Narzędzia Android Studio

➢ ➢ Przeglądanie podstawowych narzędzi Android Studio

➢ ➢ Mastering SDK Manager, AVD Manager i edytor nawigacji

➢ ➢ Nauka generowania dokumentacji Javadoc w Android Studio

W poprzednich częściach dowiedziałeś się, jak Android Studio może generować kod; tu zobaczysz, jak Android Studio może zrefaktoryzować Twój kod, własne komponenty i widżety. Programowanie Androida opiera się na wielu różnych narzędziach, niezależnie od tego, czy zdecydujesz się pójść standardową drogą, czy zamiast tego zdecydujesz się na zastosowanie jednej z niezliczonych innych dostępnych metod. W rzeczywistości, przy tak wielu opcjach na początek, czasami może być trudne ustalenie, która jest dla Ciebie najlepsza. Jakie narzędzia są niezbędne? Jaki program aktywujesz, aby zacząć kodować? Omówiono najpopularniejsze narzędzia programistyczne dla Androida, w tym podstawowe i opcjonalne: Menedżer SDK, Menedżer urządzeń wirtualnych Android (AVD) i Edytor nawigacji. Android SDK oznacza tutaj zestaw narzędzi, który umożliwia programistom tworzenie aplikacji dla systemu operacyjnego Android. Zawiera wymagane biblioteki do tworzenia aplikacji na Androida, debugger, emulator, interfejsy programowania aplikacji (API) i przykładowe projekty z kodem źródłowym, dzięki czemu możesz mieć wszystko, czego potrzebujesz do tworzenia własnych aplikacji. Chociaż istnieje wiele różnych języków programowania i wiele zintegrowanych środowisk programistycznych (IDE), zestaw SDK pozostaje niezmienny. Dlatego rozwój Androida zaczniemy od Android SDK.

MENEDŻER SDK

SDK Manager oferuje szeroki wybór narzędzi wymaganych do tworzenia aplikacji na Androida lub zapewnienia jak najpłynniejszego przebiegu procesu. Niezależnie od tego, czy tworzysz aplikację w języku Java, Kotlin czy C#, potrzebujesz pakietu SDK, aby aktywować ją na urządzeniu z Androidem i uzyskać dostęp do unikalnych funkcji systemu operacyjnego. Będziesz także mógł używać emulatora SDK do testowania utworzonych aplikacji, monitorowania urządzenia i robienia wielu innych rzeczy. Teraz, gdy Android SDK jest dołączony do Android Studio, IDE wpływa na ilość wykonywanej pracy i zapewnia dostęp do wielu narzędzi i zarządzanie nimi. Mimo że konfiguracja z SDK powinna być pierwszą lekcją na temat programowania Androida, którą przechodzisz, jest w tym jeszcze trochę więcej, a pełne wykorzystanie wszystkich narzędzi programistycznych i wiedza o tym, jak dokładnie działa SDK, z pewnością zaowocuje w lepszych aplikacjach. Android SDK można zainstalować automatycznie przy użyciu najnowszej wersji Gradle lub ręcznie pobierając Android SDK na kilka różnych sposobów, jeśli takie jest Twoje preferencje. Rzućmy okiem na krótki przegląd wszystkich różnych podejść.

Instalowanie Android SDK (zautomatyzowany sposób)

Najnowsza wersja Gradle 7.0 obsługuje automatyczne pobieranie zależności. Upewnij się, że uaktualniłeś do najnowszej wersji Gradle, w przeciwnym razie zobaczysz niektóre wtyczki Gradle, które zazwyczaj zarządzają zależnościami, które mają być przestarzałe w starszych wersjach.

Instalacja dla Ubuntu Linux

Jeśli używasz Ubuntu 15.10 lub starszego, upewnij się, że zainstalowałeś następujący standardowy pakiet. W przeciwnym razie podczas próby uruchomienia programu apt, który jest częścią zestawu narzędzi Android SDK, może pojawić się komunikat „Brak takiego pliku lub katalogu”:

sudo apt-get install libc6-dev-i386 lib32z1 OpenJDK-8-JDK

Instalacja przez Homebrew

Jeśli masz uruchomiony MacOS/OS X, możesz użyć Homebrew do zainstalowania Android SDK. Aby zainstalować Homebrew - menedżera pakietów dla MacOS/OS X, musisz aktywować następujące polecenia:

brew tap homebrew/cask

brew install –cask android-SDK

Spowoduje to zainstalowanie narzędzi Android SDK w katalogu /usr/local/Cellar/android-SDK/< numer wersji >.

Instalowanie Android SDK (sposób ręczny)

Jeśli chcesz ręcznie zainstalować Android SDK, musisz pobrać Android SDK bez pakietu Android Studio. W tym celu przejdź do Android SDK i przejdź do sekcji Tylko narzędzia SDK. Tutaj skopiuj adres URL do pobrania, który jest odpowiedni dla Twojej maszyny do kompilacji. Następnie użyj wget z poprawnym adresem URL SDK:

$ wget https://dl.google.com/android/repository/tools_r25.2.3-macosx.zip

Następnie musisz rozpakować i umieścić zawartość w swoim katalogu domowym. Nazwy katalogów mogą być dowolne, ale pamiętaj, aby zapisać pliki w łatwym do znalezienia miejscu. Aby uruchomić narzędzie SDK manager, wstaw następujące funkcje:

$ tools/bin/sdkmanager –update

$ tools/bin/sdkmanager “platforms;android-25”

“build-tools;25.0.2” “extras;google;m2repository” “extras;android;m2repository”

$ tools/bin/sdkmanager –licenses

Następnie należy ustawić zmienną PATH środowiska kompilacji i inne zmienne, które zostaną zastosowane do zlokalizowania Androida. Możesz również rozważyć edycję pliku .bash_profile. A jeśli nie używasz basha, powinieneś edytować odpowiedni plik konfiguracyjny dla swojego środowiska. Ilustrować:

export ANDROID_SDK_ROOT=/Users/android/android-SDK-macosx

export PATH=$PATH:$ANDROID_SDK_ROOT/tools

Na koniec, zanim klikniesz Zapisz i zamkniesz, ponownie załaduj .bash_profile po raz ostatni:

$ source ~/.bash_profile

Instalacja przez GUI

Najpierw uzyskaj dostęp do monitu, wpisz android i naciśnij Enter, aby uruchomić Menedżera Android SDK w oknie. Jeśli to nie zadziała, oznacza to, że zmienna PATH nie została poprawnie skonfigurowana z lokalizacją Android SDK. Oznacza to również, że będziesz musiał zainstalować te same pakiety Android SDK na swoim komputerze kompilacyjnym, co w przypadku uruchomienia Gradle lokalnie. Oto nazwy pakietów SDK, od których z pewnością musisz zacząć:

Tools > Android SDK Tools

Tools > Android SDK Platform-tools

Tools > Android SDK Build-tools

Ponadto będziesz chciał również pobrać kilka dodatków:

Repozytorium wsparcia dla Androida

Biblioteka wsparcia dla Androida

Następnie wybierz narzędzia kompilacji Android SDK dla wersji Androida wymienionej w pliku build.Gradle jako cel android: buildToolsVersion. Jeśli twój build.gradle mówi

android {

buildToolsVersion "21"

...

}

następnie pobierz tę wersję interfejsu API w Menedżerze Android SDK.

Instalacja za pomocą wiersza poleceń

Możliwe jest również pobranie pakietów SDK za pomocą następującego wiersza poleceń z parametrem -no-ui:

android update SDK –no-ui –all

Jeśli chcesz być selektywny przy instalacji, możesz użyć listy android, aby wyświetlić wszystkie pakiety i zastosować opcję -filter dla selektywnych instalacji w taki sposób:

sudo android update SDK –no-UI –filter platform-tools, tools

A kiedy zdecydujesz się na selektywne wybieranie pakietów do zainstalowania, musisz również uwzględnić następujące dodatkowe repozytorium Androida Maven. W przeciwnym razie możesz nie być w stanie korzystać z najnowszej biblioteki projektów wsparcia:

android update SDK –no-ui –all –filter extra-android-m2repository

Anatomia SDK Androida

Android SDK można podzielić na kilka głównych komponentów. Obejmują one:

• Platforma-narzędzia

• Narzędzia do budowania

• Narzędzia SDK

• Most debugowania Androida (ADB)

• Emulator Androida

Z pewnością najważniejsze części tego pakietu znajdują się w narzędziach SDK. Będziesz potrzebować tych narzędzi niezależnie od docelowej wersji Androida. To właśnie one stanowią zestaw Android Package Kit (APK) - zmieniając program Java w aplikację na Androida, którą można obsługiwać nawet na telefonie. Obejmują one szereg narzędzi do kompilacji, narzędzia do debugowania i narzędzia do

obrazów. Świetnym przykładem takiego narzędzia jest usługa debugowania Dalvik Debug Monitor Server (DDMS), która pozwala nam używać Android Device Monitor do sprawdzania stanu urządzenia z Androidem. Narzędzia do kompilacji były wcześniej klasyfikowane pod tym samym nagłówkiem, co narzędzia platformy, ale od tego czasu zostały oddzielone w późniejszych wersjach, dzięki czemu można je aktualizować osobno. Jak sama nazwa wskazuje, są one również niezwykle ważne przy tworzeniu aplikacji na Androida. Przykładem takich narzędzi może być narzędzie Zipalign, które optymalizuje aplikację, aby zażądać minimalnej ilości pamięci podczas uruchamiania przed wygenerowaniem końcowego pakietu APK, oraz Apksigner, który podpisuje pakiet APK w celu dalszej weryfikacji. Narzędzia platformy mogą być bardziej szczegółowo dostosowane do wersji Androida, na którą chcesz kierować reklamy. Zazwyczaj uważa się, że najlepiej jest zainstalować najnowsze narzędzia platformy, które zwykle są instalowane domyślnie. Jednak po pierwszej instalacji musisz stale aktualizować narzędzia platformy. Narzędzia są kompatybilne wstecz, co oznacza, że są również używane do obsługi starszych wersji Androida. Podsumowując, wiele z wyżej wymienionych narzędzi ma kluczowe znaczenie dla testowania, debugowania i pakowania SDK. Zapewniają one rodzaj połączenia między Android Studio a fizycznym urządzeniem lub emulatorem, dzięki czemu Twoja aplikacja może być specjalnie pakowana, a następnie testowana w trakcie pracy. Bezpiecznie jest pozostawić SDK w spokoju przez większość programistyczną: Android Studio zaleci niezbędne aktualizacje i wywoła wymagane elementy, gdy klikniesz Uruchom lub Zbuduj APK. To powiedziawszy, kilka narzędzi jest również bezpośrednio dostępnych, które zostaną zastosowane do operacji takich jak aktualizacja SDK lub bezpośrednie monitorowanie i modyfikowanie urządzenia z Androidem. Podczas gdy Android Studio zwykle poinformuje Cię, kiedy musisz coś zaktualizować, możesz również zarządzać aktualizacjami SDK ręcznie za pomocą menedżera. Możesz to znaleźć w Android Studio, jeśli przejdziesz do Narzędzia — Android — Menedżer SDK. Aktywacja Zarządzaj pozwoli Ci na takie rzeczy, jak wybór rozmiaru urządzenia i innych specyfikacji, a zostaniesz poproszony o pobranie wymaganego obrazu systemu ×86, jeśli nie jest jeszcze zainstalowany.

Korzystanie z ADB

Korzystanie z ADB jest nieco inne. Jest to program, który pozwala komunikować się z dowolnym urządzeniem z systemem Android. W swoim założeniu opiera się na narzędziach platformy w celu odczytania wersji Androida używanej na tym urządzeniu, dlatego jest zawarte w pakiecie narzędzi platformy. Możesz użyć ADB, aby uzyskać dostęp do narzędzi powłoki, takich jak Logcat, sprawdzić identyfikator urządzenia, a nawet zainstalować aplikacje. Aby uzyskać dostęp do ADB, musisz znaleźć folder instalacyjny Android SDK i przejść do katalogu platform-tools. W systemie Windows przytrzymaj klawisz Shift i kliknij prawym przyciskiem myszy w dowolnym miejscu folderu, aby otworzyć wiersz poleceń. Na Macu po prostu otwórz Terminal z Launchpada, który znajduje się w folderze Inne). ADB to elastyczne narzędzie wiersza poleceń, które umożliwia interakcję z urządzeniem. Polecenie ADB może zarządzać różnymi akcjami urządzenia, takimi jak instalowanie i debugowanie aplikacji, i zapewnia dostęp do powłoki systemu Unix, którą można administrować w celu uruchamiania różnych poleceń na urządzeniu. Na przykład, jeśli wpiszesz „urządzenia adb”, otrzymasz listę podłączonych urządzeń z Androidem wraz z ich identyfikatorami urządzeń. A jeśli napiszesz skrypt „adb install [opcje] nazwa-pakietu”, będziesz upoważniony do zdalnej instalacji APK. Jest również uważany za program klient-serwer, który zawiera następujące trzy komponenty:

1. Klient, który przekazuje polecenia. Klient działa na komputerze deweloperskim i można wywołać klienta z terminala wiersza polecenia, wydając jedno z poleceń ADB.

2. Demon (adbd), który uruchamia polecenia na urządzeniu. Demon zazwyczaj działa w tle na każdym urządzeniu.

3. Serwer, który nawiązuje komunikację między klientem a demonem. Serwer działa jako proces w tle na komputerze deweloperskim.

Jak już wspomniano, ADB jest zawarte w pakiecie Android SDK Platform-Tools. Możesz pobrać ten pakiet za pomocą Menedżera SDK, który instaluje go pod adresem android_sdk/platform-tools/. Przy pierwszym uruchomieniu klienta ADB klient sprawdza, czy serwer ADB już działa. Tylko jeśli nie ma, uruchamia proces serwera. Po uruchomieniu serwer łączy się z lokalnym portem TCP 5037 i rejestruje wszystkie polecenia wysyłane od klientów adb – wszyscy klienci adb używają portu 5037 do komunikacji z serwerem ADB. Serwer następnie nawiązuje połączenia ze wszystkimi uruchomionymi urządzeniami. Lokalizuje emulatory, skanując nieparzyste porty z zakresu 5555–5585, z zakresu używanego przez pierwszych 16 emulatorów. Tam, gdzie serwer znajdzie demona ADB (adbd), ustanawia połączenie z tym portem. Każdy emulator używa pary portów sekwencyjnych — parzystego portu dla połączeń konsoli i nieparzystego portu dla połączeń ADB. Na przykład:

• Emulator 1, konsola: 5555

• Emulator 1, adb: 5556

• Emulator 2, konsola: 5557

• Emulator 2, adb: 5558

Jak pokazano, emulator podłączony do adb na porcie 5554 jest taki sam, jak emulator, którego konsola nasłuchuje na porcie 5556. Gdy serwer zakończy nawiązywanie połączeń ze wszystkimi urządzeniami, możesz użyć poleceń ADB, aby uzyskać dostęp do tych urządzeń. Ponieważ serwer zarządza połączeniami z urządzeniami, a także poleceniami z wielu klientów adb, możesz również kontrolować dowolne urządzenie z dowolnego klienta lub skryptu. Aby korzystać z adb z urządzeniem podłączonym przez USB, należy włączyć debugowanie USB w ustawieniach systemowych urządzenia, w opcjach programisty. W systemie Android 4.2 i nowszych ekran opcji programisty jest domyślnie ukryty. Aby było to widoczne, przejdź do Ustawienia > Informacje o telefonie i dotknij Numer kompilacji siedem razy. Po powrocie do poprzedniego ekranu na dole znajdziesz opcje programisty. Możesz teraz podłączyć swoje urządzenie przez USB. Sprawdź, czy Twoje urządzenie jest połączone, implementując urządzenia adb z katalogu android_sdk/platform-tools/. Po połączeniu będziesz mógł zobaczyć nazwę urządzenia wymienioną jako „urządzenie”.

Emulator Androida

Emulator systemu Android symuluje urządzenia z systemem Android na komputerze, dzięki czemu można testować aplikację na różnych urządzeniach i na różnych poziomach interfejsu API systemu Android bez konieczności interakcji z każdym urządzeniem fizycznym. Emulator może imitować prawie wszystkie możliwości prawdziwego urządzenia z Androidem. W ten sposób możesz odtwarzać przychodzące połączenia telefoniczne i wiadomości tekstowe, określać lokalizację urządzenia, symulować różne prędkości sieci, symulować czujniki obrotu i sprzętu, a także uzyskiwać dostęp do Google Play Store. Testowanie aplikacji na emulatorze jest zdecydowanie szybsze i łatwiejsze niż robienie tego na urządzeniu fizycznym. Na przykład możesz szybciej przesyłać dane do emulatora niż do urządzenia podłączonego przez USB. Wygodne jest również to, że emulator zawiera predefiniowane konfiguracje dla różnych telefonów z systemem Android, tabletów, urządzeń Wear OS i Android TV. Każde wystąpienie emulatora Androida używa AVD do określenia wersji Androida symulowanego urządzenia i funkcji sprzętowych. Aby produktywnie przetestować swoją aplikację, utwórz plik AVD, który odtwarza każde urządzenie, na którym aplikacja ma działać. Aby tworzyć i zarządzać AVD, możesz użyć AVD Manager. Każdy AVD działa jako niezależne urządzenie, z własną prywatną pamięcią danych

użytkownika i kartą Secure Digital. Domyślnie emulator przechowuje dane użytkownika, dane karty SD i pamięć podręczną w katalogu specyficznym dla tego AVD. Dzięki temu po uruchomieniu emulator ładuje dane użytkownika i dane karty SD z katalogu AVD.

Uruchamianie aplikacji na emulatorze Androida

Możesz uruchomić aplikację z projektu Android Studio lub możesz uruchomić aplikację zainstalowaną na emulatorze Androida, tak jak uruchamiasz dowolną aplikację na urządzeniu. Aby uruchomić emulator Androida i uruchomić aplikację w swoim projekcie: W Android Studio utwórz AVD, którego emulator może użyć do zainstalowania i uruchomienia Twojej aplikacji. Na pasku narzędzi wybierz AVD, na którym chcesz uruchomić swoją aplikację, z menu rozwijanego urządzenia docelowego i kliknij Uruchom. Jeśli w górnej części okna pojawi się komunikat o błędzie lub ostrzeżenie, kliknij łącze, aby rozwiązać problem lub uzyskać więcej informacji. Niektóre błędy, takie jak błędy Hardware Accelerated Execution Manager (Intel HAXM), należy naprawić przed kontynuowaniem. W przypadku systemu MacOS, jeśli zobaczysz następujące ostrzeżenie: błąd nie znaleziono serwerów DNS podczas uruchamiania emulatora, sprawdź, czy masz plik /etc/resolv.conf. Jeśli nie masz tego pliku, wprowadź następujące polecenie w oknie terminala:

ln -s /private/var/run/resolv.conf /etc/resolv.conf

Obecnie nie można używać rozszerzonych kontrolek emulatora, gdy jest on uruchomiony w oknie narzędzia. Jeśli przepływ pracy deweloperskiej opiera się w dużej mierze na rozszerzonych kontrolkach, nadal używaj emulatora systemu Android jako aplikacji autonomicznej. Ponadto niektóre urządzenia wirtualne — takie jak Android TV i urządzenia składane — nie mogą być uruchamiane w Android Studio, ponieważ mają określone wymagania dotyczące interfejsu użytkownika lub ważne funkcje w rozszerzonych kontrolkach. Poruszanie się po ekranie emulatora Aby poruszać się po ekranie emulatora, możesz użyć wskaźnika myszy komputera do naśladowania palca na ekranie dotykowym. Ponadto możesz wpisywać znaki i wprowadzać następujące skróty emulatora przedstawione w tabeli:

Funkcja: Opis

Przesuń palcem po ekranie : wskaż ekran, naciśnij i przytrzymaj główny przycisk myszy, przesuń palcem po ekranie, a następnie zwolnij.

Przeciągnij element : wskaż element na ekranie, naciśnij i przytrzymaj główny przycisk myszy, przesuń element, a następnie zwolnij.

Dotknij (dotknij): wskaż ekran, naciśnij główny przycisk myszy, a następnie zwolnij. Na przykład możesz kliknąć pole tekstowe, aby rozpocząć w nim pisanie, wybrać aplikację lub nacisnąć przycisk.

Dwukrotne dotknięcie : wskaż ekran, dwukrotnie naciśnij szybko główny przycisk myszy, a następnie zwolnij.

Dotknij i przytrzymaj : wskaż element na ekranie, naciśnij główny przycisk myszy, przytrzymaj, a następnie zwolnij. Na przykład możesz otworzyć opcje dla przedmiotu.

Wpisz : możesz pisać w emulatorze, używając klawiatury komputera lub klawiatury, która pojawia się na ekranie emulatora. Na przykład możesz wpisać w polu tekstowym po jego zaznaczeniu.

Uszczypnij i rozciągnij : naciśnięcie klawisza Control (Command na Macu) powoduje wyświetlenie interfejsu wielodotykowego gestów uszczypnięcia. Mysz działa jak pierwszy palec, a w poprzek punktu zakotwiczenia znajduje się drugi palec. Przeciągnij kursor, aby przesunąć pierwszy punkt. Kliknięcie

lewym przyciskiem myszy działa jak dotknięcie obu punktów, a puszczenie działa jak podniesienie obu punktów.

Przeciągnięcie w pionie : Otwórz pionowe menu na ekranie i użyj kółka przewijania (kółka myszy), aby przewijać pozycje menu, aż zobaczysz ten, który chcesz. Kliknij element menu, aby go wybrać.

Migawki

Migawka to przechowywany obraz AVD, który zapisuje cały układ urządzenia w momencie zapisania - w tym ustawienia systemu operacyjnego, stan aplikacji i dane użytkownika. Możesz przejść do zapisanego stanu systemu, ładując migawkę w dowolnym momencie, oszczędzając czas oczekiwania na ponowne uruchomienie systemu operacyjnego i aplikacji na urządzeniu wirtualnym, a także oszczędzając zasoby związane z przywróceniem aplikacji do stanu w którym chcesz kontynuować testowanie. Uruchamianie urządzenia wirtualnego należy postrzegać jako wybudzanie urządzenia fizycznego z trybu uśpienia, a nie wyprowadzanie go ze stanu wyłączenia. Dla każdego AVD można mieć jedną migawkę szybkiego rozruchu i dowolną liczbę migawek ogólnych. Najłatwiejszym sposobem wykorzystania migawek jest użycie migawek szybkiego rozruchu: domyślnie każdy AVD jest ustawiony tak, aby automatycznie robił migawkę szybkiego rozruchu przy wyjściu i ładował się z migawki szybkiego rozruchu przy uruchomieniu. Przy pierwszym uruchomieniu AVD powinien aktywować zimny rozruch, podobnie jak włączanie urządzenia. Jeśli włączony jest Szybki rozruch, wszystkie kolejne uruchomienia są ładowane z określonej migawki, a system jest przywracany do stanu zapisanego w tym zrzucie. Migawki są prawidłowe dla obrazu systemu, konfiguracji AVD i funkcji emulatora, z którymi są zapisywane. W przypadku wprowadzenia zmian w którejkolwiek z tych domen wszystkie migawki zmodyfikowanego AVD staną się nieważne. Podobnie każda aktualizacja emulatora Androida, obrazu systemu lub ustawień AVD resetuje zapisany stan AVD, więc przy następnym uruchomieniu AVD wykona zimny rozruch.

Używanie SDK niezależnie

Jeśli jesteś zdecydowany samodzielnie eksplorować SDK, możesz znaleźć cały podkatalog w folderze SDK o nazwie „Dokumenty”, co da ci dostęp do przydatnych informacji. W większości przypadków zaleca się jednak odwiedzenie https://developer.android.com/. Wcześniej pakiet Android SDK zawierał zestaw przydatnych przykładowych projektów. Jednak obecnie tak nie jest, ale nadal można je znaleźć, otwierając Android Studio i przechodząc do Plik - Nowy - Importuj próbkę. Chociaż Android SDK i Android Studio są ze sobą ściśle powiązane, nie zawsze będziesz chciał ich używać razem. Być może będziesz musiał użyć innego IDE, na przykład, jeśli chcesz usprawnić proces tworzenia gry 3D (w takim przypadku możesz również użyć rozwiązań Unity lub Unreal) lub jeśli jesteś zainteresowany cross-programowanie mobilne platformy (w takim przypadku możesz zastosować platformę Xamarin). W każdym razie będziesz musiał pokazać wybrane IDE, w którym znajduje się SDK, zwykle poprzez skryptowanie gdzieś ścieżki. Możesz także znaleźć lokalizację Android SDK w Android Studio, na wypadek, gdybyś musiał go przenieść, lub po prostu użyj go do własnego odniesienia. Wystarczy przejść do Plik- Struktura projektu. Tutaj będzie można również zobaczyć lokalizację JDK oraz Android Native Development Kit (NDK). Możesz wybrać lokalizację SDK podczas instalacji. Jeśli jednak pozostawiłeś tę opcję jako domyślną, może się zdarzyć, że znajduje się ona w katalogu AppData\Local. Należy pamiętać, że ten folder jest domyślnie ukryty w systemie Windows, więc znalezienie go może zająć trochę czasu. Wspomniany wcześniej NDK umożliwia budowanie aplikacji w językach natywnych, takich jak C i C++. Dzięki nim możesz uzyskać dostęp do niektórych bibliotek i zwiększyć wydajność urządzenia – dzięki czemu można je zastosować między innymi do tworzenia gier. Jak wspomniano, jeśli interesuje Cię tylko pakiet SDK, możesz go pobrać samodzielnie, odwiedzając stronę pobierania, a następnie wybierając menedżera SDK. Umożliwi to aktualizację SDK za pomocą wiersza poleceń.

Istnieją również sposoby uzyskania dostępu do Menedżera AVD bez Android Studio. Jednak w przypadku zdecydowanej większości użytkowników o wiele łatwiej jest po prostu zainstalować pełny pakiet i cieszyć się interfejsem graficznym i innymi funkcjonalnościami - nawet jeśli zamierzają używać innego IDE do rozwoju.

AVD MANAGER

AVD to konfiguracja, która definiuje właściwości telefonu, tabletu z systemem Android, urządzenia Wear OS, Android TV lub Automotive OS, które zamierzasz symulować w emulatorze systemu Android. Dlatego AVD Manager jest interfejsem, który można uruchomić z Android Studio, który pomaga tworzyć i obsługiwać AVD. Aby otworzyć Menedżera AVD, przejdź do funkcji Wybierz narzędzia i kliknij opcję Menedżer AVD na pasku narzędzi. AVD zawiera profil sprzętu, obraz systemu, obszar pamięci, skórkę i inne atrybuty. Ogólnie zaleca się utworzenie AVD dla każdego obrazu systemu, który aplikacja może potencjalnie obsługiwać na podstawie ustawienia <uses-sdk > w manifeście. Profil sprzętowy określa charakterystykę urządzenia w stanie fabrycznym. Menedżer AVD jest dostarczany z fabrycznie załadowanymi profilami sprzętowymi, takimi jak urządzenia Pixel, ale w razie potrzeby można również dostosować profile sprzętowe. Jednak nie wszystkie profile sprzętowe zawierają Sklep Play. Oznacza to, że tylko niektóre profile są w pełni zgodne z pakietem testów zgodności (CTS) i mogą używać obrazów systemu, które zawierają aplikację Sklep Play.

Obrazy systemowe

Obraz systemu oznaczony przy użyciu interfejsów API Google zwykle obejmuje dostęp do usług Google Play. Obraz systemu oznaczony logo Google Play w kolumnie Sklep Play zawiera aplikację Sklep Google Play i dostęp do usług Google Play, w tym kartę Google Play w oknie dialogowym Rozszerzone elementy sterujące, która zapewnia wygodny przycisk do aktualizacji usług Google Play na urządzeniu . Aby zapewnić bezpieczeństwo aplikacji i płynne działanie na urządzeniach fizycznych, obrazy systemowe z dołączonym Sklepem Google Play są podpisane kluczem wydania, co oznacza, że nie będziesz w stanie uzyskać podwyższonych uprawnień (root) za pomocą tych obrazów. Załóżmy, że potrzebujesz podwyższonych uprawnień (root), aby pomóc w rozwiązywaniu problemów z aplikacją. W takim przypadku możesz użyć obrazów systemu Android Open Source Project (AOSP), które nie zawierają aplikacji ani usług Google.

Powierzchnia magazynowa

AVD ma również dodatkowy obszar przechowywania na twoim komputerze deweloperskim. Służy głównie do przechowywania danych użytkownika urządzenia, takich jak zainstalowane aplikacje i ustawienia, a także emulowana karta SD. W razie potrzeby możesz użyć Menedżera AVD do wyczyszczenia danych użytkownika, aby urządzenie zawierało takie same dane, jak gdyby było nowe.

Skóra

Mówiąc prościej, za wygląd urządzenia odpowiada skórka emulatora. Menedżer AVD ma kilka predefiniowanych skórek, z których możesz wybierać. Dodatkowo możesz również zdefiniować własne lub użyć skórek określonych przez osoby trzecie.

Tworzenie AVD

Aby utworzyć nowy AVD, otwórz Menedżera AVD, klikając Narzędzia > Menedżer AVD. Następnie wybierz opcję Utwórz urządzenie wirtualne w dolnej części okna dialogowego Menedżer AVD. Na stronie Wybierz sprzęt wybierz profil sprzętu, a następnie kliknij Dalej. Jeśli nie widzisz odpowiedniego profilu sprzętu, możesz utworzyć lub zaimportować profil sprzętu. Po wyświetleniu strony Obraz

systemu wybierz obraz systemu dla określonego poziomu interfejsu API, a następnie kliknij przycisk Dalej. Jeśli widzisz Pobierz obok obrazu systemu, musisz go kliknąć, aby pobrać obraz systemu. Aby go pobrać, musisz mieć połączenie z internetem. Poziom interfejsu API urządzenia docelowego jest ważny, ponieważ aplikacja nie będzie mogła działać na obrazie systemu z poziomem interfejsu API niższym niż wymagany przez aplikację, zgodnie z atrybutem minSdkVersion pliku manifestu aplikacji. Jeśli aplikacja deklaruje element <uses-library> w pliku manifestu, aplikacja wymaga obrazu systemu, w którym znajduje się ta biblioteka zewnętrzna. Jeśli musisz uruchomić aplikację na emulatorze, utwórz plik AVD zawierający wymaganą bibliotekę. W tym celu może być konieczne użycie dodatkowego komponentu dla platformy AVD; na przykład dodatek Google APIs, który zawiera bibliotekę Google Maps. Na następnej stronie Sprawdź konfigurację, która się pojawi, zmień właściwości AVD zgodnie z potrzebami, a następnie kliknij przycisk Zakończ. Kliknij Pokaż ustawienia zaawansowane, aby wyświetlić więcej ustawień, na przykład karnację. Nowy AVD powinien zostać wyświetlony na stronie Your Virtual Devices lub w oknie dialogowym Select Deployment Target. Ponadto możliwe jest utworzenie AVD zaczynając od kopii: Na stronie Your Virtual Devices programu AVD Manager kliknij prawym przyciskiem AVD i wybierz Duplikuj. Następnie na wyświetlonej stronie Sprawdź konfigurację kliknij Zmień lub Poprzedni, jeśli chcesz wprowadzić zmiany na stronach Obraz systemu i Wybierz sprzęt. Wprowadź zmiany, a następnie kliknij przycisk Zakończ. AVD zostanie zaprezentowany na stronie Twoje urządzenia wirtualne.

Tworzenie profilu sprzętowego

Menedżer AVD oferuje na nowo zdefiniowane profile sprzętowe dla typowych urządzeń, dzięki czemu można je łatwo dodawać do definicji AVD. Jeśli musisz zdefiniować inne urządzenie, możesz utworzyć nowy profil sprzętowy. Masz możliwość zdefiniowania nowego profilu sprzętowego od początku lub skopiowania profilu sprzętowego na początek. Należy jednak pamiętać, że wstępnie załadowanych profili sprzętowych nie można edytować. Aby od początku utworzyć nowy profil sprzętu, przejdź do strony Wybierz sprzęt i wejdź do nowego profilu sprzętu. Na stronie Konfiguracja profilu sprzętu zmień odpowiednio właściwości profilu sprzętu, a następnie kliknij przycisk Zakończ. Twój nowy profil sprzętu pojawi się na stronie Wybierz sprzęt. Możesz także utworzyć AVD używający profilu sprzętu, klikając Dalej. Alternatywnie, kliknij Anuluj, aby powrócić do strony Twoje urządzenia wirtualne lub wybierz okno dialogowe Wybierz miejsce docelowe wdrożenia. Jeśli chcesz utworzyć profil sprzętu, zaczynając od kopii, musisz przejść do tej samej strony Wybierz sprzęt, wybrać profil sprzętu i kliknąć Klonuj urządzenie. Następnie możesz zmienić właściwości profilu sprzętu zgodnie z potrzebami na stronie Konfigurowanie profilu sprzętu. Gdy skończysz, nie zapomnij kliknąć Zakończ. Twój nowy profil sprzętu pojawi się na stronie Wybierz sprzęt. Możesz dodatkowo utworzyć AVD używający profilu sprzętowego, klikając Dalej. Lub wróć do strony Your Virtual Devices lub Select Deployment Target, klikając po prostu Anuluj. Istnieją pewne operacje, które można wykonać na istniejącym urządzeniu AVD na stronie Twoje urządzenia wirtualne:

1. Aby edytować AVD, kliknij Edytuj ten AVD i zapisz zmiany.

2. Aby usunąć AVD, kliknij prawym przyciskiem AVD i wybierz Usuń. Lub kliknij Menu i wybierz Usuń.

3. Aby wyświetlić powiązane pliki AVD .ini i .img na dysku, kliknij prawym przyciskiem myszy AVD i wybierz Pokaż na dysku. Lub kliknij Menu i wybierz Pokaż na dysku.

4. Aby przejrzeć dane konfiguracyjne AVD, możesz dołączyć do dowolnego raportu o błędzie do zespołu Android Studio, kliknąć prawym przyciskiem myszy AVD i wybrać Wyświetl szczegóły. Lub kliknij Menu i wybierz Wyświetl szczegóły.

Jeśli chcesz edytować istniejące profile sprzętu, na tej samej stronie Wybierz sprzęt możesz wykonać następujące operacje:

1. Aby edytować profil sprzętu, wybierz go i kliknij Edytuj urządzenie. Lub kliknij prawym przyciskiem myszy profil sprzętu i wybierz Edytuj. Następnie wprowadź zmiany.

2. Aby usunąć profil sprzętu, kliknij go prawym przyciskiem myszy i wybierz Usuń.

3. Upewnij się, że nie możesz edytować ani usuwać predefiniowanych profili sprzętowych.

Jeśli chodzi o właściwości AVD, możesz określić następujące właściwości w Tabeli dla konfiguracji AVD na stronie Sprawdź konfigurację. Konfiguracja AVD identyfikuje interakcję między komputerem deweloperskim a emulatorem oraz właściwości, które należy zastąpić w profilu sprzętu. Zazwyczaj właściwości konfiguracyjne AVD zastępują właściwości profilu sprzętowego. Jednak właściwości emulatora ustawione podczas działania emulatora mogą je zastąpić.

Właściwość AVD: Opis

Nazwa AVD : Nazwa AVD. Nazwa może zawierać wielkie lub małe litery, cyfry od 0 do 9, kropki (.), podkreślenia (_), nawiasy (()), myślniki (–) i spacje. Nazwa pliku przechowującego konfigurację AVD pochodzi od nazwy AVD.

Identyfikator AVD (zaawansowane): Nazwa pliku AVD jest pochodną identyfikatora i można użyć identyfikatora do odwoływania się do AVD z wiersza poleceń.

Profil sprzętu : kliknij Zmień, aby wybrać inny profil sprzętu na stronie Wybierz sprzęt.

Obraz systemu : kliknij Zmień, aby wybrać inny obraz systemu na stronie Obraz systemu. Do pobrania nowego obrazu wymagane jest aktywne połączenie internetowe.

Orientacja uruchamiania : wybierz jedną opcję początkowej orientacji emulatora:

• Pionowa — zorientowana wyższa niż szersza.

• Pozioma – zorientowana szersza niż wysoka.

Opcja jest włączona tylko wtedy, gdy jest zaznaczona w profilu sprzętu. Podczas uruchamiania AVD w emulatorze można zmienić orientację, jeśli profil sprzętu obsługuje zarówno pionową, jak i poziomą.

Kamera (zaawansowane): Aby włączyć kamerę, wybierz jedną lub obie opcje:

• Przód – Obiektyw jest zwrócony z dala od użytkownika.

• Tył — soczewka jest skierowana w stronę użytkownika.

Ustawienie Emulowane tworzy obraz generowany przez oprogramowanie, podczas gdy ustawienie Kamera internetowa używa kamery internetowej komputera deweloperskiego do zrobienia zdjęcia. Ta opcja jest dostępna tylko wtedy, gdy jest wybrana w profilu sprzętu.

Sieć: szybkość (zaawansowane) : Wybierz protokół sieciowy, aby określić szybkość przesyłania danych:

• GSM – Globalny System Komunikacji Mobilnej

• HSCSD — dane z przełączaniem obwodów o dużej szybkości

• GPRS – ogólna usługa pakietowej transmisji radiowej

• EDGE – Ulepszone szybkości transmisji danych dla ewolucji GSM

• UMTS – Uniwersalny System Telekomunikacji Mobilnej

• HSDPA — dostęp do pakietów szybkiego łącza w dół

• LTE – długoterminowa ewolucja

• Pełny (domyślny) – przesyłaj dane tak szybko, jak pozwala na to Twój komputer

Sieć: opóźnienie (zaawansowane) : Wybierz protokół sieciowy, aby ustawić, ile czasu (opóźnienia) zajmuje protokół przesyłania pakietu danych z jednego punktu do drugiego.

Emulowana wydajność: grafika : Wybierz sposób renderowania grafiki w emulatorze:

• Sprzęt — użyj karty graficznej komputera, aby przyspieszyć renderowanie.

• Oprogramowanie — emuluj grafikę w oprogramowaniu, co jest przydatne, jeśli masz problem z renderowaniem na karcie graficznej.

• Automatycznie — pozwól emulatorowi wybrać najlepszą opcję w oparciu o twoją kartę graficzną.

Emulowana wydajność: opcja rozruchu (zaawansowane):

• Zimny rozruch — uruchamiaj urządzenie za każdym razem, uruchamiając je ze stanu wyłączonego.

• Szybki rozruch – Uruchom urządzenie, ładując stan urządzenia z zapisanej migawki.

Emulowana wydajność: wielordzeniowy procesor (zaawansowane): Wybierz liczbę rdzeni procesora w komputerze, których chcesz użyć w emulatorze. Użycie większej liczby rdzeni procesora przyspiesza działanie emulatora.

Pamięć i przechowywanie: RAM : Ilość pamięci RAM w urządzeniu. Ta wartość jest ustawiana przez producenta sprzętu, ale w razie potrzeby można ją zastąpić, na przykład w celu szybszego działania emulatora. Zwiększenie rozmiaru zużywa więcej zasobów na komputerze. Wpisz rozmiar pamięci RAM i wybierz jednostki, jedną z B (bajt), KB (kilobajt), MB (megabajt), GB (gigabajt) lub TB (terabajt).

Pamięć i magazyn: sterta maszyny wirtualnej : rozmiar sterty maszyny wirtualnej. Ta wartość jest ustawiana przez producenta sprzętu, ale w razie potrzeby można ją zastąpić. Wpisz rozmiar sterty i wybierz jednostki, jedną z B (bajt), KB (kilobajt), MB (megabajt), GB (gigabajt) lub TB (terabajt).

Pamięć i przechowywanie: pamięć wewnętrzna : Ilość niewymiennej pamięci dostępnej w urządzeniu. Producent sprzętu ustawia tę wartość, ale w razie potrzeby można ją zastąpić. Wpisz rozmiar i wybierz jednostki, jedną z B (bajt), KB (kilobajt), MB (megabajt), GB (gigabajt) lub TB (terabajt).

Pamięć i przechowywanie: Karta SD: Ilość wymiennej pamięci dostępnej do przechowywania danych w urządzeniu. Aby użyć wirtualnej karty SD zarządzanej przez Android Studio, wybierz zarządzaną przez Studio, wpisz rozmiar i wybierz jednostki: B (bajty), KB (kilobajty), MB (megabajty), GB (gigabajty) lub TB (terabajty).). Do korzystania z aparatu zaleca się minimum 100 MB. Aby zarządzać miejscem w pliku, wybierz Plik zewnętrzny i kliknij Określ plik i lokalizację.

Ramka urządzenia: włącz ramkę urządzenia: Wybierz, aby włączyć ramkę wokół okna emulatora, która naśladuje wygląd prawdziwego urządzenia.

Definicja niestandardowej karnacji (zaawansowane) : Wybierz karnację, która kontroluje wygląd urządzenia wyświetlanego w emulatorze. Pamiętaj, że określenie rozmiaru ekranu, który jest zbyt duży dla skóry, może oznaczać, że ekran zostanie odcięty, przez co nie będziesz widzieć całego ekranu.

Klawiatura: włącz wprowadzanie danych z klawiatury (zaawansowane): Wybierz tę opcję, jeśli chcesz używać klawiatury sprzętowej do interakcji z emulatorem.

EDYTOR NAWIGACJI

Nawigacja między różnymi ekranami i aplikacjami to główna część doświadczenia użytkownika. Poniższe zasady stanowią podstawę stabilnego, ale intuicyjnego środowiska użytkownika w różnych aplikacjach. Komponent Nawigacja jest zaprojektowany do domyślnego wykonywania tych zasad, dzięki czemu użytkownicy mogą stosować te same techniki i wzorce w nawigacji podczas przechodzenia między aplikacjami. Świetnie nadaje się również do uproszczenia implementacji nawigacji za pośrednictwem biblioteki, a także pomaga w wizualizacji przepływu nawigacji w aplikacji. Biblioteka posiada szereg korzyści, które obejmują:

• Automatyczna obsługa transakcji fragmentarycznych

• Prawidłowa obsługa do i z powrotem domyślnie

• Domyślne zachowania animacji i przejść

• Głębokie łączenie jako pierwszorzędna operacja

• Implementacja wzorców interfejsu użytkownika nawigacji (szuflady nawigacji i dolna nawigacja) przy niewielkiej dodatkowej pracy

• Zapewnij bezpieczeństwo podczas przekazywania informacji podczas nawigacji

• Narzędzia Android Studio do wizualizacji i modyfikowania przepływu nawigacji w aplikacji

Nawigacja może być również postrzegana jako struktura do nawigacji między „miejscami docelowymi" w aplikacji na Androida, która zapewnia spójny interfejs API, niezależnie od tego, czy miejsca docelowe są traktowane jako fragmenty, działania, czy inne elementy. Zwykle komponent Nawigacja składa się z trzech kluczowych części:

1. Wykres nawigacji (nowy zasób XML): jest to zasób, który przechowuje wszystkie informacje związane z nawigacją w jednej scentralizowanej lokalizacji. Obejmuje to wszystkie miejsca w Twojej aplikacji, zwane miejscami docelowymi, oraz możliwe ścieżki, którymi użytkownik może podążać w Twojej aplikacji.

2. NavHostFragment (widok Layout XML): To specjalny widżet, który dołączasz do swojego układu. Wyświetla różne miejsca docelowe z wykresu nawigacyjnego.

3. NavController (obiekt Kotlin/Java): Jest to temat, który śledzi aktualną pozycję na wykresie nawigacyjnym. Zarządza zamianą zawartości docelowej w NavHostFragment podczas poruszania się po wykresie nawigacyjnym. Edytor nawigacji jest standardową częścią Android Studio 3.3 i nowszych. Dlatego, jeśli nadal używasz Android Studio 3.2, nawigacja będzie dla Ciebie funkcją eksperymentalną i będziesz musiał ją włączyć ręcznie:

W swoim pliku przejdź do Ustawień i wybierz kategorię Eksperyment. Tutaj wystarczy kliknąć opcję Włącz nawigację.

Wtyczki Gradle

Zanim będziesz mógł stworzyć własny wykres nawigacyjny, musisz zadbać o pewne zależności elementów. Dlatego powinieneś dodać następujące zależności dla artefaktów, których będziesz używać w pliku build.Gradle dla swojej aplikacji lub modułu:

dependencies {

– def nav_version = “2.1.0-beta01”

– def nav_version_ktx = “2.1.0-beta01”

//Dla Javy

– implementation “androidx.navigation:navigation-fragment:$nav_version”

– implementation “androidx.navigation:navigation-ui: $nav_version”

// Dla Kotlin

– implementation “androidx.navigation:navigation-fragment-ktx:$nav_version_ktx”

– implementation “androidx.navigation:navigation-ui-ktx:$nav_version_ktx”

Dodatkowo musisz zastosować następujące wtyczki w build.gradle swojego modułu aplikacji:

– apply plugin: “com.android.application”

– apply plugin: “kotlin-android”

– apply plugin: “kotlin-android-extensions”

– apply plugin: "androidx.navigation.safeargs.kotlin"

– Musisz mieć android.useAndroidX=true w swoim pliku gradle.properties zgodnie z Migracją do AndroidX – android.useAndroidX=true

Po skonfigurowaniu wszystkich wtyczek możesz rozpocząć tworzenie komponentu nawigacyjnego dla swojej aplikacji za pomocą jednej czynności i kilku fragmentów. Oto główne komponenty nawigacji, z którymi powinieneś się zapoznać:

• Twoja klasa MainActivity

• Kilka fragmentów

• Wykres nawigacyjny

• Akcja

• Cele podróży

• Wyskakuj do

• Argumenty

• Głębokie linki

• Fragment hosta nawigacji

• Kontroler nawigacji

Pierwsze dwie pozycje na liście są ogólne dla większości innych funkcji, które omówiliśmy wcześniej. Dlatego sugeruje się skupienie się na fragmentach typowych dla tworzenia komponentu Nawigacja, które pomogą Ci powiązać rzeczy z tym, co znasz i uniknąć nieporozumień.

Wykres nawigacji

Wykres nawigacyjny to plik zasobów, który przechowuje wszystkie miejsca docelowe i działania. Wykres służy do wyświetlania wszystkich ścieżek nawigacji Twojej aplikacji. Standardowa nawigacja zazwyczaj składa się z miejsc docelowych reprezentowanych przez miniaturę podglądu oraz działań łączących reprezentowanych przez strzałki, które pokazują, jak użytkownicy mogą nawigować z jednego miejsca docelowego do drugiego. Miejsca docelowe mogą być również postrzegane jako różne obszary treści Twojej aplikacji. A działania to logiczne połączenia między miejscami docelowymi, które reprezentują ścieżki, którymi mogą podążać użytkownicy. Tworzenie wykresu nawigacyjnego to dość prosty proces z kilkoma prostymi krokami do przejścia.

1. Utwórz plik zasobów nawigacyjnych „app_navigation.xml". Nazwa może być dowolna, którą do niej przypiszesz, ale musi być zgodna z podstawowymi regułami nazw plików zasobów (na przykład zawierać tylko małe litery a–z, 0–9 lub podkreślenie). Więc po utworzeniu pliku „app_navigation.xml" twój kod powinien wyglądać tak:

< ?xml version="1.0" encoding="utf-8"? >

< navigation 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:id="@+id/app_navigation"

>

< /navigation >

2. Kolejnymi krokami powinno być zdefiniowanie pierwszego widoku nawigacyjnego, czyli od którego ma się rozpocząć nawigacja i dokąd nawigacja powinna nawigować. Takie szczegóły dowodzą dzięki tagowi o nazwie „przeznaczenie". Ponadto możesz również użyć następujących trzech fragmentów, aby opisać koncepcję miejsca docelowego, w którym pracujesz w tym czasie:

- MyHomeFragment

- MySecondFragment

- MyThirdFragment

3. Po zakończeniu kroku definiowania należy dodać wszystkie fragmenty jako elementy podrzędne do elementu nadrzędnego nawigacji, upewniając się, że przypisano Ci unikalny identyfikator do swoich fragmentów. Aby zilustrować przykładem:

< ?xml version="1.0" encoding="utf-8"? >

< navigation 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:id="@+id/app_navigation" >

< fragment

android:id="@+id/myHomeFragment"

android:name="com.navigation.component.sample.ui.fragments.MyHomeFragment"

android:label=

"fragment_my_home"

tools:layout="@layout/fragment_my_home" >

< /fragment >

< fragment android:id="@+id/mySecondFragment"

android:name="com.navigation.component.sample.ui.fragments.MySecondFragment"

android:label="fragment_my_second"

tools:layout="@layout/fragment_my_second" >

< /fragment >

< fragment android:id="@+id/myThirdFragment"

android:name="com.navigation.component.sample.ui.fragments.MyThirdFragment"

android:label="fragment_my_third"

tools:layout="@layout/fragment_my_third" >

< /fragment >

< /navigation >

Powinieneś zauważyć, że następujące cztery kluczowe parametry zostały dodane w celu wyróżnienia niektórych funkcji identyfikatora w twoim kodzie:

1.android:id: Unikalny identyfikator dla fragmentu, tak jak przypisujemy id do innych widżetów w układzie XML.

2.android:name: Jest to w pełni kwalifikowana nazwa Twojej klasy fragmentów w kotlin/java.

3.android:label: Ciąg znaków identyfikujący fragment.

4.tools:layout: identyfikator pliku zasobów układu z res/layout.

Akcja

Jak już wspomniano, system nawigacji umożliwia również nawigację za pomocą działań. W celu dodania akcji we fragmencie możemy użyć tagu < action/ > wewnątrz tagu < fragment/ >. Jednocześnie możliwe jest zdefiniowanie więcej niż jednej akcji o różnym identyfikatorze. Wewnątrz < action/ > używane są pewne parametry:

• android:id: Unikalny identyfikator akcji, tak jak przypisujemy id do fragmentów.

• app:destination: Unikalny identyfikator fragmentu docelowego oznacza, że ta akcja przeniesie bieżący widok do fragmentu docelowego.

• app:popUpTo: służy do nawigacji wstecz, jeśli aplikacja przeszła z fragmentu A do fragmentu B, a następnie z fragmentu B do fragmentu C. Jeśli chcesz przejść z fragmentu A do fragmentu C, możesz

użyć tej wartości parametru jako fragmentu identyfikatora O. Komponent Nawigacja będzie obsługiwać zarządzanie back stackiem i cykl życia.

Ponadto, jeśli chcesz dołączyć animacje dla transakcji fragmentu, jak określono, powinieneś po prostu utworzyć plik animacji XML w folderze res/anim. Dzięki temu jesteś teraz gotowy do projektowania nawigacji i możesz kontynuować rozmowy nawigacyjne za pomocą NavHostFragment i NavigationController.

NavHostFragment

Aby modyfikować układy działań Nawigacji, musi zawierać specjalny widżet o nazwie NavHostFragment. NavHostFragment wymienia różne miejsca docelowe fragmentów, gdy przechodzisz przez wykres nawigacji. Aby zilustrować przykładem:

LinearLayout

.../ >

< androidx.appcompat.widget.Toolbar

.../ >

< fragment

android:layout_width=

"match_parent"

android:layout_height="0dp"

android:layout_weight="1"

android:id="@+id/my_nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"

app:navGraph="@navigation/app_navigation"

app:defaultNavHost="true"

/ >

<com.google.android.material.bottomnavigation.BottomNavigationView

Jest to uważane za funkcję standardowego działania, która zawiera globalną nawigację, w tym dolną nawigację i pasek narzędzi. Tutaj android:name="androidx.navigation.fragment.NavHostFragment" i app:defaultNavHost="true" łączą przycisk Wstecz systemu z NavHostFragment app:navGraph="@navigation/app_navigation", a także kojarzą NavHostFragment z nawigacją wykres. Ten wykres nawigacji określa wszystkie miejsca docelowe, do których użytkownik może nawigować, w tym NavHostFragment.

Kontroler nawigacji

NavController jest bardzo wygodny, ponieważ gdy wywołujesz metody takie jak nawigacja() lub popBackStack(), tłumaczy te polecenia na odpowiednie operacje frameworka w oparciu o typ miejsca docelowego, do którego lub z którego nawigujesz. Na przykład, gdy wywołujesz nawigację() z miejscem

docelowym aktywności, NavController wywołuje startActivity() w Twoim imieniu. Istnieje kilka sposobów wstawienia NavigationController:

• Fragment.findNavController()

• Widok.findNavController()

• Activity.findNavController(viewId: Int)

Podczas nawigowania do miejsca docelowego za pomocą NavController musisz najpierw połączyć przycisk Nawiguj do miejsca docelowego, aby przejść do miejsca docelowego mySecondFragment (który jest miejscem docelowym, które jest MySecondFragment. Następnie otwórz MyHomeFragment.kt lub plik fragmentu java i aktywuj onClick słuchacza lub jakiekolwiek inne działanie użytkownika, a na koniec nawiguj w następujący sposób:

widok?.findViewById<Button>(R.id.button)

.setOnClickListener(View.OnClickListener {

findNavController().navigate(R.id.action_myHomeFragment_to_mySecondFragment)

SafeArgs

SafeArgs to kolejny składnik nawigacji, który ma wtyczkę Gradle i generuje proste klasy obiektów i konstruktorów dla bezpiecznego dostępu do argumentów określonych dla miejsc docelowych i akcji. Na przykład, ponieważ wcześniej używaliśmy tagu <argument> dla MySecondFragment, SafeArgs wygeneruje klasę o nazwie MySecondFragmentArgs. Jednocześnie SafeArgs mają różne klasy parcelable dla argumentów, a także różne typy danych. Poniżej znajduje się pełna tabela typów danych SafeArgs obsługiwanych przez system Android

GENEROWANIE JAVADOC

W tej ostatniej sekcji omówimy Javadoc, pomocne narzędzie do generowania dokumentacji bezpośrednio z plików źródłowych Java. Ta niewielka część poświęcona dokumentacji Javadoc obejmuje tylko podstawowe informacje, chociaż jest nadzieja, że przyszli programiści będą szukać więcej zasobów i samouczków do nauki języka Java, aby tworzyć bardziej wyrafinowane aplikacje na Androida. Javadoc może być również postrzegana jako narzędzie dostarczane z Java SDK, które umożliwia specjalistom generowanie dokumentacji kodu z plików źródłowych Java. Środowiska programistyczne, takie jak Eclipse, mają wbudowaną obsługę Javadoc i mogą generować przeszukiwalne dane referencyjne HTML z komentarzy w stylu Javadoc. W rzeczywistości odniesienie do Android SDK jest w zasadzie formą dokumentacji Javadoc. Dokumentacja Javadoc wykorzystuje kombinację przetwarzania kodu źródłowego oraz sprawdzania typów i parametrów podczas odczytywania specjalnych znaczników komentarzy, które programista udostępnia jako metadane związane z sekcją kodu. Normalnie komentarz w stylu Javadoc powinien pojawić się tuż przed kodem, z którym jest powiązany. Na przykład komentarz Javadoc dla klasy powinien znajdować się tuż nad deklaracją klasy, a komentarz dla metody powinien znajdować się tuż nad deklaracją metody. Dodatkowo każdy komentarz powinien zaczynać się krótkim opisem, po którym następuje opcja dłuższego opisu. Następnie możesz dołączyć kilka różnych tagów metadanych, które należy podać w określonej kolejności. Niektóre ważne tagi to:

@author - kto napisał ten kod

@version -kiedy została zmieniona

@param - opisz parametry metody

@return - opisz zwracane wartości metody

@throws - opisz zgłoszone wyjątki

@see - link do innych powiązanych elementów

@since - opisz kiedy kod został wprowadzony

@deprecated - opisz przestarzały element i jaką alternatywę użyć zamiast tego

Jednocześnie pamiętaj, że możesz tworzyć własne niestandardowe znaczniki do wykorzystania w dokumentacji. Jak wspomniano, dokumentacja Javadoc składa się głównie z komentarzy umieszczanych zwykle nad klasami, metodami lub polami. Dlatego ważne jest, aby najpierw nauczyć się manipulować i modyfikować komponenty komentarza.

Dodawanie nowego komentarza

Dokumenty Javadoc można dodać po prostu za pomocą automatycznego uzupełniania komentarzy, które jest domyślnie włączone. Powinieneś wpisać /** przed deklaracją i nacisnąć Enter, IDE automatycznie uzupełni resztę komentarza do dokumentu.

Dodaj dokument Javadoc za pomocą działań kontekstowych

Możesz również dodać dokumentację Javadoc poprzez akcje kontekstowe, umieszczając karetki przy deklaracji w edytorze i wybierając opcję Dodaj dokumentację Javadoc z listy. Co więcej, istnieje również dodatkowa opcja dla komentarzy do metod, ponieważ nowy kod pośredniczący zawiera wymagane tagi (@param tags dla każdego parametru metody, @return lub @throws). Jednocześnie w Kotlinie tagi @param i inne nie są generowane, ponieważ zalecany styl wymaga włączenia opisu parametrów i zwracania wartości bezpośrednio do komentarza dokumentacji.

Wyłącz automatyczne komentarze

Aby wyłączyć automatyczne komentarze, przejdź do okna dialogowego Ustawienia/Preferencje, wybierz opcję Inteligentne klawisze edytora i usuń zaznaczenie pola wyboru Wstaw skrót komentarza do dokumentacji.

Napraw dokument Javadoc

W przypadku zmiany sygnatury metody, IDE podświetli tag, który nie pasuje do sygnatury metody i zaproponuje szybką poprawkę. Jeśli się z tym zgadzasz, po prostu naciśnij Alt + Enter, aby zastosować tę poprawkę. Możesz również zaktualizować istniejący komentarz Javadoc, aby uwzględnić zmiany w deklaracji za pomocą akcji Napraw komentarz doc. W tym celu umieść karetkę w klasie, metodzie, funkcji lub polu i naciśnij Ctrl+Shift+A. Wpisz komentarz do dokumentu naprawczego i naciśnij klawisz Enter. Co więcej, możesz użyć akcji Napraw komentarz do dokumentu, aby dodać brakujący fragment dokumentacji z odpowiednimi znacznikami: Umieść karetkę w klasie, metodzie lub funkcji i wywołaj akcję.

Renderuj dokumenty Javadoc

Platforma Android umożliwia renderowanie dokumentów Javadoc w edytorze. Zazwyczaj wyrenderowane komentarze są łatwiejsze do odczytania i nie przeciążają kodu dodatkowymi tagami. Aby dostosować wyrenderowany widok, należy kliknąć Przełącz wyrenderowany widok w rynnie obok niezbędnego komentarza dotyczącego dokumentacji (lub nacisnąć Ctrl+Alt+Q). Następnie kliknij opcję

Edycja grafiki, aby zmodyfikować komentarz. Wyrenderowane dokumenty Javadoc pozwalają również klikać łącza, aby przejść do stron internetowych, do których się odnoszą, lub przeglądać szybką dokumentację do wspomnianych tematów. Jeśli chcesz zmienić rozmiar czcionki, kliknij prawym przyciskiem myszy dokument Javadoc w edytorze i wybierz opcję Dostosuj rozmiar czcionki z menu kontekstowego. Pamiętaj, że renderowane komentarze używają tego samego rozmiaru czcionki, co wyskakujące okienko szybkiej dokumentacji.

Domyślnie renderuj dokumenty Javadoc

Możesz także ustawić IDE, aby zawsze renderować dokumenty Javadoc w edytorze. W tym celu wystarczy kliknąć prawym przyciskiem myszy ikonę w rynnie i wybrać opcję Włącz opcję Renderuj wszystko. Alternatywnie w oknie dialogowym Ustawienia/Preferencje Ctrl+Alt+S wybierz opcję Wygląd i włącz opcję Renderuj komentarze dokumentacji. Jeśli chcesz edytować wyrenderowane dokumenty Javadoc, możesz kliknąć ikonę Przełącz wyrenderowany widok w rynnie obok komentarza.

Wygeneruj dokumentację Javadoc

Android Studio zapewnia świetne narzędzie, które umożliwia generowanie dokumentacji Javadoc dla twojego projektu. Z menu głównego wybierz opcję Narzędzia i Generuj JavaDoc. W otwartym oknie dialogowym wybierz zakres plików lub katalogów, dla których chcesz wygenerować odniesienie, i ustaw katalog wyjściowy, w którym chcesz umieścić wygenerowaną dokumentację. Katalog wyjściowy jest polem obowiązkowym: nie można wygenerować pliku Javadoc, dopóki jest on pusty. Zamiast tego użyj suwaka, aby określić poziom widoczności komponentów, które zostaną uwzględnione w wygenerowanej dokumentacji. Możesz wybrać jedną z następujących opcji:

• Prywatne: dołączyć wszystkie klasy i członków do odniesienia

• Pakiet: zawiera wszystkie klasy i członków z wyjątkiem prywatnych

• Chroniony: obejmuje publiczne i chronione klasy i członków

• Publiczny: obejmować tylko klasy publiczne i członków

Po ustawieniu zaleca się również określenie ustawień regionalnych (na przykład en_US.UTF-8), argumentów wiersza poleceń i maksymalnego rozmiaru sterty. Po zakończeniu kliknij OK, aby wygenerować odniesienie.

Zobacz dokumentację Javadoc w edytorze

Na platformie możesz przeglądać dokumentację Javadoc dla dowolnego sygnatury symbolu lub metody bezpośrednio z edytora. Aby włączyć taką funkcję, należy skonfigurować ścieżki dokumentacji bibliotek lub dodać pobrane dokumenty Javadoc do IDE. Po prostu przesuń mysz na wymagany symbol, aby wyświetlić jego dokumentację, lub umieść karetkę na symbolu i naciśnij Ctrl+Q, aby móc przeglądać dokumentację i przełączać się między wyskakującym okienkiem a paskiem narzędzi. Następnie kliknij ikonę Pokaż menu opcji w wyskakującym okienku, aby zmienić rozmiar czcionki, wyświetlić pasek narzędzi szybkiej dokumentacji lub przejść do kodu źródłowego.

Rozwiązywanie problemów

Zazwyczaj najbardziej znanym błędem Javadoc, który pojawia się, jest błędna nazwa lokalizacji: en_US.UTF-8. Na szczęście istnieje dość proste rozwiązanie, o którym należy zawsze pamiętać. Zacznij od wyczyszczenia pola Locale, a gdy to zrobisz, dodaj funkcje -encoding utf8 -docencoding utf8 -charset utf8 w polu Inne argumenty wiersza poleceń. Tutaj segment -encoding określi kodowanie plików źródłowych. Część – docencoding powinna identyfikować kodowanie wyjściowych plików HTML,

podczas gdy -charset oznacza zestaw znaków określony w sekcji nagłówka HTML plików wyjściowych. Po zakończeniu wstawiania wystarczy ponownie załadować plik, aby sprawdzić, czy błąd został naprawiony. Podsumowując ten rozdział, opisaliśmy najpopularniejsze narzędzia programistyczne dla Androida i przystąpiliśmy do wdrażania niektórych ich funkcji. Zdefiniowaliśmy podstawowe i opcjonalne instrumenty, takie jak SDK Manager, AVD Manager i Navigation Editor. Wprowadziliśmy wymagane biblioteki do tworzenia aplikacji na Androida, debugger, emulator, interfejsy API i przykładowe projekty z kodem źródłowym.