МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІСЬКА ПОЛІТЕХНІКА»

Кафедра ІСМ

**Звіт**

До лабораторної роботи №3

З дисципліни:

«Технології проектування в розподілених інформаційних системах»

На тему:

**«Розроблення програми для розподіленого середовища**»

Виконав:

студент групи КН-38

Ткачук Н.В.

Прийняв:

Щербак С.С.

**Львів-2018**

**Мета:** придбання і закріплення навиків в роботі з Firebase, в інтеграції даних, в модульному програмуванні.

**Завдання:** Для заданої прикладної області розробити опис об'єктів цієї області.

Розробити додаток на основі БД Firebase, що реалізують базові операції над цими об'єктами, зокрема:

текстове введення-виведення;

присвоювання;

видалення;

редагування

задання константних значень;

порівняння (не менше 2-х типів);

пошук за шаблоном.

БД повинна зберігати не менше 10 значень конкретних об’єктів. Додаток повинен забезпечувати введення з клавіатури та зчитування з файлу початкових даних в БД та відображення даних з підтримкою сортування по алфавітному і по числовому параметру.

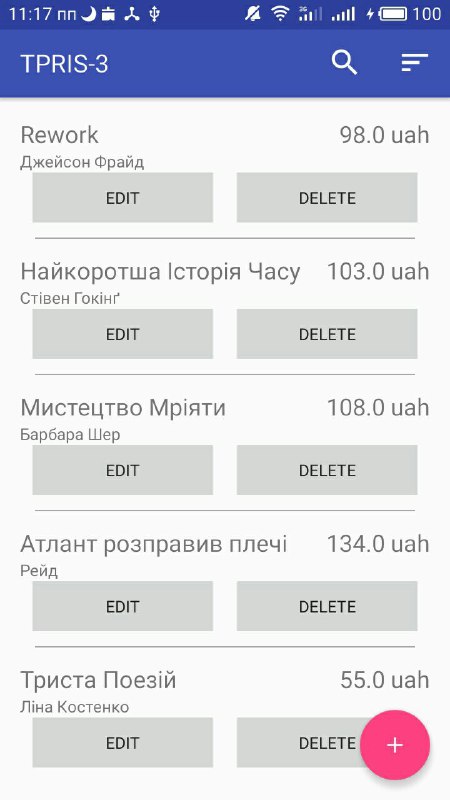
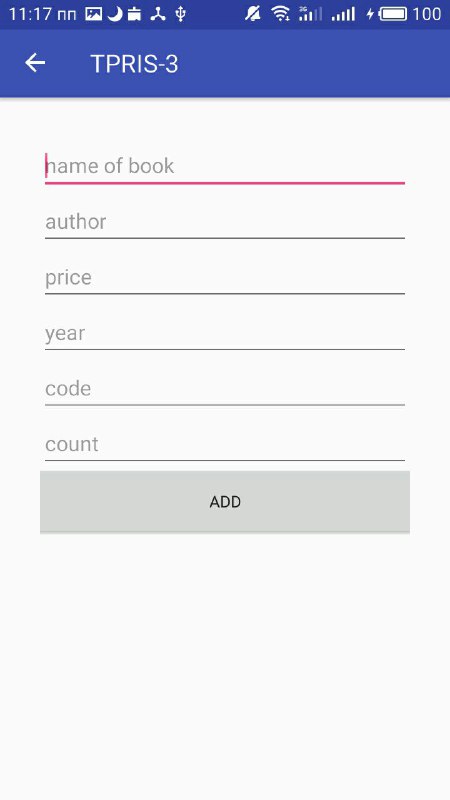
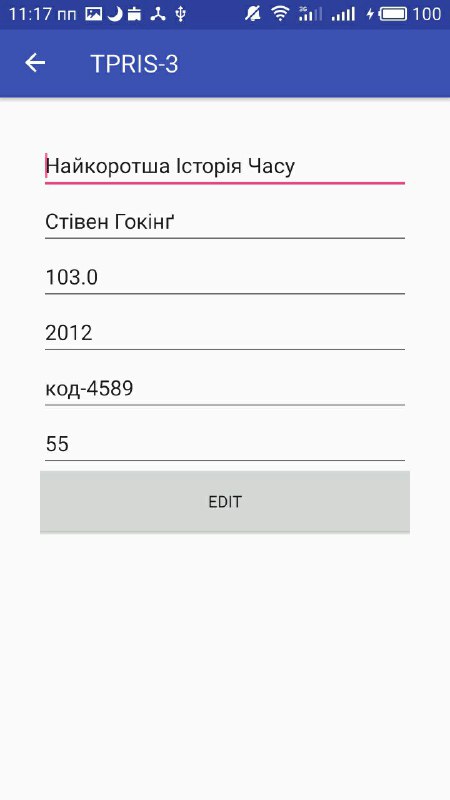


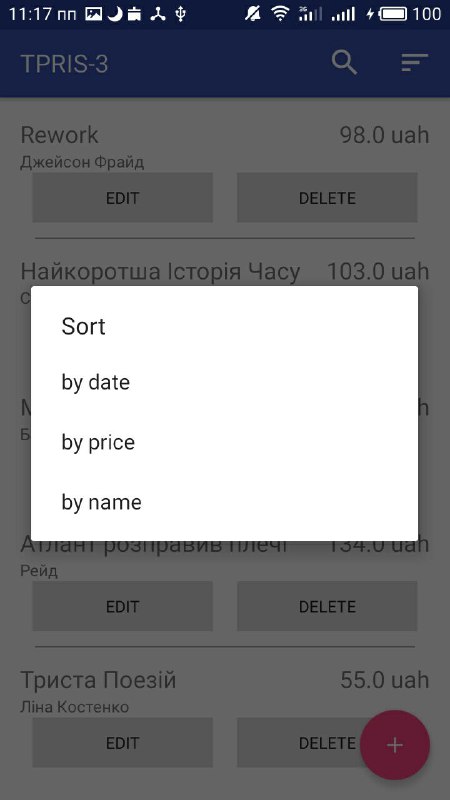
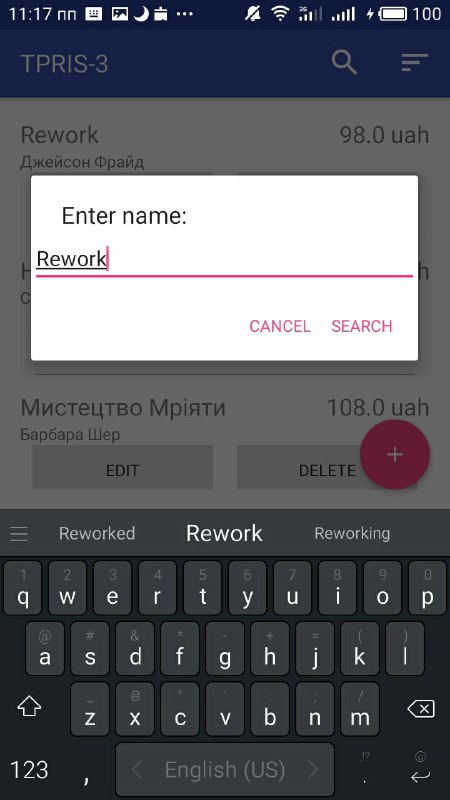
**Хід роботи**

**Код програми:**

**public class** FirebaseManager **implements** IFirebaseManager {  
  
 DatabaseReference **database**;  
  
 **public** FirebaseManager() {  
 **database** = FirebaseDatabase.*getInstance*().getReference();  
 }  
  
 @Override  
 **public void** addBook(Book book, **final** addBookCallBack addBookCallBack) {  
 **database**.child(**"books"**).child(book.getTimestamp()).setValue(book).addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if**(task.isSuccessful()){  
 addBookCallBack.onAdded();  
 }**else** {  
 addBookCallBack.onFailure(task.getException().getMessage().toString());  
 }  
 }  
 });  
 }  
  
 @Override  
 **public void** editBook(Book book, **final** editBookCallBack editBookCallBack) {  
 **database**.child(**"books"**).child(book.getTimestamp()).setValue(book).addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if**(task.isSuccessful()){  
 editBookCallBack.onEdit();  
 }**else** editBookCallBack.onFailure(task.getException().getMessage().toString());  
 }  
 });  
 }  
  
 @Override  
 **public void** deleteBook(String timestamp, **final** deleteBookCallBack deleteBookCallBack) {  
 **database**.child(**"books"**).child(timestamp).removeValue().addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if**(task.isSuccessful()){  
 deleteBookCallBack.onDelete();  
 }**else** deleteBookCallBack.onFailure(task.getException().getMessage().toString());  
 }  
 });  
 }  
  
 @Override  
 **public void** loadList(**final** loadListCallBack loadListCallBack) {  
 **database**.child(**"books"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
  
 ArrayList<Book> bookArrayList = **new** ArrayList<>();  
  
 **for**(DataSnapshot snapshot: dataSnapshot.getChildren()){  
  
 Book book = snapshot.getValue(Book.**class**);  
 bookArrayList.add(book);  
 }  
 loadListCallBack.onLoad(bookArrayList);  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 loadListCallBack.onFailure(databaseError.getMessage().toString());  
 }  
 });  
 }  
  
 @Override  
 **public void** sortListByName(**final** sortListByNameCallBack sortListByNameCallBack) {  
 **database**.child(**"books"**).orderByChild(**"name"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 ArrayList<Book> bookArrayList = **new** ArrayList<>();  
  
 **for**(DataSnapshot snapshot: dataSnapshot.getChildren()){  
  
 Book book = snapshot.getValue(Book.**class**);  
 bookArrayList.add(book);  
 }  
 sortListByNameCallBack.onSorted(bookArrayList);  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 sortListByNameCallBack.onFailure(databaseError.getMessage().toString());  
 }  
 });  
 }  
  
 @Override  
 **public void** sortListByPrice(**final** sortListByPriceCallBack sortListByPriceCallBack) {  
 **database**.child(**"books"**).orderByChild(**"price"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 ArrayList<Book> bookArrayList = **new** ArrayList<>();  
  
 **for**(DataSnapshot snapshot: dataSnapshot.getChildren()){  
  
 Book book = snapshot.getValue(Book.**class**);  
 bookArrayList.add(book);  
 }  
 sortListByPriceCallBack.onSorted(bookArrayList);  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 sortListByPriceCallBack.onFailure(databaseError.getMessage().toString());  
 }  
 });  
 }  
  
 @Override  
 **public void** searchBook(String name, **final** searchBookCallBack searchBookCallBack) {  
 **database**.child(**"books"**).orderByChild(**"name"**).equalTo(name).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
  
 ArrayList<Book> bookArrayList = **new** ArrayList<>();  
  
 **for**(DataSnapshot snapshot: dataSnapshot.getChildren()){  
  
 Book book = snapshot.getValue(Book.**class**);  
 bookArrayList.add(book);  
 }  
 searchBookCallBack.onFind(bookArrayList);  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 searchBookCallBack.onFailure(databaseError.getMessage().toString());  
 }  
 });  
 }  
}

**Результат виконання програми:**

**  **

** **