МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи № 3 з дисципліни

Програмування під Android

Виконав: Перевірив:

ст. гр. ПЗПІ-23-4 ст. викладач кафедри ПІ

Юзков Олександр Сокорчук Ігор Петрович

Харків 2024

ЗАВДАННЯ

Ознайомитися з основними видами діалогових вікон в Android.

Навчитися створювати та використовувати діалогові вікна в Android-застосунках.

Засвоїти методи налаштування діалогових панелей та організації їх взаємодії з користувачем.

Зрозуміти концепцію черги повідомлень (Message Queue) в Android.

Дослідити роботу з обробником повідомлень (Handler) для взаємодії з потоками.

Опанувати створення та використання Handler для асинхронної обробки даних.

Навчитися працювати з компонентом RecyclerView для створення списків з прокручуванням в Android, а також ознайомитися з принципами створення адаптерів (Adapters) і керування даними у списках.

ХІД РОБОТИ

Створюємо новий проєкт та обираємо шаблон Empty Views Activity

RecyclerView – це компонент для ефективного відображення списків або сіток елементів. У цьому проєкті було реалізовано відображення списку з елементами, які завантажуються асинхронно.

Асинхронне завантаження даних

Для завантаження даних було створено метод loadDataInBackground(). Він працює у фоновому потоці, що дозволяє уникнути блокування основного потоку (UI thread).

Код методу:

private void loadDataInBackground() {

new Thread(() -> {

List<String> initialData = Arrays.asList(

"Елемент 1", "Елемент 2", "Елемент 3", "Елемент 4", "Елемент 5",

"Елемент 6", "Елемент 7", "Елемент 8", "Елемент 9", "Елемент 10"

);

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

handler.post(() -> {

data.addAll(initialData);

adapter.notifyDataSetChanged();

});

}).start();

}

Пояснення:

* Фоновий потік: Завантаження даних відбувається у фоні через Thread, що запобігає блокуванню основного потоку.
* Імітація затримки: Використано Thread.sleep(2000) для моделювання затримки при отриманні даних із сервера.
* Оновлення інтерфейсу: Завантажені дані передаються в основний потік через handler.post() і додаються до адаптера.

Додаємо RecyclerView у файл розмітки

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/list"

android:layout\_width="0dp"

android:layout\_height="0dp"

tools:listitem="@layout/list\_item"

tools:itemCount="10"

android:layout\_marginStart="4dp"

android:layout\_marginEnd="4dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent" />

Цей фрагмент коду додає компонент RecyclerView, який буде відображати список елементів у вигляді вертикального лінійного списку завдяки використанню LinearLayoutManager.

Налаштування RecyclerView у MainActivity

Код ініціалізації:

private void setupRecyclerView() {

RecyclerView recyclerView = findViewById(R.id.list);

adapter = new MyAdapter(data);

recyclerView.setAdapter(adapter);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

}

Пояснення:

* Адаптер: Передається список даних до MyAdapter, що відповідає за відображення елементів у RecyclerView.
* Менеджер макету: LinearLayoutManager забезпечує лінійне розташування елементів.

Створюємо адаптер

Код класу адаптера:

static class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

private final List<String> data;

MyAdapter(List<String> data) {

this.data = data;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(android.R.layout.simple\_list\_item\_1, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

holder.textView.setText(data.get(position));

}

@Override

public int getItemCount() {

return data.size();

}

static class ViewHolder extends RecyclerView.ViewHolder {

TextView textView;

ViewHolder(View itemView) {

super(itemView);

textView = itemView.findViewById(android.R.id.text1);

}

}

}

Пояснення:

1. Конструктор: Приймає список даних для відображення.
2. Метод onCreateViewHolder: Використовує стандартний макет для відображення кожного елемента.
3. Метод onBindViewHolder: Встановлює текст для кожного елемента зі списку.
4. Метод getItemCount: Повертає кількість елементів у списку.
5. Клас ViewHolder: Зберігає посилання на елементи інтерфейсу для повторного використання.

Перевірка роботи

* Після запуску методу loadDataInBackground() дані завантажуються у фоні.
* Через 2 секунди список оновлюється, і в RecyclerView відображаються нові елементи.
* Адаптер забезпечує оптимізоване повторне використання елементів через ViewHolder.

ВИСНОВКИ

Під час практичної роботи було створено додаток, який дозволяє ознайомитися з основними типами діалогових панелей та їх атрибутами для налаштування. Також було засвоєно концепцію черги повідомлень і обробки подій за допомогою Handler. Крім того, набуто навичок роботи з компонентом RecyclerView для створення списків з прокручуванням в Android, а також використання адаптерів (Adapters) і керування даними у списках.

Посилання на відеозвіт завантажений на YouTube. URL: https://youtu.be/R4VGiF7v7T0

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Документація AndroidDevelopers Recycler View. URL: <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView> (дата звернення: 31.11.2024)
2. Документація AndroidDevelopers handlers. URL: <https://developer.android.com/reference/android/os/Handler> (дата звернення: 31.11.2024)
3. Документація AndroidDevelopers dialogs. URL: <https://developer.android.com/develop/ui/views/components/dialogs> (дата звернення: 31.11.2024)