**BayanNails** – nails that aren’t just tales.



מגישים: סרגיי מורוזוב ועבדאללה מסרי

ת.ז: 302979687 , 321320384

שם המנחה: איתמר פלדמן

תאריך: "14.07.23"

*פרק ראשון – תיאור התוכנה ומטרותיה*

**מהות האפליקציה:**

שמה הגנרי של האפליקצייה הינו – **ניהול תורים/הזמנות של לקוחות**.  
**Que Management System**

הרעיון הוא לקחת את הבסיס של אפליקציה לניהול תורים ו"להלביש אותו" לצרכים של לקוחות שונים.  
בדוגמא הנ"ל "הלבשנו" את האפליקציה על זימון תור לסלון ציפורניים – ומנגד, אנו יכולים ל"הלביש" את אותו קוד הבסיס למשל למספרה, לעשות שינויים מינוריים וככה לחסוך בכסף, זמן, משאבים.  
לכן לכרגע היא נקראת BayanNails על שם הלקוח.  
  
כפי שניתן להבין, המהות עצמה היא זימון תור לשירות כלשהו(במקרה שלנו סלון ציפורניים) והנגשת השירותים דרך האפליקציה.   
☘הלקוח יוכל להנות מזימון תור מהיר, ניווט למקום, תקשורת מהירה וכו'.  
☘הבעלים יוכל להנות מתפעול מהיר, מערכת ניהול תורים אוטומטית, העלאת תמונות וכו'.

**תיאור האופציות שהתוכנה נותנת למשתמש:**

בכניסת המשתמש לאפליקציה, יופיע מסך פתיחה שיאפשר למשתמשים חדשים להירשם ולמשתמשים קיימים להתחבר.  
ההירשמות תהיה ע"י שם משתמש וסיסמה בלבד(כרגע).  
בהתחברות יוצג הדף או של האדמין או של הלקוח(נתון לפי זיהוי המשתמש).  
לאחר ביצוע ההזדהות:  
האדמין:  
☘יוכל לראות את כל התורים שנקבעו אליו.  
☘יוכל להעלות ולמחוק תמונות ל/מ ה/גלריה.

הלקוח:  
☘ יוכל לקבוע תאריך לביקור ע"י בחירת תאריך ושעה, לבטל תור קיים, לראות תור קיים.  
☘ יוכל לתקשר עם נותן השירות דרך ווצאפ, פייסבוק, אינסטגרם וכו'.  
☘יוכל לנווט למיקום בית העסק.  
☘יוכל להנות מגלריית התמונות המרהיבה של בית העסק.  
(בהמשך מתוכננת האופצייה לרכישות דרך האפליקציה ועוד פיצ'רים מעניינים).

**דרישות להפעלת התוכנה:**

♦פלאפון עם מערכת הפעלה אנדרואיד 5.0 (21 API ) ומעלה.  
♦גישה לאינטרנט.

*פרק שני – מדריך למפתח*

**מחלקות בפרויקט:**

**תיאור מחלקת :MainPage**

היא פעילות שמשמשת כמסך הראשי של אפליקציית ,BayanNails היא מכילה אלמנטים ותכונות שונות של ממשק משתמש ותפקודיות להתקשרות עם המשתמש וביצוע פעולות כגון הצגת מידע משתמש, הצגת הזמנות, ניווט לפעילויות שונות והתקשרות עם שירותים חיצוניים כמו Instagram, Facebook ו-WhatsApp.

♠ *חתימת המחלקה:* java public class MainPage extends AppCompatActivity   
♠ *שדות:*

**String myNumber**: משתנה מחרוזת המאחסן את מספר הטלפון של המשמש.

**List<Order> orderList**: רשימה לאחסון הזמנות.

**OrderService orderService**: משמש לטיפול בפעולות הקשורות להזמנות.

♠ *שיטות:*

onCreate(Bundle savedInstanceState): שיטה זו נקראת כאשר הפעילות נוצרת. היא מגדירה את

תצורת התצוגה, מאתחלת אלמנטים של ממשק משתמש, מתחברת ל-OrderService ומגדירה מאזין ללחצנים שונים(אונקליקים).

**showDateTimePicker():** שיטה להצגת תיבת תאריך ושעה כשהמשתמש רוצה להוסיף הזמנה חדשה.

**getUserOrder(String userName):** שיטה לאחזור והצגת הזמנת המשתמש ממסד הנתונים.

**deleteOrder(String userName):** שיטה למחיקת הזמנת המשתמש ממסד הנתונים.

**updateOrder(String userName, Order newOrder):** שיטה לעדכון הזמנת המשתמש במסד הנתונים.

**addOrder(Order order):** שיטה להוספת הזמנה חדשה למסד הנתונים.

**onDestroy():** שיטה זו נקראת כאשר הפעילות מושמדת.

**serviceConnection**: ServiceConnection .משמש להתחברות והתנתקות מה

**תיאור מחלקת LoginActivity:**

מחלקת LoginActivity היא פעולה המתמודדת עם הזדהות המשתמש ופונקציות ההתחברות. היא מתקשרת עם מסד הנתונים בזמן-אמת של Firebase כדי לבדוק אימות משתמש ולהעניק גישה לאפליקציה.

♠ *חתימת המחלקה:* public class LoginActivity extends AppCompatActivity

♠ *שדות:*

**private DatabaseReference userRef**: ייחוס לצומת "משתמשים" במסד הנתונים זמן-אמת של פיירבייס. משמש לשאילתה ולאחזור פרטי המשתמש.

private SharedPreferences sharedPreferences: משמש לאחסון ואיחזור נתוני המשתמש מקומית.

♠ *שיטות:*

onCreate(Bundle savedInstanceState): שיטה זו נקראת בעת יצירת הפעילות. היא מאתחלת את האלמנטים של הממשק משתמש, מגדירה האזנה ללחיצות על כפתורים ומאחזרת את מופע SharedPreferences. היא גם מאתחלת את DatabaseReference לנתוני המשתמש במסד הנתונים של Firebase.

onClick(View v): שיטה זו מתמודדת עם אירועי הלחיצה על כפתור "אישור". היא מאחזרת את שם המשתמש והסיסמה שהוזנו, בודקת האם המשתמש הוא מנהל וממשיכה לפעילות המנהל אם הפרטים מתאימים. אם לא, היא בודקת האם הפרטים מתאימים לערכים שמאוחסנים בSharedPreferences וממשיכה לפעילות המנהל אם הם תואמים. אם הפרטים אינם תואמים, היא מבצעת שאילתה למסד הנתונים של firebase כדי לבדוק האם המשתמש קיים. אם המשתמש קיים, היא מחזירה את אובייקט המשתמש וממשיכה לפעילות הדף הראשי.

אם המשתמש לא קיים, היא מציגה הודעת Toast ונוהגת לפעילות ההרשמה.

onClick(View v): שיטה זו מתמודדת עם אירוע הלחיצה על כפתור "הרשמה". היא מנווטת לפעילות ההרשמה.

**תיאור מחלקת Admin\_Activity:**

מחלקת Admin\_Activity אחראית לניהול הפונקציות המנהליות באפליקצייה BayanNails היא מספקת ממשק למנהלים להצגת וניהול הזמנות, הוספת תמונות לגלריה וניווט לדף הראשי.

♠ *שדות:*

**private List<Order> orders**: רשימה לאחסון ההזמנות שמתקבלות ממסד הנתונים בזמן-אמת של Firebase.

**private ListView orderListView**: ListViewלהצגת רשימת ההזמנות מ .

**private ArrayAdapter<Order> orderAdapter**: אדפטר למילוי אורדרליסט עם נתוני הזמנה.

**private ImageView iv\_uploadPic**: ImageView להצגת התמונה שהועלתה מ.

**private TextView tv**: מרכיב TextView להצגת כתובת ה-URL של התמונה שהועלתה.

**private StorageReference storageRef**:פיירבייס עבור העלאה ואחזור תמונות. ייחוס לאחסון של

**private DatabaseReference picRefDB**: ייחוס לצומת "pics" במסד הנתונים הזמן-אמת של Firebase לאחסון כתובות ה-URL של התמונות.

♠ *שיטות:*

onCreate(Bundle savedInstanceState): שיטה זו נקראת בעת יצירת הפעילות. היא מאתחלת את אלמנטי הממשק, מגדירה את ListView ו־ArrayAdapter עבור ההזמנות ומאחזרת ייחוסים לאחסון ה-Firebase ולמסד הנתונים הזמן-אמת. היא גם מגדירה האזנה ללחיצה על כפתורי "הוספה לגלריה" ו"מעבר לדף הראשי".

onDataChange(DataSnapshot dataSnapshot): שיטה זו מופעלת כאשר נתוני ההזמנות במסד הנתונים הזמן-אמת של Firebase משתנים. היא מאחזרת את נתוני ההזמנות המעודכנים ומעדכנת את ה orderAdapter כדי לרענן את ה ListView.

onCancelled(DatabaseError databaseError): שיטה זו נקראת כאשר קורת שגיאה במסד הנתונים במהלך אחזור נתוני ההזמנות.

onActivityResult(int requestCode, int resultCode, Intent data): שיטה זו נקראת כאשר הפעילות מקבלת תוצאה מפעילות אחרת, כגון פעילות המצלמה ללכידת תמונה. היא מאחזרת את התמונה שנלכדה, מציגה אותה ב ImageView ומאחסנת אותה באחסון ה-Firebase.

saveBitmapToStorage(Bitmap bitmap):שיטה זו מאחסנת תמונת bitmap באחסון ה-Firebase. היא מדחיסה את התמונה, ממירה אותה לביטים, מייצרת שם קובץ ייחודי ומעלה אותה לתיקיית "userImage" באחסון. היא גם מאחסנת את כתובת ה-URL להורדה של התמונה בצומת "pics" במסד הנתונים הזמן-אמת.

**תיאור מחלקת SignUp\_Activity:**

מחלקת SignUp\_activity היא פעולה המתמודדת עם תהליך הרישום של המשתמש. היא מאפשרת למשתמשים להזין את הפרטים האישיים שלהם, כגון שם, שם משפחה, כתובת דוא"ל, מספר טלפון, סיסמה ושם משתמש. המחלקה מאמתת את הקלט של המשתמש ושומרת את מידע המשתמש במסד הנתונים של Firebase Realtime אם כל הקלטים/נתונים תקינים.  
♠ *שדות:*

etName: תצוגת EditText להזנת שם הפרטי של המשתמש.

etFamilyName: תצוגת EditText להזנת שם המשפחה של המשתמש.

etEmail: תצוגת EditText להזנת כתובת הדוא"ל של המשתמש.  
 etPhone: תצוגת EditText להזנת מספר הטלפון של המשתמש.

etPassword: תצוגת EditText להזנת הסיסמה של המשתמש.

etUser\_Name: תצוגת EditText להזנת שם המשתמש הרצוי של המשתמש.

btnSignUp: כפתור Button להתחלת תהליך הרישום.

userRef: הפניה לצומת מסד הנתונים של Firebase Realtime בשם "users" שבו מאוחסן מידע המשתמשים.

♠ *שיטות:*

onCreate(Bundle savedInstanceState): שיטה זו נקראת כאשר נוצרת הפעילות. היא מגדירה את תצורת הפעילות, מאתחלת את אלמנטי ממשק המשתמש, ומגדירה את מאזין הלחיצה על כפתור ההרשמה.

validateInputs() שיטה זו מאמתת את קלטי המשתמש ומבצעת בדיקות מתאימות אם הנתונים שהוזנו הינם נכונים. מייצרת מפתח מיוחד בעזרת פוש.גט-קי ושומרת את האובייקט במסד הנתונים זמן-אמת, מציגה הודעת ברוך הבא למשתמש ומנווטת לMainPage.

**תיאור מחלקת MapsActivity:**

מחלקת**MapsActivity** היא פעילות ביישום אנדרואיד שמתקנת את תצוגת המפה ומתחילה את הניווט ליעד ספציפי באמצעות Google Maps. היא כוללת גם פונקציות לבדיקת רמת הסוללה ותצוגת הודעת אזהרה אם היא נמוכה.

♠ *שדות:*

**DESTINATION\_LATITUDE**: ערך קבוע מסוג double שמייצג את קו הרוחב של היעד.

**DESTINATION\_LONGITUDE**: ערך קבוע מסוג double שמייצג את קו האורך של היעד.

**batteryLowReceiver**: מופע של המחלקה BroadcastReceiver לטיפול באירועים של סוללה נמוכה.

♠ *שיטות:*

onCreate(Bundle savedInstanceState):

שיטה זו נקראת כאשר הפעילות נוצרת. היא מאתחלת את תצוגת הפעילות, מגדירה את מקבל האירועים של סוללה נמוכה, ומרשימה אותו לקבל אירועי שינוי ברמת הסוללה. היא גם בודקת את רמת הסוללה ומתחילה את הניווט אם רמת הסוללה גבוהה מ-15%. אחרת, היא מציגה הודעת toast להזהרה למשתמש לטעון את הסוללה ומסיימת את הפעילות.

onDestroy():

שיטה זו נקראת כאשר הפעילות נמחקת. היא מבטלת את מקבל האירועים של סוללה נמוכה כדי למנוע דליפות זיכרון ולטפל במשאבים.

startNavigation():

שיטה זו נקראת כדי להתחיל את הניווט לקואורדינטות היעד המוגדרות מראש. היא יוצרת **Uri** עם קו הרוחב והאורך של היעד ויוצרת intent עם פעולת הניווט של Google Maps. ה-intent מוגדר להפעיל את אפליקציית Google Maps ולהתחיל את הניווט.

**תיאור מחלקת Gallery\_Activity:**

מחלקת Gallery\_Activity מחזירה נתוני תמונות מבסיס הנתונים הזמן-אמת של Firebase ומאפשרת תצוגה של התמונות ב-RecyclerView.

♠ *שדות:*

**recyclerView**: אובייקט RecyclerView לתצוגת התמונות.

**imageAdapter**: אובייקט ImageAdapter המשמש כאדפטר ל-RecyclerView.

**picRefDB**: אובייקט DatabaseReference המתייחס לצומת "pics" בבסיס הנתונים הזמן-אמת של Firebase.

♠ *שיטות:*

onCreate(Bundle savedInstanceState)

שיטה זו נקראת בעת יצירת הפעילות. היא מאתחלת את פריסת הפעילות, מגדירה את RecyclerView ו-ImageAdapter ומבצעת אנימציה על RecyclerView. היא גם מביאה נתוני תמונות מבסיס הנתונים הזמן-אמת של Firebase וממלאה את RecyclerView בתמונות.

onDataChange(DataSnapshot snapshot)

זו גם קולבק המופעל כאשר הנתונים בצומת המסד נתונים הזמן-אמת משתנים. היא מביאה את כתובות ה-URL של התמונות מתוך ה-DataSnapshot ומעדכנת את ה-ImageAdapter עם הנתונים החדשים. היא גם מבצעת רענון תצוגה ל-RecyclerView כדי לשדרג את השינויים.

onCancelled(DatabaseError error)

שיטת קולבק המופעלת כאשר תהליך איסוף הנתונים בוטל עקב שגיאה.

**תיאור מחלקת ImageAdapter:**

המחלקה ImageAdapter אחראית לקישור URL של תמונה ל ImageView בתוך RecyclerView. היא מרחיבה את המחלקה RecyclerView.Adapter ומשתמשת בתבנית ViewHolder כדי לאפשר יצירת ומיחזור מתצוגות באופטימיזציה.

♠ *בנאי:*

ImageAdapter(): יוצר מופע חדש של המחלקה ImageAdapter.

♠ *שיטות:*

setData(List<String> imageUrls): מגדיר את כתובות ה-URL של התמונות שיש להציג ב-

RecyclerView.

onCreateViewHolder(ViewGroup parent, int viewType): ViewHolder: מפעילה את התצוגה עבור כל פריט ב-RecyclerView ומחזירה מופע חדש של מחלקת ViewHolder.

onBindViewHolder(ViewHolder holder, int position): מקשרת את כתובת ה-URL של התמונה במיקום הנתון ל-ImageView המתאים בתוך ViewHolder.

getItemCount(): int: מחזירה את מספר הפריטים הכולל ב-RecyclerView.

מחלקה פנימית: ViewHolder

ImageView imageView: מחזיקה ייחודית ל-ImageView להצגת התמונה.

**תיאור מחלקת Order:**

המחלקה Order מייצגת הזמנה עם מידע כגון היום, החודש, השנה, השעה והמשתמש. זו מחלקת נתונים פשוטה עם getters, setters ושיטת toString.

♠ *בנאי:*

Order(int day, int month, int year, int hour): יוצר מופע חדש של מחלקת Order עם היום, החודש, השנה והשעה המצויינים.

Order(): יוצר מופע חדש של מחלקת Order עם ערכי בררת המחדל.

♠ *שדות:*

int day: היום של ההזמנה.

int month: החודש של ההזמנה.

int year: השנה של ההזמנה.

int hour: השעה של ההזמנה.

User user: המשתמש הקשור להזמנה.

♠ *שיטות:*

Getters ו-setters עבור כל מאפיין.

toString(): String: מחזירה מחרוזת המייצגת את אובייקט ההזמנה.

**תיאור מחלקת OrderService:**

המחלקה OrderService היא שירות המספק שיטות לאינטראקציה עם הזמנות שמאוחסנות במסד נתונים בזמן אמת של Firebase. היא מרחיבה את המחלקה Service ומספקת ערך binder מקומי לגישה לשירות.

**מחלקה פנימית: LocalBinder OrderService getService(): מחזירה מופע של מחלקת OrderService**

♠ *שיטות:*

onBind(Intent intent): IBinder: מחברת את השירות ללקוח ומחזירה מופע של מחלקת LocalBinder

getUserOrder(String userName, ValueEventListener listener): מחזירה את ההזמנה עבור משתמש ספציפי ממסד הנתונים בזמן אמת של Firebase

deleteOrder(String userName, DatabaseReference.CompletionListener listener): מוחקת את ההזמנה עבור משתמש ספציפי ממסד הנתונים בזמן אמת של Firebase.

updateOrder(String userName, Order newOrder, DatabaseReference.CompletionListener listener): מעדכנת את ההזמנה עבור משתמש ספציפי במסד הנתונים בזמן אמת של Firebase.

addOrder(Order order, DatabaseReference.CompletionListener listener): מוסיפה הזמנה חדשה למסד הנתונים בזמן אמת של Firebase.

**תיאור מחלקת User:**

המחלקה User מייצגת משתמש עם מידע כגון שם פרטי, שם משפחה, דוא"ל, סיסמה, מספר טלפון, שם משתמש ורשימה של הזמנות. זו מחלקת נתונים פשוטה עם getters ו-setters.

♠ *בנאי:*

User(): יוצר מופע חדש של מחלקת User עם ערכי בררת מחדל.

User(String firstName, String lastName, String email, String password, String phoneNumber, String userName, List<Order> order): יוצר מופע חדש של מחלקת User עם הערכים המצויינים.

♠ *שדות:*

String firstName: שם הפרטי של המשתמש.  
String lastName: שם המשפחה של המשתמש.

String email: הדוא"ל של המשתמש.

String password: הסיסמה של המשתמש.

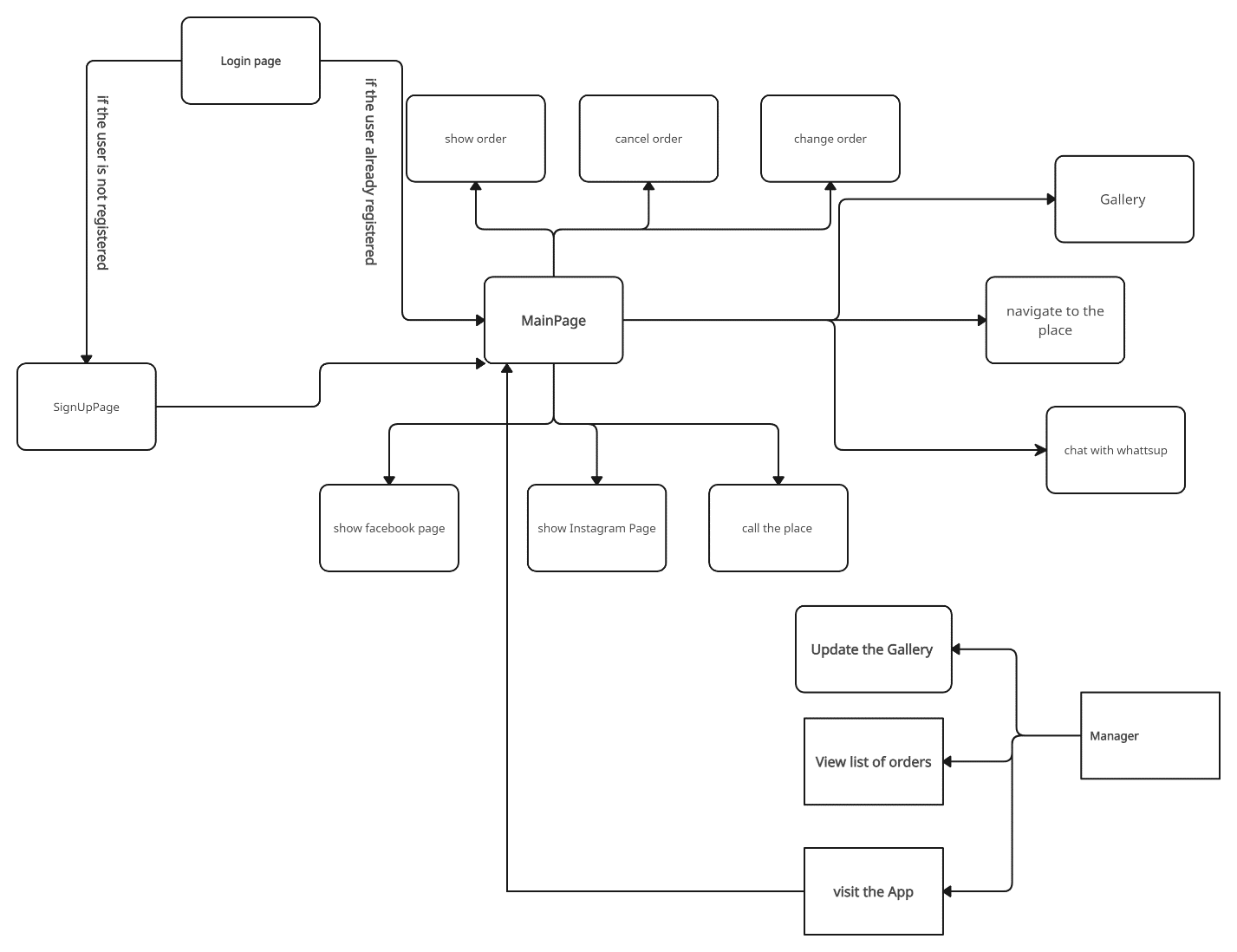
String phoneNumber: מספר הטלפון של המשתמש.

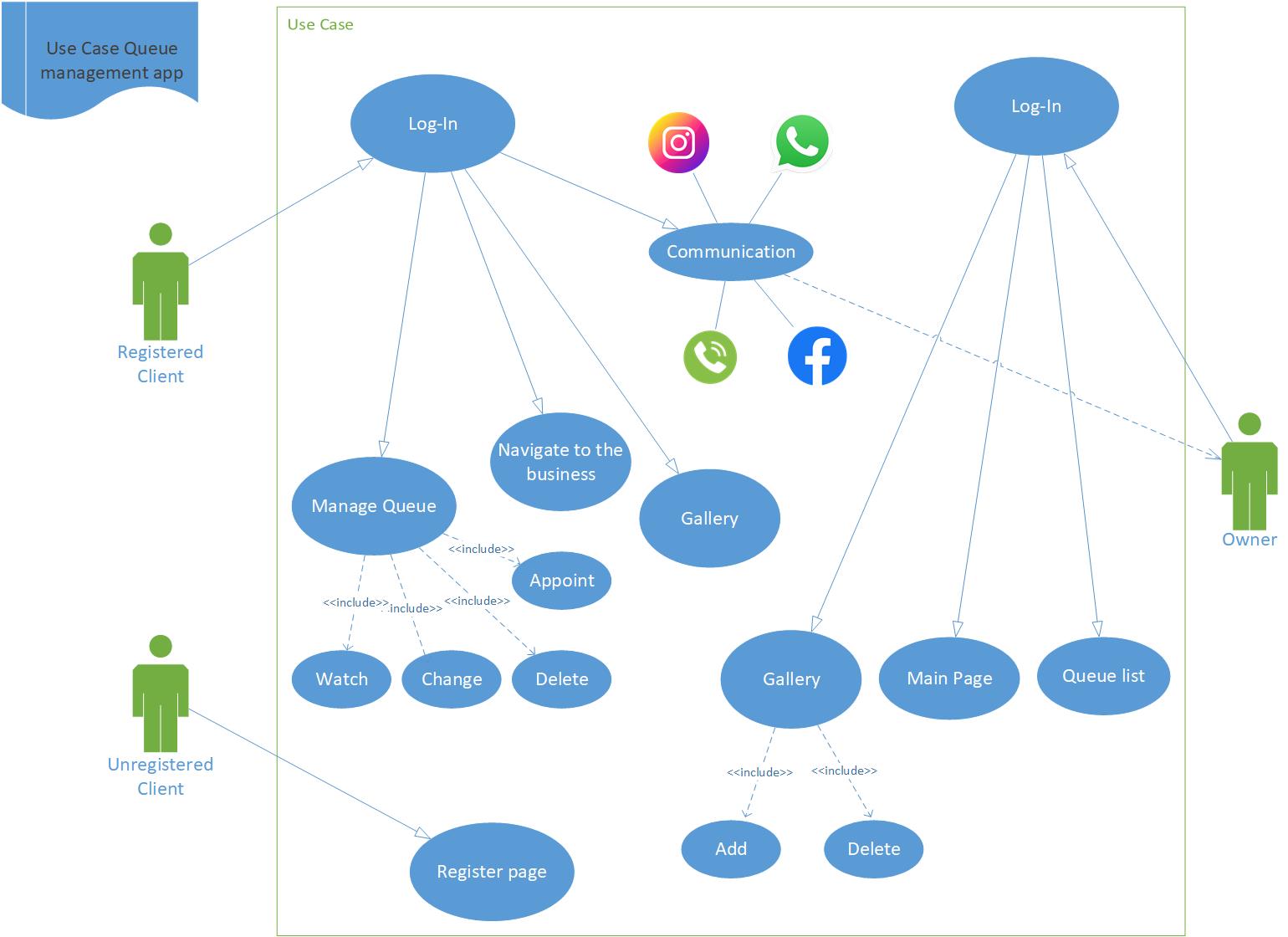
String userName: שם המשתמש של המשתמש.

List<Order> order: רשימת ההזמנות הקשורות למשתמש.

♠ *שיטות:*

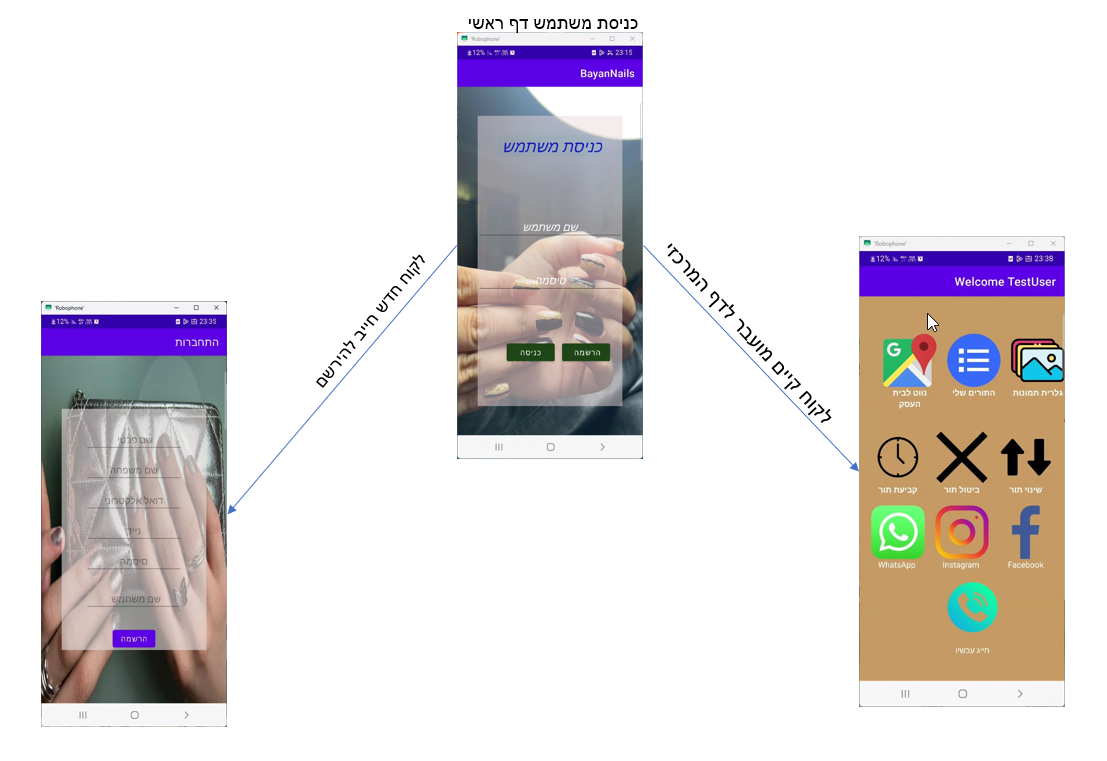
Getters and Setters.

**תרשים לוגי של האפליקציה : ****

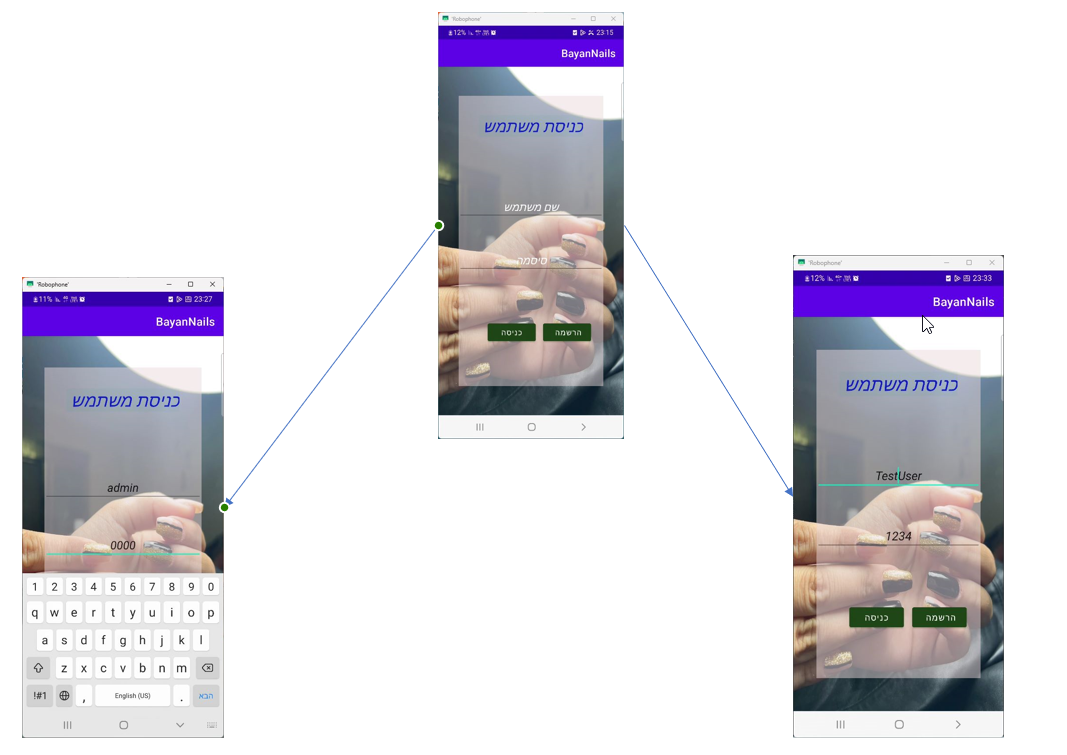
**Use Case:  
**

***אופן השימוש במסכי האפליקציה:***

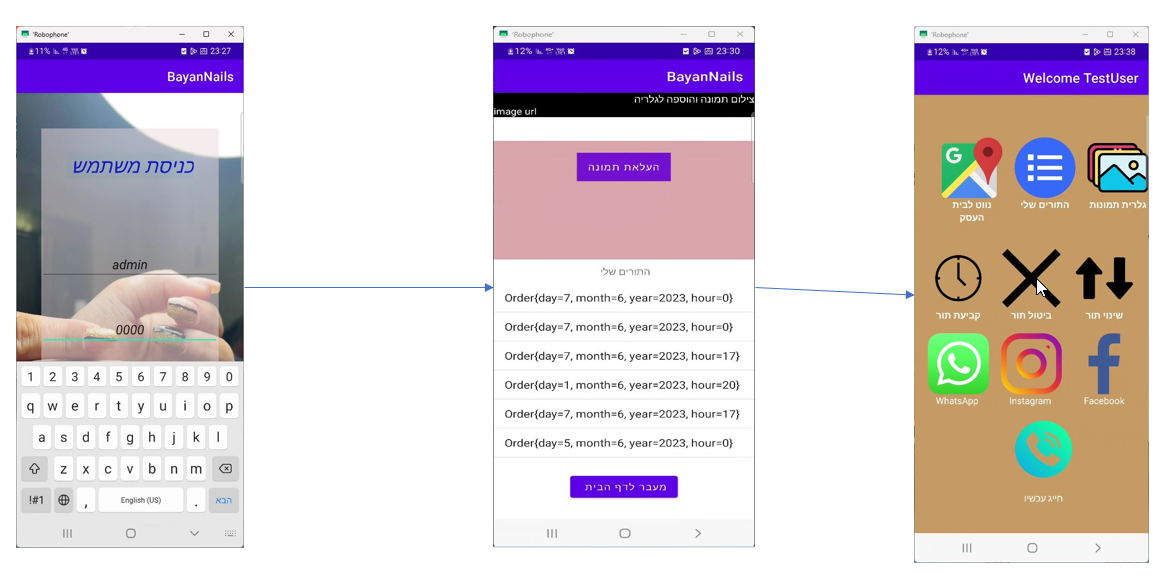
1. ***כניסה של כל מי שמוריד האפליקציה לדף התחברות***



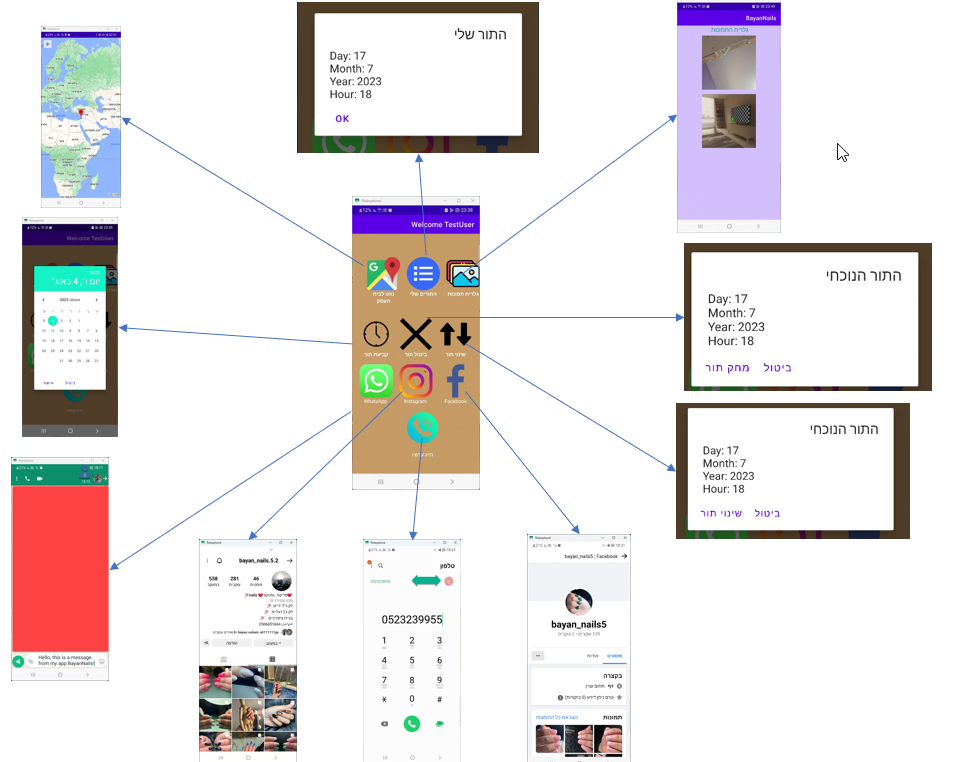
**2.אפשריות התחברות לאפליקציה :**



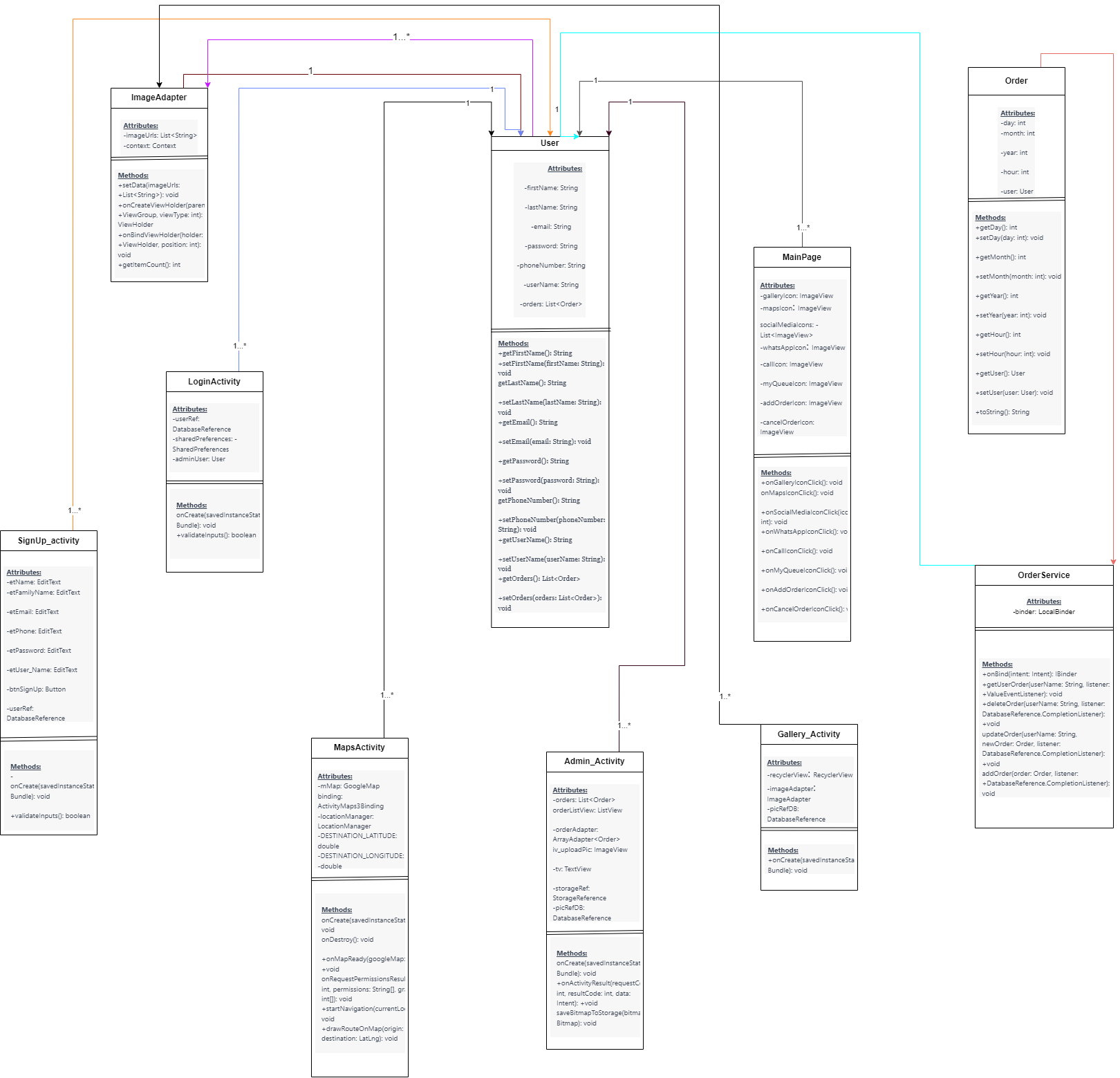
**3.דף כניסה למנהל :**

****

**4.דף כניסה של משתמש:**



**תיאור קשרי המחלקות ע״י תרשים UML:**

****

**תיאור המחלקות והפעולות בעזרת Javadoc:**

| **Class** | **Description** |
| --- | --- |
| User | Represents a user in the application. This class stores user information such as username, email, and password. It provides methods for accessing and manipulating user data. |
| ImageAdapter | The ImageAdapter class acts as an adapter between the list of image URLs and the RecyclerView. It handles the data management, view inflation, data binding, and item count calculations. By using this adapter, you can easily display a list of images in a RecyclerView by providing the image URLs. |
| LoginActivity | LoginActivity handles user authentication and login functionality. It provides methods for user authentication, validation, and login process. It interacts with the backend server or authentication service to verify user credentials. |
| MainPage | MainPage represents the main page of the application. It is responsible for displaying the main interface and handling user interactions. It provides navigation options and access to different features and functionalities of the application. |
| SignUp\_activity | SignUp\_activity manages the user sign-up process. It allows new users to create an account by providing necessary information such as username, email, and password. It validates user input and communicates with the backend server to create a new user account. |
| MapsActivity | MapsActivity displays a map using Google Maps API and provides navigation functionality. It allows users to view their current location, search for places, and get directions to a specific destination. It utilizes GPS or network location services to track user's location and integrates with navigation apps for turn-by-turn directions. |
| AdminActivity | AdminActivity is designed for administrative tasks in the application. It provides features for managing user accounts, handling content moderation, and performing other administrative actions. It requires proper authentication and authorization to access and perform administrative tasks. |
| GalleryActivity | GalleryActivity displays a gallery of images to the user. It allows users to browse through images, view them in a larger size, and perform actions such as downloading or sharing images. It provides a visually appealing and user-friendly interface for browsing and interacting with images. |
| OrderService | OrderService acts as a bridge between the application and Firebase Realtime Database. It provides methods for retrieving, deleting, updating, and adding orders in the database. It encapsulates the necessary logic for these operations, allowing other components of the application to interact with the database in a simplified manner. |
| Order | Order represents an order in the application. It stores information about a user's order, such as the items, quantity, price, and delivery details. It provides methods for accessing and manipulating order data, allowing the application to manage and process user orders efficiently. |

**תיאור ארגון הנתונים הנשמרים ב-FireBase:**

בFireBase הנותנים נשמרים ב-3 מחלקות order,pics,users

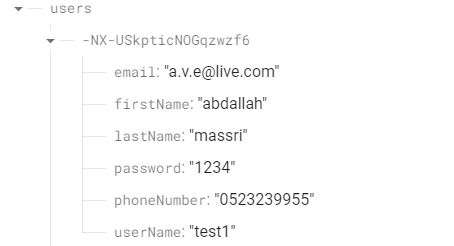


ה-**Users**( משתמשי האפליקציה) נשמרים בצורה שבה לכל אחד יש ID , שונה אבל בוצע בדיקה בתוך האפליקציה עצמה שלא יהיה שני משתמשים עם אותו אימייל או מספר או שם משתמש .

לכל משתמש יש :

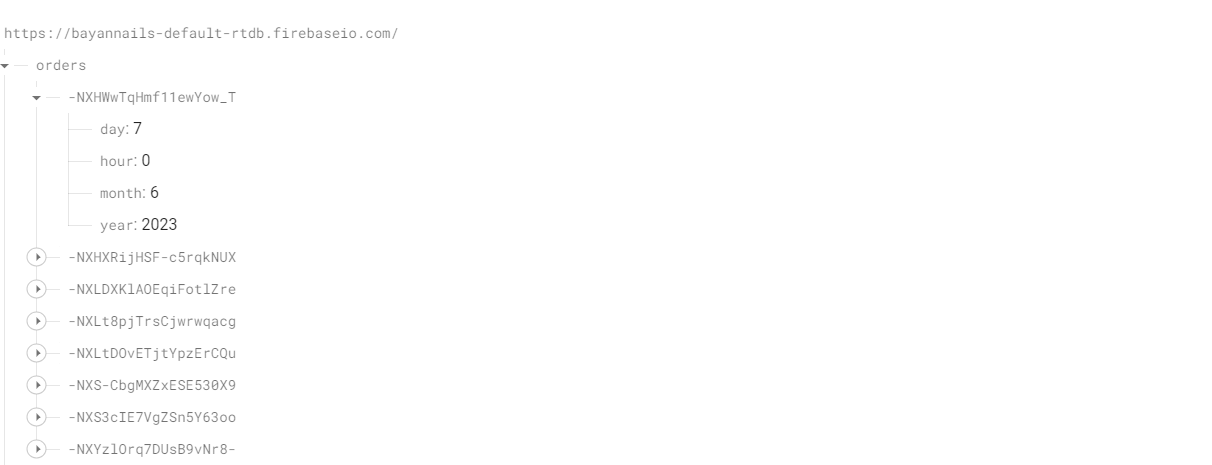
* String : email
* String : firstName
* String : lastName
* String : password
* String: phoneNumber
* String: username

משתמש חדש נשמר באזור זה במסד הנתונים 🡨



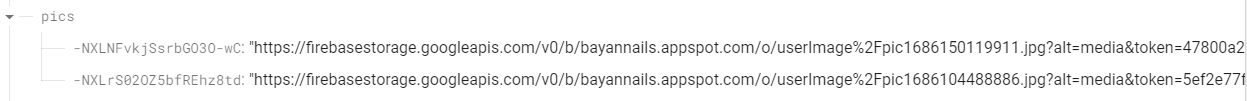
ה-**order** ( התורים) נשמרים בצורה שלכל הזמנה יש מפתח יחודי משלו

* Int : day
* Int : hour
* Int : month
* Int : year



ה-pic (התמונות) נשמרים בצורה שלכל תמונה יש מפתח יחודי משלה :

* String : pic



שימוש באבני יסוד:

**ה-Activities :**

בפרויקט זה קיימים 6 מסכים , ולכולם יש קבצי תבנית (Layout) :

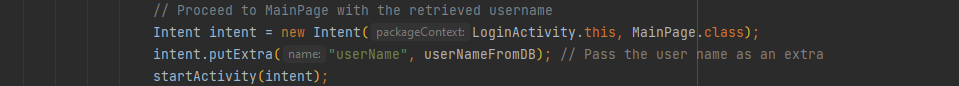
**המסכים הם :**

Admin\_Activity , Gallery\_Activity , LoginActivity , MainPage , SignUp\_activity , MapsActivity

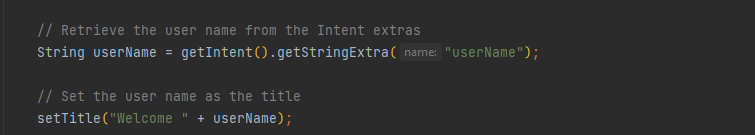
**ה-Intents :**

בפרויקט הזה נזערתי ב Intents כדי לעבור ממסך לשני (לעיתים בתוספת של מידע )

דוגמא ל Intent למעבר לדף הראשי וגם להעביר השדות של שם המשתמש (קטע הקוד נמצא בקובץ LoginActivity )



דוגמא לשליפה של מידע מתוך Intent (במסך שאליו עוברים) . במסך הראשי שולפים המידע ומגדירים כותרת הדף עם ברוך הבא והשם של המשתמש



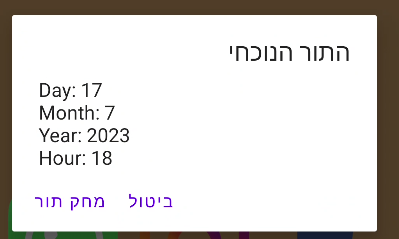
**ה-Dialogs :**

בפרויקט זה קיים 3 דיאלוגים . הדיאלוגים הם מסכים "צפים".

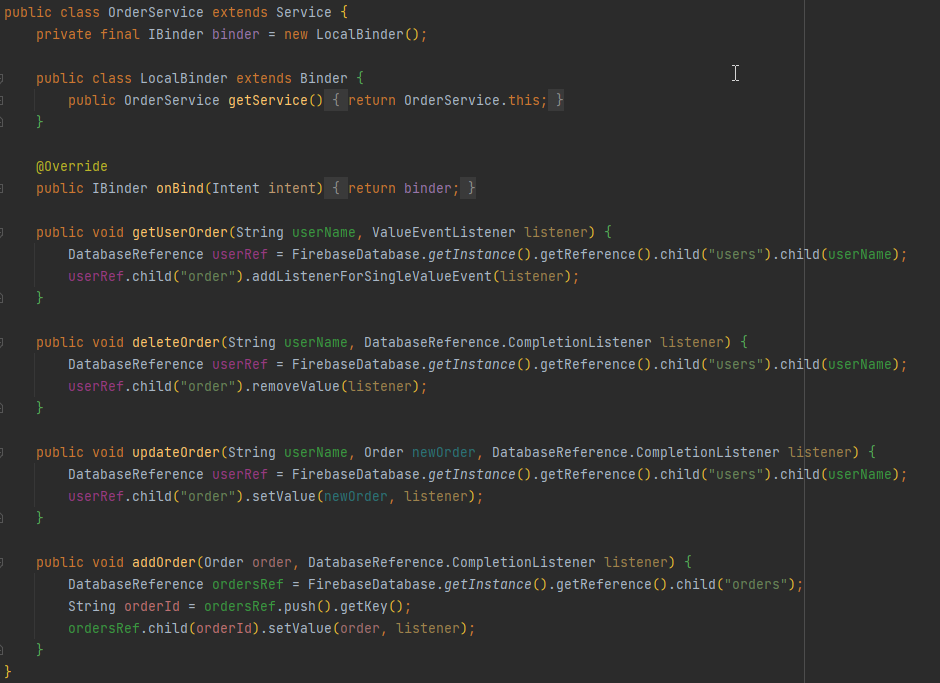
דוגמה לקוד שמופיע דיאלוג ( במקרה הזה – הדיאלוג הוא כאשר המשתמש רוצה לבטל תור הדיאלוג מראה את התור הנוכחי של המשתמש ומבקש אישור . אחרי האישור מציג הודעה ב Toast במסך הראשי )



**דוגמה למראה של דיאלוג:**

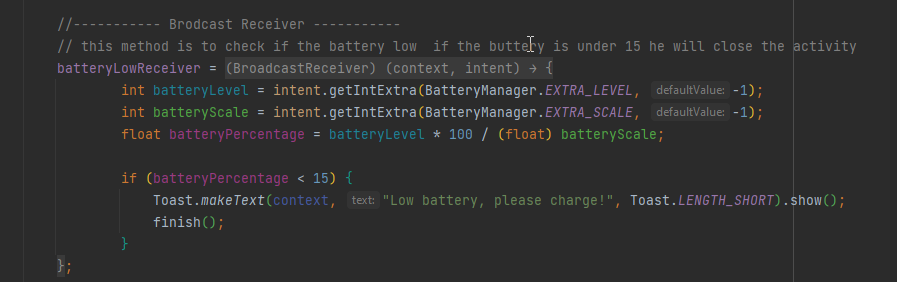


**ה-Service :**

****

**רכיבי ה-BroadcastReciever :**

בפרויקט הזה בחרנו להשתמש ב- BrodcastReciever שבודק אם הסוללה היא קטנה מ 15 אחוז אז הוא לא יפעיל ניווט לבית העסק כי זה מצטרך סוללה ומוריד באחוז הסוללה מהר יותר .

****

**פרק שלישי – סיכום אישי**

**רפלקציה על תהליך הפרויקט:**

החלטנו לפתח את האפליקציה שלנו ב -studio android ממספר סיבות:

- להביא למשתמש בעל מערכת הפעלה android להנגשת השירותים של זימון תורים וניווט לבית העסק ותפעול מהיר וכו'.

- לצבור ניסיון בתוכנת ה-studio android , תוך כדי התמודדות עם עבודה בסביבות פיתוח חדשות .

העבודה על מסד הנתונים היתה מאתגרת. לא היינו רגילים לעבודה עם מסד נתונים שאינו מבוסס על טבלאות, וגילינו שהוא יכול להיות מאוד נוח וקל לשימוש לאחר שלומדים כיצד להתממשק איתו.

במהלך הפרויקט נתקלנו במספר קשיים לדוגמא העלאת תמונה לFireBase ושליפת תמונה , סרביס , תוך כדי עבודה ועזרה מהמנחה איתמר פלדמן הצלחנו להתגבר על הקשיים.

אנו מרגישים כי העשרנו את הידע שלנו בעולם התכנות , הפרויקט לימד אותנו להתמודד עם בעיות מורכבות ולדעת איך לנתח אותן, לימד על כלים מגוונים שנותנים אפשרות להקל על בניית התוכנית.

כמו כן, הפרויקט תרם לפיתוח כישורי הלמידה העצמית שלנו תוך כדי צפייה בסרטונים קשורים , חיפוש פתרונות באינטרנט ובאתר stackoverflow, נעזרנו גם כן בצ'אט-ג'י-פי-טי ולמדנו גישת דיבור איתו.

**פיתוחים להמשך:**

העלאת האפליקציה לחנות האפליקציות – על מנת שהלקוחות החדשים,וותיקים יוכלו להנות ממנה.

בהמשך תהליך הפיתוח נוכל להוסיף אופציה לבצע תשלום מראש , הלקוח יוכל לשלם עבור הטיפול שהינו הולך לעשות בבית העסק לצורך טיפול בבעיות העודף הנפוצות .

רשימת מקורות מידע:

\*לצורך פיתוח הפרויקט, נעזרנו בסרטונים של המנחה – איתמר פלדמן.

לינק לערוץ היוטיוב של איתמר פלדמן:

<https://www.youtube.com/channel/UCsWLxroHLJH4z9wSp0z92JA>

\*נעזרו בכלי החיפוש google כדי למצוא את השגיאות/בעיות וללמוד איך לטפל בהן.  
\*נעזרנו בכלי של אינטלגנצייה מלאכותית – Chat GPT.