = caption;  
    }  
}
```

Mockup Design

Πριν την ανάπτυξη της εφαρμογής, σχεδιάσαμε ένα mockup design και είμαστε πολύ ευχαριστημένες με το τελικό αποτέλεσμα της εφαρμογής, καθώς αντικατοπτρίζει το αρχικό μας σχέδιο για το πως θα έπρεπε να είναι η εφαρμογή και τι λειτουργίες θα έχει.





Logo Designed by katemangostar / Freepik

Ομάδα:

Λάλλου Ιουλία (3956)

Νταντουρή Αλεξία (3871)

Πλιόγκα Καλλιόπη (3961)